2009

Construction legal support for differing site conditions (DSC) through statistical modeling and machine learning (ML)

Tarek Said Mahfouz
Iowa State University

Construction legal support for differing site conditions (DSC) through statistical modeling and machine learning (ML)

by

Tarek Said Mahfouz

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Civil Engineering (Construction Engineering and Management)

Program of Study Committee:
Amr Kandil, Major Professor
Edward J. Jaselskis
Kelly C. Strong
Kejin Wang
Nick Pendar

Iowa State University

Ames, Iowa

2009

DEDICATION

I would like to dedicate my doctorate research to my parents for their absolute love, care and support, my wife Lamia Moustafa for her full support and understanding, and my sons Youssef and Adam for all the joy that they bring to my life.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 5

LSIT OF FIGURES

CHAPTER 1

CHAPTER 2

ACKNOWLEDGMENT

I would like to express my gratitude to my mentor and academic advisor Dr. Amr Kandil for all his valuable advice, guidance, and support during my doctoral program. His continued support academically and morally during and beyond my studies is an asset that I will continue to depend on. I would like also to extend my appreciation to Dr. Edward J. Jaselskis, Dr. Kelly C. Strong, Dr. Kejin Wang, and Dr. Nick Pendar for their services on my thesis supervisory committee and all their guidance and constructive feedback. I would also like to gratefully acknowledge and thank Dr. Safwan Abbas Khedr Professor of Construction Engineering at the American University in Cairo for his moral and perennial support and professional advice through and beyond my study period at Iowa State University.

I would like to thank my parents and wife for their support, understanding, encouragement, and patience that made my Ph.D. studies possible. I would like also to extend my gratitude to my dear friend and colleague Mohammed Al Qady for his valuable support during my Ph.D. studies. Finally, I want to sincerely thank Dr. Konstantina Gkritza and Dr. Yasser El Manzalawy from Iowa State University for their valuable assistance.

ABSTRACT

Construction is one of the industries with a major contribution to the nation's economy. It is estimated that the world construction market has reached US $5.5 trillion at the end of 2007 (Harmon 2003). In the U.S., the construction industry employs 7.5 million full and part time employees and contributes to nearly $1.2 trillion to its economy making it the largest single production sector (El-adaway 2008). With that magnitude, it is not only considered as the backbone of the nations' economy, but also a significant indicator of its advancement, efficiency, and success. However, due to the dynamic nature of the construction industry and the increasing sophistication and complexity of construction projects, its contribution is negatively affected by the increasing number of disputes. Unfortunately, the rate and frequency of conflicts has risen with the growing complexity of projects. Modern construction projects require increasingly sophisticated construction methods and extensive interaction of diversified parties, thus enhancing the likelihood of conflicts and disputes.

Construction disputes are ultimately resolved in courts unless a private construction contract calls for other resolution mechanisms. In fact, some in the construction industry prefer litigation; however, their preference comes at great cost. Despite the numerous advantages of litigation, which includes being the most formal and binding process, it has two main shortcomings, which make the process undesirable and unsupportive of the growth and development of the construction industry. First, depending on the jurisdiction, complex construction disputes may take anywhere from two to six years before they reach trials. Second, the prolonged,

detailed, factual discovery process makes litigation very expensive due to the need for specialized personnel with extensive legal knowledge and construction experience, a combined skill set that is not widely available in the industry. In order to overcome these major drawbacks that impact the construction industry's advancement and contribution to the nations' economy, legal decision support systems are needed to effectively and efficiently mitigate these shortcomings and in turn allow for better control and management of construction projects.

In construction disputes the initiation of the conflict can be attributed to a number of reasons including: change orders, escalation, and differing site conditions, etc. Each of these reasons leads to a separate method for addressing and handling the disputes and accordingly, each reason can be considered as a different dispute type. Among these types, one of the most important and frequently occurring disputes is Differing Site Conditions (DSC) which results from contractors encountering conditions materially different from those expected or described by the owner. This warrants special attention to this kind of dispute due to their potential for deviating construction projects from their planned time and cost.

A number of researchers in Artificial Intelligence (AI) fields have developed tools and methodologies for modeling judicial reasoning and predicting the outcomes of construction litigation cases in an attempt to provide the above mentioned decision support capabilities. Despite the significant contributions of these systems to the advancement of legal decision support capabilities in construction, their success was limited because they were not based on a detailed analysis of legal concepts that govern litigation outcomes.

Consequently, the objective of this dissertation is to provide a coherent and integrated methodology for construction legal decision support for Differing Site Conditions (DSC) disputes through statistical modeling and machine learning. To attain this goal, the current study designed and implemented a 4 step methodology targeting the following goals: (1) to extract a comprehensive set of legal factors that govern DSC litigation outcomes in the construction industry; (2) to devise a litigation prediction model for DSC disputes in the construction industry based on the extracted set of legal factors; (3) to create a methodology for automated extraction of significant legal factors that governs DSC litigation outcomes from case documents; and (4) to develop an automated retrieval model for identifying DSC precedent cases from a large corpus based on similarity to newly introduced ones. The 4 steps of this methodology were implemented incrementally, and each step relied on the outcome of its predecessor.

First, a comprehensive set of significant legal factors that govern DSC litigation cases verdicts were extracted through statistical modeling. Binary Probit and Logit Choice Models were developed (a) to identify the effect of each extracted factor on the prediction of the winning party; (b) to identify the best combination of factors with the highest significance on the prediction model; and (c) to perform a sensitivity analysis to prioritize the most significant legal factors. Among the main findings of this step are (1) in general, cases in which the Federal Government is a party of the dispute, judgments are in favor of the government (owner) over contractor; (2) “the presence of evident facts that the encountered conditions caused a change in the nature and cost of the contract” had the highest impact among

variables causing a decrease in the prediction of judgment in favor of the owner, and causing an increase of 17.77% in prediction on favor of the contractor; (3) "the presence of evident facts that the specifications included a warning against the presence of DSC from those conveyed in the contract documents" caused the highest increase in the prediction of judgment in favor of the owner amounting to an increase of 56.56%; and (4) the development of Binary Probit and Logit Choice Models extracted a joint set of 13 statistically significant legal factors related to DSC disputes in the construction industry. This set provided the grounds for the other three steps of the current research methodology.

Second, an automated litigation prediction model for DSC disputes in the construction industry through machine learning was developed based on the identified factors in the first step. The framework under this step incorporates analysis of different machine learning methodologies including support vector machines (SVM), Naïve Bayes (NB), and rule induction classifiers like Decision Trees (DT), Boosted Decision Trees (AD Tree), and PART. Ten machine learning models were developed using these machine learning methodologies to evaluate the best methodology for predicting litigation outcomes. The analysis of all developed models showed that the SVM Kernel Polynomial 3rd degree model has the best performance. This model attained an overall prediction accuracy of 98%.

Third, an automated significant legal factors extraction model for DSC disputes in the construction industry through machine learning was developed. The framework under this step (1) developed 24 machine learning models in which 4 weighting schemes namely Term Frequency (tf), Logarithmic Term Frequency (ltf),

Augmented Term Frequency (atf), and Term Frequency Inverse Document Frequency (tf.idf) were implemented for each type of classifier; and (2) developed two C++ algorithms for the preparation of the corpus and implementation of the required weighting mechanisms. The highest prediction rate of 84% was attained by NB classifier while implementing tf.idf weighting. The model was further validated by testing newly un-encountered cases, and a prediction precision of 81.8% was attained.

Finally, the fourth step of the methodology developed an automated machine learning model for the retrieval of supporting DSC precedent cases from large corpi. This step, therefore, (1) implemented Latent Semantic Analysis algorithm; and (2) developed 9 reduced feature spaces with feature sizes of 5, 10, 15, 20, 100, 200, 300, 400, and 500 for analysis and validation of the implemented algorithm. Among the findings of this step are (1) low dimension reduced feature spaces are more representative of documents closely related to the domain problem; (2) high dimension reduced feature spaces, are more representative to domain problems modeling dispersed and unrelated document collections; and (3) LSA reduced feature space of 10 features is the best reduced feature space to adopt for automating the extraction of similar DSC cases from a large corpus.

The main research developments of this research contribute to the advancement of the current state of the art in construction legal decision support and Knowledge Management (KM) in the construction legal domain by developing much needed systems for (1) litigation outcomes prediction; (2) automated legal factor extraction; and (3) automated precedent case retrieval. Those developments hold

promises to decrease the costs of legal experts in the construction industry by decreasing time spent on non-value adding tasks such as documents analysis, and offering initial estimates of the legal situation of a disputing party; (2) decrease the time consumed in the litigation processes; (3) facilitate access to legal knowledge needed by practitioners in the construction industry; (4) provide a better understanding of the legal consequences of decision making in the construction industry; and (5) provide solid support documents and probabilistic measures about the strength of a legal situation of a disputing party for better decision making about resolution mechanisms. All these expected outcomes have promising potential to decrease the negative impact of disputes on the construction industry, and thereby creating significant opportunities for the growth of this important sector of the US economy.

CHAPTER 1

INTRODUCTION

1.1 Overview:

The famous English lawyer, statesman and philosopher Francis Bacon (1561-1626) said that "Man seeketh in society comfort, use and protection," and law has always been a very crucial tool for achieving these important human societal objectives. From the time of Hammurabi's code (the first known legal code in history); the way in which humans live has been structured by laws and legal systems that regulate how humans operate within the bounds of civil society (Johns 2007). Laws are rules and customs that the members of a society regard as binding and are upheld and enforced by a judiciary (Britannica 2007). As society evolved, special branches of law developed to govern different aspects of commerce and industry. Of those specialty laws, construction law has evolved as an important field due to the importance of the construction industry to modern society. Construction is one of the major sectors of industry that has a major impact on the nation's economy. It is estimated that the world construction market has reached US $5.5 trillion at the end of 2007 (Harmon 2003). Construction works represent approximately 4.6% of the nation's Gross Domestic Product (El-adaway 2008). In the U.S., the construction industry employs 7.5 million full and part time employees and contributes to nearly US $1.2 trillion to its economy making it the largest single production sector (El-adaway 2008). The magnitude of this contribution illustrates the importance of the branch of law that regulates this industry. The importance of

construction law also stems from the unique nature of each construction project which requires further binding regulations that are construed in construction contracts and contract conditions (Fisk 2000). Laws and contract clauses represent the assuring protocols that protect the rights of each participating party in a construction project. However, it is a fact that the execution of each construction project often takes place under significantly different conditions from those under which it was conceived (Caldas et al. 2002). This implies that frequently projects are constructed under conditions that differ from those under which contracts have been construed. This dynamic nature of the modern construction projects makes it virtually impossible to complete a large construction project without having disputes between project parties (Merrill 2006).

The efficiency of the construction industry has always been negatively impacted by conflicts and disputes that unfold and oftentimes escalate as projects progress (Merrill 2006). Unfortunately, the rate and frequency of conflicts has risen with the growing complexity of projects. Modern construction projects require increasingly sophisticated construction methods and extensive interaction of diversified parties, thus enhancing the likelihood of conflicts and disputes (Caldas et al. 2002, Arditi et al. 1999). In large, complex projects, the impact of these conflicts can be very significant, both in terms of the high costs directly associated with the process of dispute settlement as well as the cost of the delays and possible shutdown of the project while disputes are being settled (Levin 1998).

As stated by Jervis and Levin (1988) disputes will ultimately have to be resolved in courts unless a private construction contract calls for a binding arbitration

clause. In fact, some in the construction industry prefer litigation; however, their preference comes out at great cost. Despite the numerous advantages of litigation, among which it is being a formal and binding process, it has two main shortcomings, which make the process undesirable and inefficient for the development of the construction industry. First, depending on the jurisdiction, a complex construction dispute may take anywhere from two to six years before it reaches trial. Treacy (1995) demonstrated that within a period of 8 years from 1984 to 1992, the number of construction litigation cases that have been in courts with no final decision for three or more years have doubled. In addition, court decisions may be appealed if any of the involved parties wish to contest the first judgment. Escalation mechanisms of litigation cases for appeals differ from one jurisdiction to the other. Generally, court decisions are considered to be final if not appealed to or reversed by decisions of a higher court. Second, the prolonged, detailed, factual discovery process makes litigation exceedingly expensive due to the need of specialized personnel with extensive legal knowledge and construction experience, a combined skill set that is not widely available in the industry (Jervis and Levin 1988). Practitioners are few in number and thus command high salaries (Cobb and Diekmann 1986). A study indicated that fees paid to lawyers and experts in litigation had increased 425% within the period of 1979–1990 while settlement and verdicts had increased only 309% (Marcotte 1990). It costs more to get less in litigation than ever before (Callahan et al. 1990). In addition; Ren et al. (2001) pointed out that 52% of all construction projects in UK end up with a claim that could reach up to £1.2 billion. In US and Canada, 50% of the construction projects claims represent an

extra value of 30% of the original contract price, 33% reached up to 60% of the original contract price, and others exceeded 100% of the original contract price as reported by Cheeks 2003. Peña-Mora et al (2003) estimated the total annual cost of construction conflicts and disputes in the U.S. to be $5 billion.

1.2 Problem Statement

The increasing numbers of claims and disputes have hampered the advancement as well as the growth of the contribution of the construction industry to the economy. The negative effects of claims and disputes on the construction industry include: (1) the increase in contingencies included in project bids leading to the increase of contract values; (2) the decrease in the effectiveness of project management causing projects to cost more and take longer; (3) the loss of direct communication between involved parties in construction projects which potentially leads to additional project inefficiencies; and (4) the deterioration of ongoing and future relations between construction parties leading to loss of confidence in current and future works (Peña-Mora et al 2003). In addition to these direct impacts of claims and disputes, the previously highlighted disadvantages of litigation as a method of dispute resolution are causing parties in construction disputes to (1) face project delays not only due to long periods required for reaching a final verdict, but also due to potential project shutdowns; and (2) carry high financial burdens due to high costs and limited number of practitioners needed in the construction industry. This necessitates a close look at the dominant judicial system of the United States which is Anglo-Saxon legal system, a crucial aspect of which is reliance on legal

precedence (Elhadi 2001). Precedence could be defined as the reliance of a court on decisions in previous relevant cases. Court rulings in the United States are archived in highly sophisticated electronic information storage and retrieval systems which (1) are extremely complex; (2) are time-consuming; and (3) require legal knowledge and expertise for effective utilization (Kowalski and Maybury 2000). This makes it very difficult for information seekers, especially construction practitioners, to make legal decisions or evaluate their legal position in case of conflicts.

Consequently, as claims and disputes increase, the construction industry struggles to find ways to provide legal decision support capabilities to aid in dispute mitigation and resolution. Recently, Artificial Intelligence (AI) is being used to address increasingly sophisticated and diverse problems in the construction industry. It has been extensively utilized to enhance information models, document integration, inter-organizational systems, and expert systems (Labidi 1997). A number of researchers in AI fields have developed tools and methodologies for modeling judicial reasoning and predicting the outcomes of construction litigation cases in an attempt to provide the above mentioned decision support capabilities. Attempts ranged from initial rule based systems (RBR) (Diekmann and Kruppenbacher 1984, Cobb and Diekmann 1986, and Kim 1987), to artificial neural networks systems (ANN) (Arditi 1998, Chau 2005, and Chau 2006a), case based reasoning systems (CBR) (Arditi and Tokdemir 1999, and Chau 2006b), and hybrid systems (Arditi and Pulket 2005, and Chen and Hsu 2007). Despite the significant contributions of these systems to the advancement of legal decision support capabilities in construction, their success was limited because they were not based

on a detailed analysis of legal concepts that govern litigation outcomes. The significance of this drawback stems from the fact that the success of decision support systems is highly dependent on its input parameters. In an attempt to provide advanced construction legal decision support capabilities for the construction industry, a detailed analysis of the legally governing factors that are utilized by judges in resolving such disputes must be performed. In addition, the legal precedence of these factors to one another and to others utilized in the development of earlier systems must be explored.

The input parameter analysis is an important initial step in creating advanced construction legal decision support capabilities that needs to be followed with a thorough investigation of AI algorithms and methodologies. The importance of this investigation stems from the fact that the success of previous construction legal decision support system was limited due to the capabilities of the utilized AI algorithms. For example, the success of some of the RBR models or expert systems in legal decision support was limited due to (Bubbers and Christian 1992): (1) the failure to deduce all necessary rules upon which the system operates; and (2) the assumption of the existence of a full domain model that captures all required rules about a specific claim type. ANN systems achieved improvements over RBR, but as Watson (1997) highlighted, their excessive training limits their effectiveness.

The investigation of advanced AI methodologies and algorithms needs to a have a target data set identified for utilization in testing, analysis, and verification. While a lot research studies targeted construction claims and disputes in general, focusing on a single type of dispute offers the ability to analyze the particular details

of the dispute, and thereby enhancing the overall construction legal decision support capabilities provided. One of the most significant types of claims and disputes in construction projects is the Differing Site Conditions (DSC) disputes that deal with contractors facing site conditions that differ materially from those expected or described in contract documents. The focus on this type of dispute in the development of advanced construction legal support capabilities will provide much needed support in this common and very important type of dispute, without loss of generality in the approach used for creating those capabilities. Therefore, in order to address the increasing need to provide legal decision support in construction claims and disputes in general and in DSC disputes in particular, the main focus of this study is to thoroughly investigate four important domain problems namely: (1) analyzing and identifying significant legal concepts that govern litigation cases related to DSC; (2) developing litigation prediction models related to DSC cases; and (3) enabling automated extraction of legal concepts affecting litigation outcomes of DSC disputes; and (4) exploring and evaluating the suitability of developing an automated assisting tools for extracting related president cases from large corpi.

1.3 Research Objectives

New advancements in the AI field present real opportunities for advancing the management of legal knowledge in the construction industry and developing an innovative construction legal support methodology. *In order to seize these opportunities, the main goal of this dissertation is to develop an integrated and coherent methodology for Construction Legal Decision Support through Statistical*

Modeling and Machine Learning (ML). Since construction disputes cover a wide range of causes, the focus of this dissertation, to achieve the above general goal, will be directed towards *Differing Site Conditions (DSC)* disputes in the construction industry. To accomplish this, the objectives of this study, along with its relevant research questions and hypothesis are summarized as follows:

Objective 1: To create a solid point of departure for the current study through investigating recent research development in the areas of legal decision support, statistical modeling, and machine learning in the construction and legal domain.

Research Questions: (a) What are the new requirements imposed by new and emerging contracting methods on construction decision makers? (b) What are the characteristics of DSC clauses imposed by formal contract documents like American Institute of Architect (AIA), Federal Acquisition Regulations (FAR), and Fédération Internationale Des Ingénieurs-Conseils, French for the International Federation of Consulting Engineers (FIDIC)? (c) What are the requirements imposed on construction decision makers due to DSC? (d) What are the capabilities current construction legal support systems? and (e) What are the different types of reasoning implemented by machine learning techniques and their implementation?

Hypothesis: The investigation of (1) the latest research developments in the area of legal decision support litigation outcome prediction, and text mining applications in the construction and legal domains; and (2) the history, types, and

legal context of DSC clauses in the construction industry can provide a better definition of the domain problems investigated under this study.

Objective 2: To identify, quantify, and measure the impact of significant legal factors on the prediction of outcomes of DSC disputes in the construction industry.

Research Questions: (a) What are the legal factors upon which judges base their verdicts in DSC cases within the construction industry? (b) How does each legal factor affect the judgment? (c) What are the legal factors that favor the side of an owner over a contractor and vice versa? (d) What is legal precedence of these factors to one another? (e) What are the statistically significant legal factors related to DSC disputes in the construction industry? and (f) Which statistical modeling techniques should be investigated further for implementation in the current study?

Hypothesis: Statistical models can be utilized to identify, quantify, and measure the impact of legal factors on outcomes prediction of DSC. Those statistical models would be able to produce a set of factors that could be utilized for developing efficient and effective construction litigation outcome prediction models.

Objective 3: To develop litigation outcome prediction models for DSC disputes in the construction industry using Artificial Intelligence (AI) and Machine Learning (ML).

Research Questions: (a) What are the capabilities and constrains of the available AI and ML algorithms? (b) What are the decision variables that need to be considered in the model? and (c) Which ML modeling techniques can yield the highest accuracy in predicting litigation?

Hypothesis: Recent AI and ML algorithms can be used to create effective litigation outcome prediction models for DSC disputes in the construction industry. These litigation outcome prediction models could (1) provide a better understanding to decision makers about the legal consequences of their decisions; (2) save time and cost related to the need of specialized legal expertise (3) help to relieve the negative consequences associated with lengthy claims and disputes resolution in the construction industry.

Objective 4: To create an automated methodology for the extraction of legal factors from textual DSC cases in the construction industry using AI and ML.

Research Questions: (a) Which weighting and search methodologies are best suited to create this methodology? (b) What are the capabilities and constraints of the available AI and ML algorithms and methodologies? and (c) Which ML modeling techniques should be utilized for creating this automated methodology?

Hypothesis: The automation of significant legal factors extraction from DSC using AI and ML algorithms is both feasible and effective. This automated legal factor extraction methodology can facilitate the process of reviewing and analyzing construction dispute documents and increase the effectiveness of the use of legal experts on these cases.

Objective 5: To create AI and ML models for automating the extraction of relevant precedent cases from large corpi.

Research Questions: (a) Which weighting and search methodologies are best suited to create this model? (b) What types of AI and ML algorithms and mechanisms are best suited for searching in large corpi? (c) What are the capabilities and constraints of these algorithms in DSC case document corpi? (d) How can a large corpus of DSC case documents be represented in a feature space? (e) What is the optimal feature space size for the DSC case document corpus? (f) What are the DSC case features that need to be considered in the model? and (g) Which ML algorithm should be used for creating the DSC precedent case automated extraction model?

Hypothesis: The automation of the relevant DSC case retrieval from large corpi using AI and ML algorithms is both feasible and practical. This automated precedent case extraction model will provide a much needed tool for professionals in construction industry for seeking and retrieving legal knowledge.

1.4 Research Significance

The proposed research developments are designed to create construction legal decision support capabilities for DSC disputes in the construction industry. The primary goals of this research are (1) to identify significant legal factors in DSC disputes; (2) to develop a litigation outcome prediction model for DSC disputes in the construction industry; (3) to automate the extraction of significant legal concepts that affects litigation outcomes of DSC disputes in the construction industry from textual

representations of newly encountered cases; and (4) to develop an automated extraction tool for relevant precedent DSC cases from large corpi. The application of these research developments holds strong promise to support decision makers in the construction industry in understanding the consequences of their legal decision regarding DSC disputes through knowledge of their odds of winning or losing a case at the litigation level. This will consequently lead to more informed decisions about escalating disputes to litigation or settling through other means of dispute resolution mechanisms like amicable settlements, mitigation, or arbitration. These advancements can also lead to minimizing costs of legal expert support in dispute presentation and defense. Finally, the proposed developments can provide assisting tools to retrieving supporting precedent cases to encountered DSC disputes in the construction industry. This tool is not only anticipated to provide support to construction practitioners but also to legal experts in this field.

1.5 Research Methodology

In order to achieve the aforementioned objectives, the research work in this study is organized into five main research tasks that are designed to: (1) conduct a comprehensive literature review of the latest research developments in the construction and legal domain related to litigation outcomes prediction and machine learning applications; (2) identify and quantify significant legal factors that affect DSC litigation outcomes in the construction industry through statistical modeling; (3) develop a litigation outcome prediction model for DSC disputes in the construction industry using AI and ML; (4) automate significant legal factor extraction model from

textual representations of DSC cases using AI and ML; and (5) automate relevant precedent cases extraction from large corpi of DSC cases. these main tasks and their research products are shown in Figure 1.1. In addition, the following is a brief account of these main tasks.

1.5.1 Task 1: Conduct a Comprehensive Literature Review

The objective of this task is to investigate the latest research developments to form a solid point of departure for the present study. The work under this research task is organized in the following four sub-tasks that investigate:

1. New and emerging litigation outcome prediction models in the construction and the legal domains.
2. New and emerging Case Based Reasoning (CBR) models in the construction industry.
3. The field of Natural Language Processing (NLP) and its applications in the construction research.
4. Machine Learning techniques and their application in construction and other fields.
5. The history, types, nature, characteristics, application, risk allocation, and legal concepts behind DSC contract clauses.

1.5.2 Task 2: Identify and Quantify Significant Legal Factors that Affect DSC Litigation Outcomes in the Construction Industry

The purpose of this task is to identify and quantify the impact of significant legal factors that affect litigation prediction outcomes of DSC disputes in the

construction industry though statistical modeling. The research work under this task is organized in the following five sub-tasks that:

1. Develop a corpus of construction DSC precedent cases.
2. Identify the set of legal factors that constitute the bases of judgments in construction DSC cases.
3. Create statistical models that relate the likelihood of a DSC cases being judged in favor of one party over the other to the identified set of legal factors.
4. Explore possible combinations of factors to find the best combination that yields the highest significance to outcome prediction.
5. Perform a sensitivity analysis to prioritize the identified significant legal factors.

1.5.3 Task 3: Develop a litigation Outcome Prediction Model for DSC Disputes in the Construction Industry.

The purpose of this task is to develop a litigation outcome prediction model for DSC disputes in the construction industry through machine learning. The research work under this task is organized in the following five sub-tasks that aim to:

1. Evaluate the different types of reasoning (Induction, Deduction, and Abduction) implemented by machine learning techniques and decide on the appropriate one for the current task.

2. Investigate and evaluate the effectiveness of machine learning techniques, namely support Vector Machines (SVM), Naïve Bayes classifiers, rule Based Induction Classifiers like Decision trees and ADTrees, and decide on the appropriateness of their use for the current task.
3. Determine the appropriate data representation and transformation method for creating the DSC litigation outcome prediction model.
4. Determine the variables that need to be considered for the model development.
5. Develop and evaluate the effectiveness of different prediction models to decide on the best one to be adopted by this task.

1.5.4 Task 4: Automated Extraction of Significant Legal Factors

The purpose of this task is to automate the extraction of the significant legal factors identified in task 1.5.2 and utilized to create the prediction models in task 1.5.3 from textual representation of DSC cases in the construction industry using AI and ML algorithms. The research work under this task is organized in the following five sub-tasks that:

1. Determine the appropriate weighting and representation mechanisms for textual corpi of cases.
2. Develop an algorithm for the implementation of the chosen weighing and representation mechanisms.

3. Develop machine learning models (SVM, Naïve Bayes, and Rule Inductive) to automate the extraction of significant legal factors based on the chosen weighting and representation mechanisms.
4. Cross-validate the developed models through to decide on the best one to adopted for the current task.
5. Test and validate the best developed model with a set on newly un-encountered cases.

1.5.5 Task 5: Automated Extraction of Precedent DSC Cases.

The purpose of this task is to automate the retrieval of relevant DSC precedent cases from large corpi based on similarity measures to other cases using ML algorithms and NLP. The research work under this task is organized in the following three sub-tasks that aim to:

1. Investigate and evaluate Latent Semantic Analysis (LSA) methods for the retrieval of DSC precedent cases.
2. Select the best feature space size to be adopted for the developed automated extraction method through the development and testing of variety of feature space sizes.
3. Test and validate the developed models to select the one yielding the best results.

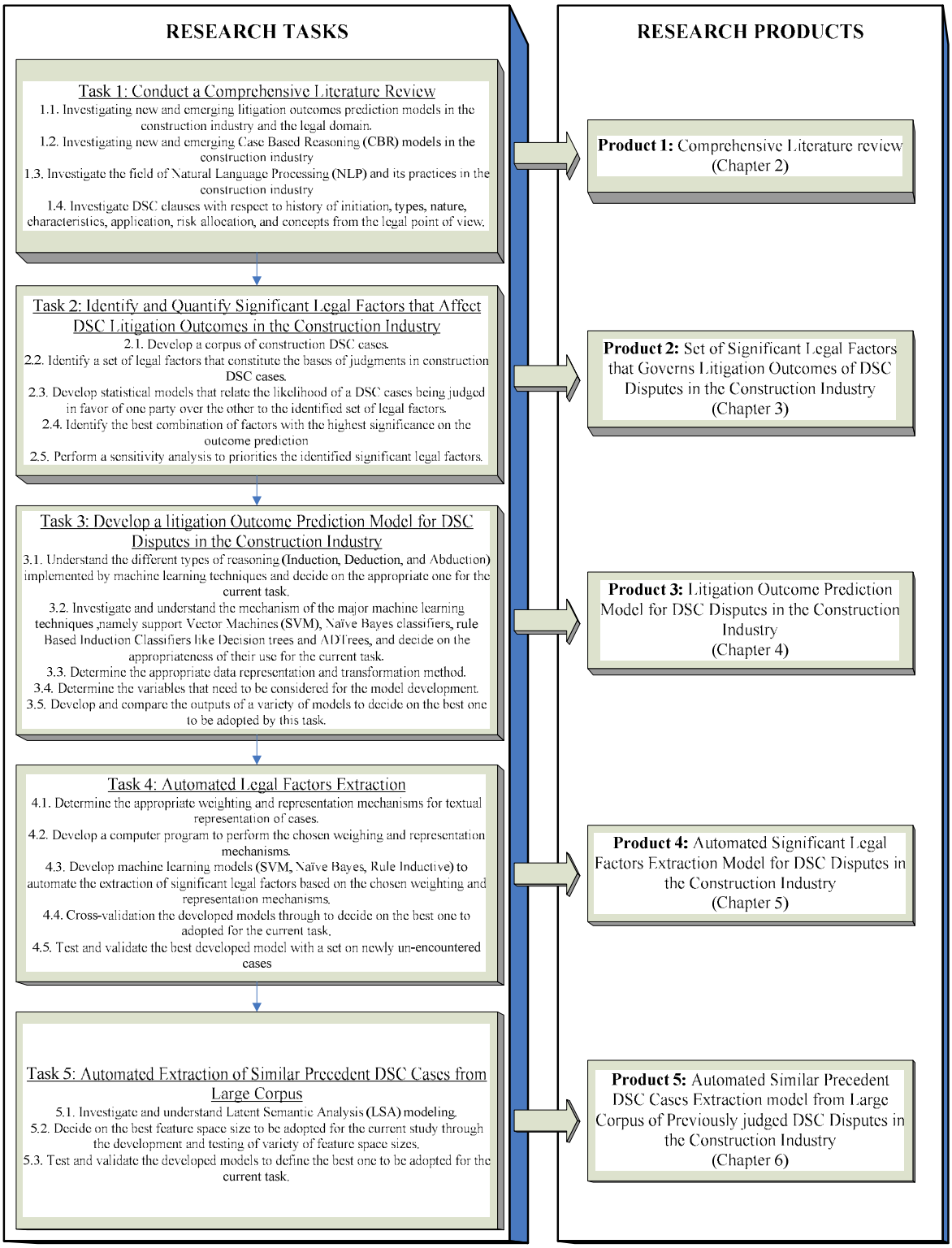


Figure 1.1 Research Tasks and Products

1.6 Thesis Organization

The organization of this thesis and its relation to the main research tasks of the current study is illustrated in Figure 1.1. Chapter 2 presents a detailed literature review that investigates (1) the latest research in litigation outcomes prediction in the construction and legal domains; (2) concerned CBR research in the different areas of the construction industry; (3) the field of NLP and its suitability for this research; (4) the different reasoning types implementing by ML algorithms; (5) the procedures of different ML algorithms like SVM, Naïve Bayes Classifiers, Rule Induction Classifiers (Decision Trees and ADTres), and Latent Semantic Analysis (LSA); and (6) the history, nature, characteristics, risk allocation, and application of DSC clauses in the construction industry.

Chapter 3 presents the identification and quantification of statistically significant legal factors that affect litigation outcomes of DSC disputes in the construction industry through (1) the development of a corpus of DSC cases; (2) manual extraction of legal factors upon which judges base their verdicts in DSC disputes in the construction industry; and (3) the development of statistical discrete binary choice models (namely Probit and Logistic models) to quantify the effect of the identified legal factors on the likelihood of entitlement. The chapter will compare the output of the developed models to (1) identify the effect of each legal factor on the prediction of the winning party; (2) identify the best combination of factors with the highest prediction precision; and (3) perform a sensitivity analysis to prioritize the most significant legal factors.

Chapter 4 presents the development of a litigation outcome prediction model for DSC disputes in the construction industry using ML. The chapter will aim to (1) identify the machine learning models' parameters; (2) prepare the data for model implementation; (3) develop SVM, Naïve Bayes Classifiers, and Rule Induction Classifiers litigation outcomes prediction models; and (4) validate and compare the developed models.

Chapter 5 presents the development of an automated significant legal factors extraction model using ML. The chapter will (1) identify the extraction model parameters such as number of folds, degree, and weighing mechanisms; (2) prepare the data for model implementation; (3) develop C++ algorithms for performing the data preparation processes and implement weighting schemes; (4) develop SVM, Naïve Bayes Classifiers, and Rule Induction Classifiers automated extraction models; and (5) validating and comparing the developed models.

Chapter 6 illustrates the development of an automated relevant precedent DSC cases retrieval model using ML and NLP techniques. The chapter will (1) investigate the main procedures of LSA algorithms; (2) identify the relevant model parameters such as the size of the reduced feature space and internal and external weighing mechanisms; (2) prepare the data for model implementation; (3) develop a set of different reduced feature spaces; (4) implement LSA automated extraction models; and (5) validate and compare the developed models.

Chapter 7 presents the conclusions, expected contributions, and recommended future research of the present research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The present research was motivated by the escalating damages and associated costs of claims and disputes on the construction industry. This escalation creates a need to devise ways, methodologies, and tools to equitably, economically, and rapidly resolve these disputes to minimize their damages on the construction industry. Consequently, the focus of this chapter is to create a solid point of departure for the current research through providing extensive background information about previous researches in the construction and legal domains focusing on the use of AI techniques for the developments of litigation outcome prediction models. This chapter will also illustrate the use of AI algorithms by researches in the construction domain to solve a variety of other problems. Furthermore, this chapter will provide background information about the history, nature, characteristics, risk allocation, and application of DSC clauses in the construction industry.

2.2 Litigation Outcomes Prediction Models:

As mentioned earlier in chapter one, instigated from the litigation drawbacks, a number of studies in the AI field attempted developing judicial reasoning methodologies and prediction tools to support the construction industry.

As a first attempt to provide a rule based computer system for legal analysis and claim assessment, Diekmann and Kruppenbacher 1984 developed an artificial

intelligence expert system for the analysis of differing site conditions claims called (DSCAS). The system prototype provides legal guidance to whether a claim, on the grounds of differing site conditions, has a likelihood of entitlement or not. The system was designed based on knowledge pertinent to the Federal Government Standard form General Conditions (2B-A, GP-4). Its logic was based on question/answer methodology that utilizes “if-then” logic. Each question is pertinent to a specific legal rule and each answer defines a different path to be followed within the logic. Legal rules of the system were carefully crafted after thorough investigation of the domain and lengthy consultation with claim specialists and construction attorneys yielding 22 modules. Each module included internal rules that would decide on the nature of the next module to be addressed within the logical process of deciding a certain claim. The DSCAS development was very promising to the use of AI techniques in this field. In further development for the DSCAS, Cobb and Diekmann 1986 developed knowledge based expert system titled Claim Expert Knowledge System (CEKS) in the same domain of DSC analysis to aid inexperienced legal advice seekers. The developed system was based on four concepts (1) the Federal Government Standard Form 23-A (Rev. 4-75) was chosen as the binding contract between the different involved parties; (2) the system was based on the owner’s prospective when deciding on the entitlement of a claim; (3) the system is intended for technically competent personnel supervising the contractor’s performance with a minimal legal knowledge; and (4) the right of entitlement of a claim is only based on expressed contract language and not any other implied rules or laws. Similar to the DSCAS, the logic of the CEKS was based on an expanded set of questions and

answers. The system was tested against 13 DSC cases which appeared before a Board of Contract Appeals (BCA) and predicted a similar decision to that of the BCA.

Inspired by the work of Diekmann and Kruppenbacher, in 1987, the US Army Construction Engineering Laboratory (USA-CERL) developed an expert system for the analysis of DSC titled Claim Guidance System (CGS-DSC) (Kim 1989). The methodology of the system was based on the DSC clause (FAR-52.236-2) used by the U.S. Government in its contracts. CGS-DSC utilized 13 modules to decide on the entitlement of a DSC claim. These modules were crafted after (1) careful consideration of the (FAR-52.236-2); (2) a detailed study of Diekmann and Kruppenbacher research; and (3) thorough analysis of the construction domain performed by six experts (2 experienced legal counsels and 4 experienced engineers in construction contract management). Since the system was intended for internal use of the USACE engineers, the decision about a claim was not very elaborate. The decision falls into one of the followings: (1) Very poor chance; (2) Poor chance; (3) Difficult to decide; (4) Fair chance; (5) Good chance; and (6) Excellent chance. To make-up the shallow decisions produced by the system, a set of 23 cases, gathered from LexisNexis and Westlaw, related to various DSCs were integrated into the system. The CGS-DSC retrieves a relevant case to the current situation from the case base after deciding about its entitlement as a supporting document for the reviewer. Later the scope of the expert system was expanded to cover different types of claims.

Hegab and Nassar (2005) implemented decision support systems in predicting the best solution for a contractor in commencement delay related claims.

The system utilized decision trees and probabilistic calculation methods in predicting the most cost effective alternative among litigation, relinquish of right, and amicable settlement. The system was implemented on one of the largest infrastructure projects in Cairo, Egypt (the New Sewer System). The new sewer system required new lines of 600 and 1000 mm diameter to connect it to the old existing system. The project was assigned to a joint venture of one Egyptian and four British companies under Design-Build contract. The decision tree analysis implemented for this project considered three alternatives (1) completing the project on time by increasing the resources without claiming extra time and money; (2) going to court claiming the delay costs and costs associated with accelerating the project; and (3) offering an a amicable settlement against a percentage value of the claim. Probability values were assigned by the contractor to each alternative and decision tree analyses were implemented. The analysis yielded alternative three to be the cheapest.

The success of expert systems in contract administration and legal prediction was very limited due to their failure in deducing all the necessary rules upon which the system operates (Bubbers and Christian 1992). They assume the existence of a full domain model that captures all required rules about a certain topic. As a result, they are much localized to a specific aspect of a certain domain. In addition, their accuracy and performance is crucially affected by the computational limitations. Consequently, other methodologies have been tackled to model judicial reasoning. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques (Aleksander

and Morton 1995). Inspired from this notion, Artificial Neural Networks (ANN) was utilized to model judicial reasoning by Arditi 1998 and Chau 2005 and 2006. Arditi (1998) attempted predicting the outcomes of construction litigation cases from Illinois Circuit Court using ANN system. The system utilized a software named Brainmaker and 102 cases that were defined by 43 input features (ranging between parties involved, contract type and conditions, project changes … etc) and 1 output feature defining the outcome of the court's decision (winner of the case either the Contractor or Owner). The ANN system attained a prediction precision of 67%. Chau (2005 and 2006) implemented a particle swarm optimization (PSO) model to train the perceptrons of an ANN system in an attempt to predict the outcome of construction litigation cases in Hong Kong. Similarly, the system utilized a set of 1105 of construction cases that were predefined by 13 input features and 1 output feature. Chau's model achieved a prediction precision of 80%. In 2006 Chau was able to attain higher prediction rate of 83%. The new system augmented the PSO earlier model with Levenberg-Marquardt (LM) algorithm to benefit from its global search capability.

Although ANN was able to achieve significant advancements to decision support capabilities in this domain, their excessive training stage and their ability to deal only with numerical data opened the horizon for the use of other AI techniques like Case Based Reasoning (CBR). CBR is a problem-solving paradigm that is fundamentally different from other major AI approaches like expert systems and Neural Networks. Aamodt and Plaza (1994) illustrate that instead of relying solely on the general knowledge of a problem domain, or making associations along

generalized relationships between problem descriptors and conclusions, CBR is able to utilize the specific knowledge of previously experienced concrete cases.

The literature in this domain illustrates the superiority of case-based reasoning systems over rule-based ones. For further illustration a comparison between the two systems will be presented concerning: (1) domain knowledge; (2) knowledge of reasoning; (3) development time; and (4) system maintenance and servicing. Firstly, rule-based reasoning systems are based on the presence of a full knowledge domain model that depicts all necessary rules to develop the system, which is nearly impossible to exist as stated by Bubbers and Christian (1992). Consequently, all developed systems are localized to a certain type of claims. On the other hand, case-based reasoning systems can accommodate for missing data since they are based on similarity measures, although that might affect their prediction precision. Secondly, rule-based reasoning systems are highly dependent on human judgment that must be coded in the form of logical rules that mimic the human judgment process. Whereas case-based reasoning systems depend on implicit human judgments that are available in the case base of the system. Thirdly, the development time of rule-based reasoning systems in comparison to case-based reasoning systems is enormous due the extensive investigation to derive and test the required rules by domain experts. Lastly, rule-based reasoning systems require continuous maintenance due to laws and codes modification that take place with time. However, the case base of a case-based reasoning system is automatically enriched with tested cases.

The superiority of CBR systems discussed above instigated its investigation in the construction domain. In 1999, a CBR model for the prediction of construction litigation outcomes was developed by Arditi and Tokdemir. The system implemented a CBR development tool named ESTEEM and the 102 cases from Illinois Circuit Court were augmented with an additional 12 recent cases for testing purposes. The prediction precision was enhanced to 83%. A higher prediction precision of 84% was attained by Chau 2006 by adopting a CBR reasoning approach to predict the outcomes of construction litigation cases in Hong Kong.

Furthermore, hybrid systems were investigated by few researchers in an attempt to improve the prediction precision. Arditi and Pulket (2005) utilized a boosted decision tree (BDT) system to predict the outcome of construction litigation. The study was conducted by using the same 114 Illinois court cases that were used in earlier prediction studies conducted with artificial neural networks in 1998 and case-based reasoning in 1999, augmented with an additional 18 cases that were filed in the period between 1990 and 2000. In this research, a boosting algorithm (ADABOOST) was utilized with decision tree algorithm through a software titled SEE 5. As stated by Arditi and Pulket (2005) “The conclusions indicated that ADABOOST can be used in many settings to improve the performance of a learning algorithm. When starting with relatively simple classifiers, the improvement can be especially dramatic, and can often lead to a composite classifier that outperforms more complex “one-shot” learning algorithms”. The main advantage of this system over ANN CBR is that the boosting algorithm works as a plug-in program and helps the primary learning machine to reduce the error rate by repeating decision tree learning

for a number of trials and by focusing on the attributes that have effects on error rates. The best prediction result obtained with boosted decision trees was 90%, which as illustrated by Arditi and Pulket (2005) is helping create a dispute-free construction industry. In addition, Chen and Hsu (2007) developed an ANN-CBR model for the prediction of the outcomes of construction litigation cases initiated due to change orders disputes. The model (HACM) integrated the learning feature of the ANN approach with similarity measures of the CBR model to achieve a prediction rate of 84.61%. The hybrid model constituted cases gathered from Supreme and Appellate courts over 48 states and districts in USA. They were characterized based on 23 input features, 6 of which are related to project data and 17 were change order related.

Research in the area of construction litigation outcomes prediction was initially motivated by the accomplishments in other domains. The legal domain, being very knowledge constrained, provided a very rich soil for developing tools for prediction of litigation outcomes. One of the first and most pioneering case based reasoning (CBR) tools HYPO was provided by Ashley and Rissland (1988a). HYPO was created to assist attorneys in building arguments about actual cases in the area of trade secret law. HYPO utilizes a set cases stored in its Case Knowledge Base (CKB) to derive an argument. It builds a claim-lattice of all the cases in the CKB that are relevant to a current case, by making “factual comparisons of cases relative to the problem situation and determine the legal significance in comparisons in terms of arguments about the problem situation” (Ashley and Rissland 1988b). The pioneering aspect of HYPO is that it provides: (1) factual arguments in favor of the

case in hand supported by similar cases in its CKB; (2) counter factual arguments to the case in hand supported by similar cases in its CKB; (3) suggestion of combination of facts for new hypothetical arguments that might provide new prospective for attorneys supported by similar cases in its CKB.

Believing that factual extraction alone in building a reliable CBR system is not sufficient; attempts have been made to develop methods of utilizing unformatted textual representation of cases to enhance the CBR systems potentials. SPIRE combines CBR and Information Retrieval (IR) techniques to locate text passages related to a certain legal situation within long textual representation of cases (Daniels and Rissland 1997). Weber (1998) developed Pruentia to support jurisprudential research by providing a case based retrieval engine over a database of automatically indexed textual legal cases. More recently, Bruninghause and Ashley (2001) experimented with Natural Language Processing (NLP) techniques to enhance the reasoning capability of a CBR system by understanding meaningful features and relations expressed by words. The developed Textual Case Based Reasoning (TCBR) system implemented AutoSlog with Smart Indexing Learner (SMILE). AutoSlog is a NLP/Information Extraction (IE) system that was developed by Ellen Riloff at the University of Utah (Riloff 1996). It utilizes a powerful heuristic sentence segmenter, Sundance, and module for generating extraction rules from unformatted textual representation. SMILE “integrates IE and Machine Learning (ML) methods for automatically assigning abstracted indexing concepts to text cases” (Bruninghause and Ashley 2001). Weber et al. (2001) employed domain ontology for TCBR system development.

For further development, researchers within the legal domain have attempted hybrid or mixed approach to predict outcomes of litigation cases. HELIC-II models legal reasoning using two engines, a case-based engine identifies similar cases and extracts legal concepts from them, and a rule-based engine uses the legal concepts and the current case's facts to infer all possible legal consequences (Ohtake et al. 1993). CABARET (Rissland et al. 1989), GREBE (Branting 1999), and Anapron (Golding and Rosenbloom 1996) are hybrid systems that combined rule based reasoning with case based reasoning techniques for prediction purposes. CARMA (Branting et al. 2001) and IBP (Brunghause and Ashley 2003) are algorithms that combine case based reasoning and model based reasoning for the prediction of litigation outcomes. In 1995, Egri and Underwood utilized ANN to provide the Hybrid Integrated Legal Decision Assistant (HILDA) tool to extract legal knowledge and predict litigation outcomes concerned with the question of "unjust" contracts based on the Contract Review Act 1980 (New South Wales). HILDA integrated similarity measures of RBR and CBR methods as well as the patchy domain theory presented in the legal domain. Legal rules are implemented through ANN to categorize cases in question either for plaintiff, against plaintiff, or undecided. Cases that fall within the undecided region are then tested with the CBR component to fit it to one of the other two categories. Brunghause and Ashley (2005) combined the SMILE system with IBP system developed in 2001 and 2003 respectively in a hybrid system to achieve higher prediction rates. The attained results were promising but indicated that further research is needed in the field of NLP. In a recent research, El Hadi (2007) developed a statute base Information Retrieval Case Based Reasoning (IR-CBR)

hybrid system that implements natural language description of actual situations as its input to retrieve related cases to enhance prediction of litigation outcomes in Bankruptcy Case Law.

Research on the prediction of litigation outcomes was not only performed by researchers in universities. Its significance has captured the interest of Government institutes like Donald Berman Laboratory for Information Technology and Law in Australia over the years. In 1991, Donald Berman Laboratory for Information Technology and Law provided a hybrid object oriented rule based system named Intelligence Knowledge BAsed Legal System (IKBALS) to decide upon worker compensation in work care cases in Australia. The second version of the system IKBALSII augmented a case based reasoner and intelligent information retrieval components to the rule based reasoner (Zeleznikow 2003). In 1995, Donald Berman Laboratory for Information Technology and Law built the Split-Up expert legal system that provided advice on the distribution of property under the Australian Family Law. The Split-Up system is a rule based/ Artificial Neural Network (ANN) system derived from factors attained from thorough investigation of the governing legal factors with domain experts (Zeleznikow 2003).

From the literature in this domain, it was noticed that there is still work to follow in this area. It is apparent that AI research in the legal and construction domain has been progressing along similar lines. An important aspect in both of these domains is that they rely heavily on textual material expressed in human language: legal references and judicial opinions in the legal domain, contract conditions, specifications, correspondences, etc. in the construction domain. This

creates a strong need for well defined methodologies that are capable of effectively analyzing textual material and efficiently retrieving pertinent information from them. Besides, in all the above mentioned research studies in the construction domain, information extraction and case attributing were manually performed and fed to these systems. It is a fact that, the accuracy of the output of a machine learning system is largely dependent on the availability of reliable information about the attributes used to define the training cases. As Arditi and Pulket (2005) state "Finding a complete and reliable set of training examples is difficult in construction litigation cases". The use of natural language processing techniques NLP can enhance and facilitate the use of construction litigation prediction models. Automatic cases classification and knowledge extraction can be improved through NLP techniques (Bruninghause and Ashley 2001). It can further provide the ease of access to legal knowledge for legally inexperienced personnel in the field. The highly sophisticated electronic information storage and retrieval systems available for researching the law are extremely complex and time consuming. Sometimes this complexity creates problems for information seekers and can limit their access to relevant information. Consequently, accurate legal decisions within the construction realm are exceedingly time consuming and may require knowledgeable professionals to obtain the required decision support. As a result, an automated legal support system that utilizes natural language processing techniques to identify, retrieve, reorganize legal information, and predict construction litigation outcomes will reduce the time required and costs spent by construction firms and improve overall project control.

2.3 Case-Based Reasoning Models:

The success of legal prediction models, which depended crucially on the adequacy of learning from experience, has contributed to the birth of similar line of research in different fields. One of the problems being tackled through the use of AI in different domains is how to represent and reuse knowledge and previous experience. Earlier attempts constituted developing knowledge-based systems (KBS), which are considered one of the success stories of AI research. “In a recent survey the UK Department of Trade & Industry found over 2000 KBS in commercial operation (the survey excluded KBS in University research laboratories)” (DTI 92). KBS utilize domain model based systems like rule based and object models (Clancey 85). Despite its success, several problems were reported by developers and users of KBS (Watson and Marir 2007). Some of these problems are

- Knowledge elicitation is a difficult process, often being referred to as the *knowledge elicitation bottleneck*;
- Implementing KBS is a difficult process requiring special skills and often taking many man years;
- Once implemented model-based KBS are often slow and are unable to access or manage large volumes of information; and
- Once implemented they are difficult to maintain (Bachant & McDermot 1984, Coenen and Bench-Capon 1992, Watson et al. 1992).

Consequently, more efficient tools and techniques have been thought of as a solution to these problems. Case-Based Reasoning (CBR) is a paradigm solving mechanism that mimics previous knowledge about a solution of a similar problem to

solve newly introduced ones (Kolodner 1993). In CBR systems, expertises are embodied in a library of past cases, rather than being encoded in classical rules. Each case typically contains a description of the problem, plus a solution and/or the outcome. The knowledge and reasoning process used by an expert to solve the problem is not recorded, but is implicit in the solution. In fact, the work of Schank and Abelson (1977) in the field of philosophy is considered to be the main focal point and origin of CBR systems. They claimed that human general knowledge is build up in the form of scripts based on our experiences and used to derive judgments and expectations of newly encountered situations (Schank 1982). Based on these philosophical roots of CBR, the first CBR applications were introduced by Roger Schank's group at Yale University in the early eighties (Watson and Marir 2007). As discussed in section (2.2), the legal system in the United States of America, being an Anglo Saxony system and intensively concerned with previous experience derived from precedent cases, is a very rich domain for applications of CBR systems.

2.4 CBR in Civil, Architectural, and Construction Engineering:

The success of CBR systems in different domains contributed to the birth of its use in the engineering field. Construction Engineering is a very dynamic field. Decisions in this field are influenced by factors that vary from one project to the other like project size and complexity. These factors may influence decisions concerning the involvement of diversified parties with different specializations, site conditions, contract type and conditions, and project location …etc. (Caldas et al. 2002).

Decisions of this nature are highly unstructured and no clear rules are available to provide a clear basis for making tem. Consequently, decision makers employ previously acquired knowledge through experience and similar cases. This property made construction a very prominent field for the use of CBR (Chua 2001). The rest of this section is dedicated to provide a literature review for the use of CBR in the fields of Structural, Architectural, and Construction Management engineering.

2.4.1 Architectural/Structural CBR Models:

CBR had been used within the design field to facilitate re-use of architecture and structure designs. As defined by Schmitt (1988), architecture design is the art of producing a complete building specification from an incomplete problem description. Consequently, architects employ acquired positive and negative experiences when solving design problem rather than generating the building design from scratch every time. This is supported by the notion that there are no definite formulas, methodologies, or algorithms that can map a design problem to a formalized architecture design solution since there is no formalized definition of architecture quality. “Consequently, traditional architectural design education makes extensive use of architectural cases” (Schmitt 1993). This aspect of design problems initiated research that aim to utilize CBR as an aid to the design problem. CBR has been utilized to solve new design problems by adopting, modifying, or combining existing cases. Pearce et al. (1992) developed ARCHIE a CBR architectural design system. CADSYN and DDIS are structural design system proposed by Maher and Zhang (1991) and Wang (1991) respectively. Schmitt (1993) provides one of the most

successful Architectural Case-Based Design (CBD) system named Architecture Case Based Design System (ACABAS) that was applied to contemporary designs of the Ticino architect Campi and Pessina in Switzerland. CBDs are a specific type of CBRs that have a wide spectrum of capabilities ranging from generating a description of existing buildings or designs in the case base to the creation of a complete building specification for a new design problem (Schmitt 1988). The ACABAS system utilizes an object oriented database that supports binary large objects (BLOBs) which stores structured and unstructured information about the different cases. This database includes CAD models that are precisely generated for the CBD system. A developed pre-processor (Mod-4) was designed for this function. It accepts geometric description of the building and requires further information like room labeling, materials description, and building design specifications to generate object database and graphical representation of each case. The later constitutes a set of unstructured information like scanned images, text description of the building and its location, interview with the occupants, energy bills, acoustical and thermal problem areas, textual description of repairs history … etc. Normative and Functional constraints are further identified as parameters of each case. When a new design situation is introduced to ACABAS, it retrieves the most similar case and implements adaptation mechanisms to fully satisfy the parameters of the new problem. Topological and dimensional discrepancies are identified as the first step of adaptation. In case of discrepancies, adjustments are applied based on transformation rules that are built into the system while maintaining the normative

and functionality constrains un-violated. ACABAS undergoes an iterative process until all transformations are applied without violation of the defined constrains.

In addition, Watson and Abdullah (1991) employed CBR in building defect diagnoses through the development of PAKAR. Flemming and Woodbury (1995) built up the SEED project, which utilizes case-based reasoning to provide computational support for the early design phase. Roddis and Bocox (1997) developed a hybrid system for resolving fabrication errors in steel highway bridges that is in operation in Kansas Department of Transportation (KDOT). The Bridge Fabrication error solution eXpert system (BFX) integrates a case and rule based modules. The former, case-based BFX (CB-BFX) was created using the programming language CommonLISP and the CBR tool MEM-1. The system which was developed to provide a formalized methodology for repair of fabrication error had a case base of 112 cases of previously experienced errors and corrective actions gathered entirely from KDOT projects. Cases were classified into 13 sub modules based on the type of fabrication error as follows: mis-located holes (33 cases), mis-cut members (20 cases), nicks and gouges (13 cases), mis-located members (10 cases), mis-shaped holes (8 cases), edge distance (6 cases), laminations (6 cases), mis-aligned members (6 cases), mis-attached members (4 cases), size error (2 cases), stress fracture (2 cases), end distance (1 case), and partially drilled holes (1 case). Evaluating the use of CB-BFX module alone yielded a precision of 82%, which was an impressive advancement over the use of the rule-based module that attained 63%. The combined hybrid system, using both modules, achieved an overall success rate of 91%.

Caldas et al. (2002) stated that the complexity of modern construction projects leads to the use of increasingly sophisticated construction methods and requires extensive interactions between diversified parties. This increasing complexity could be the reason that system analysis and design has been gaining increasing importance in the development of complex technical systems (Praehofor and Kerschbaummayr 1999). As an example of that, facilitated Computer Aided Systems Architecting CASA, a technique combining systems and requirement engineering approaches with AI, is growing rapidly to cope with the market competition (Caldas et al. 2002). Praehofor and Kerschbaummayr (1999) developed a case-based approach to be augmented with CASA to support reusability of designs of existing systems in determining the architectural requirement fulfillment of new components under design. Retrieved solutions by CASA are accompanied by a degree of fulfillment factor (DOF) between [-1, 1] signifying the extent of similarity and required adaptation to new paradigms. To further explain the DOF concept, a DOF value of:

- 1 means full fulfillment of the new system requirements and can be adopted as a solution with no modifications.
- 0 to <1 means partial fulfillment of the new system requirements and can be adopted as a solution with some architecture tailoring.
- -1 to <0 means does not fulfill the new system requirements and cannot be adopted as a solution.

CASA employs predefined language and lexical structures, which are domain dependent, with object oriented structure to define new components' properties and

requirements and had showed significant success in transportation and material handling design.

Likewise, Sirca and Adeli (2005) developed an intelligent hybrid decision support system (IDSS) that utilizes CBR and ANN to assist bridge engineers to semi-automatically convert the rating of bridges from Working Stress Design (WSD) method to Load Resistance Factor Design (LRFD) method. According to Sirca and Adeli (2005), in 1995, the Federal Highway Administration (FHWA) required that all bridges, regardless of the design method used for the original design, be based on the load factor design (LRFD) method. However, steel bridges originally rated using WSD had crucial data missing to make the proper conversion to the LRFD method. As illustrated by Sirca and Adeli (2005), a steel girder bridge rated by either method requires input into the BARS-PC program, software used by Ohio Department of Transportation (ODOT) for bridges design, which describes the girder's section properties. For the WSD-based bridge rating, a general description of the section properties including only the cross sectional area, moment of inertia, and section modulus of each girder cross-section would suffice. For the LRFD-based rating, however, a detailed description of the section properties is required including information about individual elements making up the steel girder cross section such as the total height of the section, and the areas of the web and flange elements and their individual moments of inertia and the distances from their centroids to a reference axis. In addition, another major piece of information that is required for the conversion, and not included in the WSD design method, is information regarding the spacing of lateral bracing of the girders. Such an aspect made the rating

conversion very hard and labor intensive, for an engineer has to use his knowledge to make decisions about the lateral bracing spacing from the design data available from the WSD design method and the design guidelines utilized at the period of designing the bridge (Waheed and Adeli 2005). As a consequence, the expert system was developed to assist in deriving the missing data about lateral bracing requirements from similar cases for bridges under the jurisdiction of ODOT. The system employed structure analysis files attained from AASHTO Bridge Analysis and Rating System (BARS-PC) as its case based knowledge database. CBR is utilized to define a similar case and attain input data that are employed in the ANN, a system that was developed in an earlier research by Sirca and Adeli (2004), to define the required missing parameters of section properties description. After attaining all required parameters, the BARS-PC data file is updated and saved. The CBR shell, Induce-It, is used to create and manage the CBR module for the Intelligent Decision Support System (IDSS). The IDSS case base included 39 cases that were characterized by textual or numerical nature, field names that represent the case data such as the year in which the bridge was designed, the span length(s) of the bridge, and the number of cross-frame spaces. As stated by Sirca and Adeli (2005), the year at which the bridge was built is the most crucial information in determining the appropriate lateral bracing due to the number of changes that were made to the design process through the years. Based on that, weights are assigned to each field based on its relevance. These data characteristics as well as assigned weights to each field are utilized to define the similarity between a new case and those available in the case base. Similarity measures of cases are based on linear

weighted similarity functions which are then ranked using Nearest-neighbor matching to define the most similar case. When a matching case is defined, its lateral bracing data are retrieved from separate database and inputted to the ANN to decide on the conversion required.

2.4.2 Construction Engineering CBR Models:

In addition to Architectural and Structural design, CBR approaches were implemented in variety of construction engineering management problems including construction duration estimation, productivity estimation, cost of building estimation, bid decision making, procurement criteria selection, construction negotiation methodologies, and contract strategy formulation.

Project scheduling is one of the key factors in determining the success of construction projects. Interest in developing and formalizing good scheduling practices has always been of significance in the construction research community (Miyashita and Sycara 1992). The process of scheduling assigns a set of tasks to a set of resources with finite capacity over time (Hinze 1998). Successful scheduling requires judgment about variety of interrelated factors and criteria concerning diversified and characteristically conflicting set of constrains (French 1982). Over the last decade, there has been an increasing interest in techniques that exploits previous experience in developing and modifying project schedules (Hinze 1998). Sycara and Miyashita (1994) provided a CBR approach in CABINS for iterative schedule revision in job shop schedules. CABINS is composed of three modules (1) an initial schedule builder based on constraint-based scheduler; (2) an interactive

schedule repair module, and (3) an automated schedule repair module. Schedules developed in the first module are not optimized due to the absence of the complete knowledge of the scheduling domain model and user preferences (Miyashita and Sycara 1994). To attain an optimized schedule, CABINS implements the second and third modules through a CBR approach that adopts previous optimizations in the case base. CABINS gathers the following information in the form of cases through interaction with a domain expert in its training phase.

- A suggestion of which repair heuristic to apply: a user's decision on what repair heuristic to be applied to a given schedule for quality improvement.
- An evaluation of a repair result: a user's overall evaluation of a modification result. The evaluation categories currently employed are 'acceptable' and 'unacceptable'.
- An explanation of an evaluation: when a user evaluates the modification result as unacceptable, she/he indicates the set of undesirable effects that have been produced. The explanation given to CABINS consists of the numerical rating of each identified effect. (Sycara and Miyashita 1994).

In the optimization process, CABINS identifies vulnerable activities based on the user's preference criteria. The system then works in an iterative manner and optimizes schedule activity by activity and not the whole list at once. The most similar modification requirement retrieved from the case base using K-Nearest Neighbor is adopted for the first activity. The outcomes and effects on the schedule corresponding to the user's preference criteria are identified and presented to the user. If the optimization is accepted, the case base is enriched with this particular

optimization. On the other hand, if the optimization is not accepted, the user is asked to provide a justification that is tagged with the optimization process in the case base and other iterations are performed.

Amicable settlement through negotiation is another construction problem that entails extensive expertise and knowledge of similar cases. Li (1996) provided a CBR intelligent support system to construction negotiation. “This model has been implemented in the MEDIATOR, a computer program that utilizes previous cases as a basis for addressing new problems. In contrast to conventional expert systems (ESs) that use compiled knowledge in problem solving, the system selects similar cases to help in solving a given negotiation problem” (Li 1996). Cases in the case-base are represented in terms of 6 factors: (1) case number and indexing keywords, (2) situational description addressing the background of the negotiation, (3) negotiating parties, (4) disputant issues and goals, (5) final settlements, if it is successful, or unsuccessful, and (6) negotiation history. MEDIATOR allows each of the parties to illustrate their “issues and goals” which are used as factors for retrieval of similar cases. The solution of the most similar case is adopted as a solution to the new situation, which could be accepted, rejected, or employed to derive new users’ goals.

Yau and Yang (1998) developed CBR-CURE a case-based reasoning system for estimating the construction duration and cost of building construction project at the preliminary stage to decide which design is feasible and most beneficial to the owner. CBR-CURE was developed using ESTEEM, a Window based tool for developing CBR systems, which is commercially available through Esteem Software

Incorporated since 1991. The case database constituted of 60 hypothetical projects generated using a construction planning expert system. The Time/Cost Integrated System (TCIS) integrates rules from experienced construction experts and mean cost data. The cases are identified by 13 input features, among which are project's name, start and finish dates of the project, and 4 output features defining the duration, equipment cost, material cost, and labor cost of the project. The system input interface allows the user to assign weights for each of the 13 input features. These weights are utilized to determine case similarity. The interface also allows the user to define a minimum similarity value above which cases are deemed similar and are retrieved. The duration and cost of a new case is determined by using adjustment factors that are built into the system to modify the values attained from a retrieved case.

Furthermore, the dynamic nature of the construction bidding decision making process also led to the development of an automated CBR system CASEBID that proposes a markup level, based on the criterion of maximized expected profit, for a newly introduced bidding situation from previous bidding cases and domain knowledge (Chua et al. 2001). The system focuses on risk and competition factors that affect the bidding decision by integrating domain knowledge, derived from a thorough investigation with domain experts of internal and external factors affecting the nature of a decision, with case based knowledge. In a comparative study CASEBID outperformed the conventional statistical approach. It posed 55% bid wins, yielding an average 7.4% expected profit compared to 41% bid wins, yielding an average 6.15% expected profit in the case of the latter approach.

As a matter of fact, construction projects include many repetitive and cyclic activities (Kaneta et al. 1999). Likewise, judgment about the best methods and techniques to be adopted for cyclic processes is based on previously attained expertise concerning productivities and technologies. Graham and Smith (2004) proposed a CBR based estimator (CBE) to predict the productivities of concreting cyclic operations from previous cases. The model consisted of 5 input features and one output feature. “CBE was validated, not only against the performance of past operations (which were not used in the model development), but also against estimates provided by a professional construction planner. The model was found to provide more precise and consistent estimates than the planner, with 90% of the estimates being within a 10% relative error of the observed value” (Graham and Smith 2004).

In such a dynamic environment as that of the construction industry, procurement decisions are crucial to the success of project. In such decisions previous knowledge is the corner stone of decision making (Love et al. 1998). Timely deliveries are major aspects of the successful completion of any construction project (Luu et al 2006). Consequently, Companies tend to work with suppliers with whom they had good experience. Researchers have pointed out that the identification and use of a suitable procurement system could contribute immensely to the success of a construction project (Naoum 1994; Rwelamila and Meyer 1999), and this has been a driving force for the development of various procurement selection approaches. Such dependency on previous experience gives a high potential for CBR approaches for modeling the procurement selection decision within a complex

dynamic environment. Luu et al. (2005) examines the suitability of CBR approaches for procurement selection by creating a prototype model of procurement selection criteria (CaPSC) to assist decision makers in selecting appropriate procurement systems. The model applies CBR approach to procurement criteria selection irrespective of the variability in the characteristics of the client, project, and external environment. These factors are very hard to model based on their wide diversity (Luu et al 2005). As a consequence, the prototype model relates these parameters to their associated factors that can affect such a decision like speed, time certainty, quality, flexibility, risk allocation …etc. For more illustration, if "on-time completion" is a key objective of the client, not only the speed but also the time certainty, flexibility, and quality are considered during the evaluation process. The evaluation factors were derived from a methodical investigation of the different procurement selection criteria techniques and semi structured interviews were conducted with managers of five major client organizations in Australia (four governmental and one private) experienced in construction procurement selection.

One of the main construction problems that is normally resolved using previously gained knowledge and managerial expertise is contract strategy formulation. In fact, it is inherently, too complex, too personal, and too dynamic to be modeled in a fully automated manner (Reuber 1997). Despite this difficulty, CBR approaches can be utilized to facilitate automation of the use and reuse of these expertises. Chau and Loh (2006) developed a prototype of a decision support system, CB_Contract, which exploits CBR approach for contract strategy formulation. The system incorporates the four main components of contract strategy

formulation, namely work packaging, functional grouping, Contract type, and award method. It further integrates these components with other crucial factors, such as "form of contract, currency and timing of payment, nomination of subcontractors by the client, type of specifications (performance or construction method), penalty scale for liquidated damages, and occasionally the provision of contractual motivation and incentives" (Chau and Loh 2006). ReCall, an interactive human machine system, was used for the development of CB-Contract. "The case retrieval process takes place within the ReCall environment using inputs from the user. Thereafter, the user will carry out the necessary adaptation to the cases to formulate the contract strategy for the current project based on three important considerations: (1) robustness of the retrieved set of sub strategies; (2) compatibility of the sub strategies; and (3) effectiveness of the alternative solutions." (Chau and Loh 2006). To assist the user in making such decisions, each case is associated with a brief description of the project. The adopted method and the case parameters are then augmented into the knowledge base of the system for reuse in future models.

All of the above studies illustrate the growing application of CBR approaches in the engineering disciplines. Motivated by this growth, Dogan et al. (2006) performed a detailed study to compare the performance of three optimization techniques, namely feature counting, gradient descent, and genetic algorithms (GA) in generating attribute weights that were used in a spreadsheet-based case based reasoning (CBR) prediction model. The model was utilized for early cost prediction of structural systems and was tested by using data pertaining to the early design parameters and unit cost of the structural system of 29 residential building projects.

The results indicated that GA-augmented CBR performed better than CBR used in association with the other two optimization techniques.

It is evidently clear from the reported research studies that CBR approaches are very helpful in utilizing previously learned experiences to solve newly encountered ones. In contrast, the success of model based and rule based approaches is hampered by the fact that they are dependent solely on the computational efficiency, and the assumption that there exists a strong domain model. These characteristics have limited its use in real world tasks since the existence of a strong domain model can almost never be assumed. However, as mentioned earlier in section (2.1), the success of CBR approaches comes at a higher cost of manually extracting information pertaining to the different cases. A possible solution to this problem can be obtained through Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques as will be shown in the following section.

2.5 Natural language processing (NLP):

Natural Language Processing (NLP) is wide and very active area of research. NLP covers a wide spectrum of techniques ranging from rule based techniques to statistical probabilistic tools. Consequently, there is not a single agreed-upon definition of what NLP exactly is. However, there are some agreed upon aspects of what NLP is. NLP is a theoretically motivated range of computational techniques for the analysis and representation of naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a

range of tasks or applications (Manning and Schèutze 1999). Naturally occurring texts can be of any language, mode, genre, etc. The language can be expressed orally or in writing. The only requirement is that they be in a language used by humans to communicate with each other. Also, the language being analyzed should not be specifically constructed for the purpose of the analysis, but rather that the text is gathered from actual usage (Allen 1995).

In fact, the field of NLP was originally referred to as Natural Language Understanding (NLU) in the early days of AI. However, it is agreed today the true NLU is not yet accomplished (Jurafsky and Martin 2000). A full NLU System should be able to accomplish tasks like paraphrase an input text, translate the text into another language, answer questions about the contents of the text, and draw inferences from the text. While NLP has made outstanding achievements in some of these venues, NLU still remains the goal of NLP due to the fact that NLP systems cannot by themselves draw inferences from text (Liddy 2003). As stated by Manning and Schèutze in their book Foundations of Statistical Natural Language Processing, NLP practices are governed by nature of the domain of its application. Among the key contributors to the discipline of NLP are: “Linguistics - focuses on formal, structural models of language and the discovery of language universals - in fact the field of NLP was originally referred to as Computational Linguistics; Computer Science - is concerned with developing internal representations of data and efficient processing of these structures, and; Cognitive Psychology - looks at language usage as a window into human cognitive processes, and has the goal of modeling the use of language in a psychologically plausible way” (Liddy 2003).

Research in natural language processing has been going on for several decades dating back to the late 1940's (Jurafsky and Martin 2000). Machine translation (MT) was the first computer-based application related to natural language. Weaver and Booth started one of the earliest MT projects in 1946 on computer translation based on expertise in breaking enemy codes during World War II (Hutchins 1997). Throughout late 1960's and early 1970's NLP related researches focused on improving of theories concerning how to represent meaning and developing computational solutions that the existing theories of grammar, at that time, were not able to produce (Jurafsky and Martin 2000). Alongside theoretical development, many prototype systems were developed to demonstrate the effectiveness of particular principles. Weizenbaum's ELIZA was built to replicate the conversation between a psychologist and a patient by simply changing the order of the user input (Jurafsky and Martin 2000). ELIZA plays the role of a therapist, asking questions based on the answers of the user, who plays the role of the patient. The program contains a database of keywords and a specification of output for each keyword. The program searches for a keyword in the user's answer and asks the following question based on the output specified for the keyword. ELIZA therefore does not actually understand the dialogue with the user, nor does it make any arguments, conclusions, or claims. This is acceptable in this particular dialogue between a therapist and a patient in which the therapist can pretend to not know anything about the real world (Jurafsky and Martin 2000).

Perhaps the most recognized uses for NLP techniques today are those related to commercial applications such as the spelling and grammar correcting

capabilities of modern word processors (Church and Rau 1995). However, text-based NLP techniques have been utilized in numerous applications such as information extraction and retrieval, automatic text summarization and machine translation (Allen 1995). Such NLP-enabled applications have been used in various areas including the financial field, computer software development and law.

Motivated by the success of NLP techniques and the advancements in computational resources, the awareness within each community of the potential solutions to textual dependent problems has grown. In pursuit to enhance information models, document integration, and inter-organizational systems in construction engineering and management, AI and Natural NLP techniques have been employed extensively through a variety of automated and semi-automated tools (Labidi 1997). Text mining methodologies, document clustering techniques, controlled vocabularies schemes, and web based models were some of the techniques utilized to perform the above mentioned tasks (Caldas and Soibelman 2003). Most of the present construction information integration tools are designed to work with structured data like CAD models and construction scheduling databases. However, most of the available data are stored in semi-structured or unstructured format like contract documents, change orders, RFIs, and meeting minutes that are normally stored as text files (Caldas et al. 2002). Consequently, facilitating the use of these documents through integrated methods has become a necessity to enhance project control, performance, and data reuse. A number of previous research studies attempted to achieve this objective. A computerized database for the classification, documentation, storage, and retrieval of documents about rising construction

technologies was presented by Ioannou and Liu (1993). Controlled vocabularies were proposed by Yang et al. (1998). The researchers used manual and text mining techniques to scrutinize a number of methodologies thesauri to promote design information reuse. Kosovac et al. (2000) investigated the use of controlled vocabularies for the representation of unstructured data.

In further attempts, Hajjar and Abou Rizk (2000) provided a document collaboration methodology. Their approach employed common data model customized to a unique segment of the construction domain to define projects and document data. Wood (2000) provided a method for hierarchical structure of concepts extraction from textual design documents. Scherer and Reul (2002), on the other hand, utilized text mining techniques to classify structured project documents.

Over the last few years, there has been a significant growth in the use of databases in different sectors like business, government, and scientific at a rate that developments in traditional data analysis methods cannot cope with. The nature of the data, being expressed in natural language and stored in unstructured format, represents the main hurdle hampering the efficient use of traditional data analysis methodologies. “The traditional methods can create informative reports from data, but cannot analyze the contents of those reports” (Soibelman and Kim 2002). A significant need exists for a new generation of techniques and tools with the ability to automatically assist humans in analyzing the very large amount of data for extracting useful knowledge. Inspired by this pressing need, in 2002, Soibelman and Kim utilized knowledge discovery in databases (KDD) and data mining (DM) techniques to develop a new tool to automatically analyze and derive knowledge from

construction databases. The tool was implemented within a frame work of the Resident Management System (RMS), a system developed by the US Army Corps of Engineers for project management and control, to extract knowledge about causes of delay in Flood Control Projects at Fort Wayne. The system integrated data mining techniques through decision trees, and ANN in two modules. In data mining, feature subset selection was first used to calculate the relevance of features that were implemented in decision tree algorithm to extract rules from the data sets. Rules from decision tree made the input selection for the neural network a simple task and the understanding of outputs of neural network easier. Finally, neural networks were used to make predictions of the future trends in a construction project. The C 4.5 decision tree algorithm was used to predict the effective causes of delays that were used as input data for the ANN. The 224 projects at Fort Wayne were classified into a downward expanding decision tree, in which each node represents a cause of delay. In addition, each node is also associated with a percentage value defining the relevancy of the cause. For example, among the 224 projects, 120 projects (54%) were delayed. The 120 projects were first tested for Inaccurate Site Survey as a cause of delay yielding 36 cases (16%) with related delays and 84 cases with other causes of delay. The first node is further branched by testing the cases against Shortage of Equipment cause of delay. The C4.5 algorithm defined nine effective cases of delay (inaccurate site survey, number of workers, incomplete drawing, change order, shortage of equipment, duration, season, weekends, rain/snow) that were implemented in the ANN. As mentioned by Soibelman and Kim (2002), a great number of NN were run to find that the best

results were achieved with 1% learning rate and 3 layers back propagation NN architecture. The results of the implementation were promising and identified that the main cause for delays of Flood control project at Fort Wayne was inaccurate site survey rather than the weather related problems initially assumed by site managers.

Furthermore, Caldas et al. (2002) and Caldas and Soibelman (2003) presented the use of information retrieval techniques to enhance information organization and the use of inter-organizational systems through automated classification of construction projects. The research proposed a methodology for the use of information retrieval via text mining techniques to facilitate information management and permit knowledge discovery through automated categorization of various construction documents according to their associated project component using standard classification configuration of the Construction Information classification Systems (CICSs).

Due to the persisting need to facilitate access, use, and reuse of unstructured construction project documents, Xie at al. (2003) also provided an integrated model for the retrieval of construction project documents to facilitate decision making, logical judgment, and control by project managers. The proposed system utilized a user provided model of construction project management and a user configurable visitor to retrieve information based on users' needs. Moreover, a study for scrutinizing a methodology for incorporating construction project documents in architectural engineering, construction, and facility management (AEC/FM) model based information systems was investigated by Caldas et al. (2005). The study focused on methods of augmenting and facilitating entry of large documents in

project management information systems to improve overall project control through semi-automated support integration.

Demian and Fruchter (2005) investigated the use of different text analysis methodologies to highlight and quantify potential significance and similarity among objects from an archive of building models to facilitate and improve design reuse. They made use of a corporate model (CoMem) prototype which provides an overview of a corporate memory in the form of a map to aid the process of finding reusable design items. Their proposed methodology examined the use of vector model text analysis augmented with latent semantic indexing, context sensitive comparison, and tree matching retrieval techniques.

Ng et al. (2006) implemented Knowledge Discovery in Databases (KDD) and Data Mining (DM) to define common characteristics of maintenance records as they relate to different types of university facilities (Housing and academic), location of different university facilities, and the nature of the required maintenance reported in the Facility Condition Assessment database. The FCA database contains deficiency information in the form of textual reports on facilities located at three campuses within a statewide university system. The developed KDD system implemented a combination of statistical analysis techniques and cluster analysis for text mining to discover common patterns in the deficiency description reports available at the university's FCA. Statistical analysis was utilized to derive a consistent representation of each deficiency report in the FCA in terms of the frequency of words repetition within the data base. To attain similarity measures between the different reports, Support vector Machine (SVM) methodology was implemented.

SVM stores a list of terms and their frequencies for each document. Every document (deficiency report) becomes a vector in ***S*** dimensional space, where ***S*** is the number of terms in the group of documents. VSM is based on the assumption that similar vectors in the ***S*** dimensional space will represent similar documents. After attaining a consistent representation of all reports, automated clustering of the deficiency reports is performed based on the deficiency type. K-nearest neighbor clustering algorithm was utilized for that purpose. Such methodology was very efficient in deriving knowledge about the relation between the housing type and location with respect to maintenance nature. For example it was found that housing facilities have similar deficiencies on all three campuses whereas the deficiencies in academic facilities are unique to the three different campuses. Furthermore, Housing and academic facilities have similar deficiencies in the area of old components and systems, such as compliance with the American Disability Act for fire protection (sprinkler systems and emergency lighting), and adequate space in bathrooms. As stated by Ng et al. (2006), the developed KDD system assisted in acquiring knowledge form the FCA that is far beyond traditional data analysis techniques.

In one of the latest researches, Lin and Soibelman (2007) developed a NLP based approach to assist Architectural/Engineering/Construction (A/E/C) information acquisition from the World Wide Web (WWW) concerning materials manufacturers. Due to the inconsistence of terms used for materials description, the developed approach made use of the extended Boolean model and domain knowledge thesaurus generated through automated web aggregator. The developed thesaurus is utilized to perform query expansion which takes place in two steps. In the first

step, set of terms related to each main subject (title) under query were generated with an AND/ OR association relation in an attempt to provide standardized search terms for different materials. For example, a “Translucent Roof Panels” would have Skylight, Fiberglass, and Natural Light as related terms with an AND association. However, Day lighting panels, Translucent roof assemblies, and Translucent roof systems would be related with an OR association. Consequently, new set of queries would be generated in the following manner: (“translucent roof panels”) AND (skylight), (“translucent roof panels”) AND (fiberglass), (“translucent roof panels”) AND (natural light), (“translucent roof panels”) OR (daylighting panels), (“translucent roof panels”) OR (translucent roof assemblies), and (“translucent roof panels”) OR (translucent roof systems). In the second step, a set of stemmed terms generated from the initial quarry terms were generated to account for the lexical variation in terms representation. Before augmenting the generated terms, they were checked using “WordNet”, an extensively used dictionary in NLP, to remove under-, over-, and mis-stemmed words. As reported by Lin and Soibelman (2007), the implementation of this approach enhances the retrieval and utilization of the WWW for A/E/C information acquisition.

The use of NLP techniques for the prediction of construction litigation outcomes is a research topic that has not been tackled so far. Since the fields of construction claim management and law are closely related, as discussed in section (2.2.1 And 2.2.2), it can be presumed that the advancements achieved in the use of NLP techniques in the legal domain can be adopted and further developed in the field of construction litigation outcomes prediction.

2.6 Machine Learning (ML)

The focus of this section is to provide background information about the nature of ML tools that could be used for creating a DSC legal decision support system for the construction industry and the different types of reasoning upon which they are based. The section will first provide some background on the different types of reasoning employed in ML, and then emphasis will be given to four types of ML tools, namely:

1. Support Vector Machines (SVM);
2. Naïve Bayes Classifiers (NB);
3. Rule Induction Classifiers; and
4. Latent Semantic Analysis (LSA).

2.6.1 Type of Reasoning

Before discussing the different ML tools reviewed in this section, one should develop an understanding of the different reasoning types upon which they are based. The following is a brief description of these reasoning types. In fact, classification is a process performed by humans on daily bases even without consciously noticing. In all cases, classifications performed by humans or computer systems (ML) fit into one of three categories namely deduction, abduction, and induction reasoning. The first type, deduction, is based on deriving rules from facts that are 100% assured (Bramer 2007). An example of this would be if for a fact it is known that all humans are mortal and that X is a human, then it could be deduced that X is mortal. This methodology for rule generation would be completely reliable if

all aspects related to a problem are 100% assured facts. However, this is a luxury that is rarely available in real life problems (Bramer 2007).

The second type of reasoning is based on truth of premises. Such type may not be necessarily correct. For example, if it is known that all dogs chase cats and that Y chases cats, then it is abducted that Y is a dog. Such rule may or may not be correct. There is no assurance that Y is a dog, for it might be any other animal that chases cats or even a human.

The third type of reasoning is based on learning from examples. If there exist enough examples in which the occurrence of X leads to Y, then is could be inducted as a rule that ***if*** X ***then*** Y (Shawe-Taylor and Cristianini 2000). Such methodology of reasoning is very reliable since all required knowledge about the relation between X and Y is present implicitly in the learning examples. Consequently, the majority of ML techniques adopted for the analysis in this chapter are based on *Induction Reasoning*.

2.6.2 Support Vector Machine (SVM) Classifiers

"SVM are learning systems that use hypothesis space of linear functions in high dimensional space, trained with a learning algorithm from optimization theory that implements a learning biased derived from statistical learning theory" (Shawe-Taylor and Cristianini 2000). Support vector machine classification aims to find a classification surface that best separates a set of training data points into classes in a high dimensional space (Nguyen et al. 2006). In its simplest linear form, a support

vector machine finds a hyperplane that separates a set of positive examples from the set of negative examples with maximum margin as shown in figure 2.1.

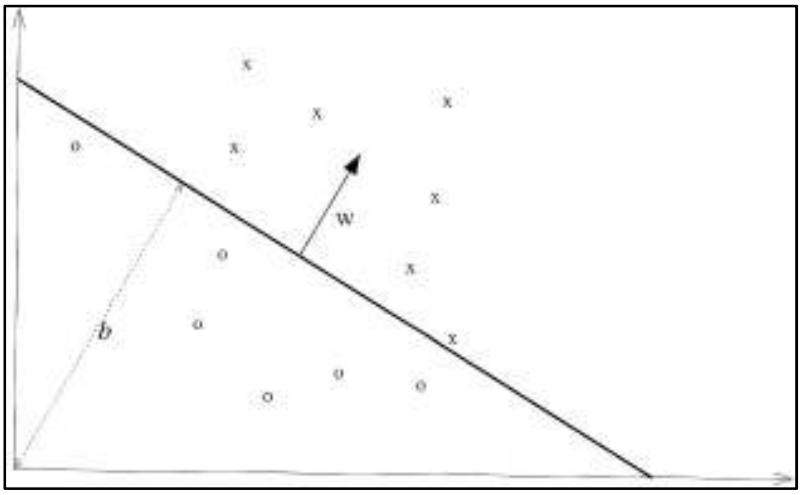


Figure 2.1 Maximum Margin Representation in SVM (Shawe-Taylor and Cristianini 2000)

Binary classification is frequently performed by using a real-valued hypothesis function, equation 2.1, where input x is assigned to the positive class if $f(x)\geq 0$; otherwise, it is assigned to the negative class.

$$y=<\mathbf{w}.\mathbf{x}>+b \qquad 2.1$$

For a binary linear separation problem a hyperplane is assigned to be $f(x) = 0$ where the separation (γ) is maximized. With respect to equation 2.1, the vector ***w*** (weight vector) and ***b*** (functional bias) are the parameters that control the function of the separation hyperplan (figure 2.1). In addition, ***x*** is the feature vector which may have different representations based on the nature of problem. For example, in text mining tasks, for a corpus including *n* number of documents, each document *d* is represented in the dimensional space in the form of a term vector (equation 2.2).

$$d_i=[t_1,t_2,\ldots,t_m] \qquad 2.2$$

Where i ε n, m = the number of words in the corpus after removing stop words, and t = frequency of the i^{th} term in the document. Previous researches in the field of construction and linguistics adopted representing each element in the vector by its normalized inverse term frequency (Salton 1989, Caldas et al. 2002, and Ng et al. 2006). This representation is selected so that terms appearing frequently in many documents have limited discrimination power (Salton 1989). This is done by multiplying the frequency of each term i by $\log(N/df_i)$, where N=total number of documents in the collection, and df_i=number of documents that contain the i^{th} term. In a vector space, each document vector represents a point (Ng et al. 2006).

SVM are applicable not only to problems of binary nature but also to multiclass classification nature. For a sample space **X** and output space **Y**, a binary classification problem will have **Y**= {-1, 1} while a multiclass one will have **Y**= {1, 2 …, m}.

From the above, the problem of classification is summarized to finding a hyperplane that separates the input data with maximum (γ). To further elaborate on this notion, one should first understand few basic concepts of the SVM. Figure 2.2 illustrates the geometric margin of two points from the hyperplane. In this case γ_i and γ_j defines the Euclidean distance of two points from the decision boundary in the input space. Consequently, the distribution of all margins over all points defines the functional margin distribution of the hyperplane with respect to a training set. In other words, the margin γ of a training set (figure 2.3) is the maximum geometric margin over all possible hyperplanes.

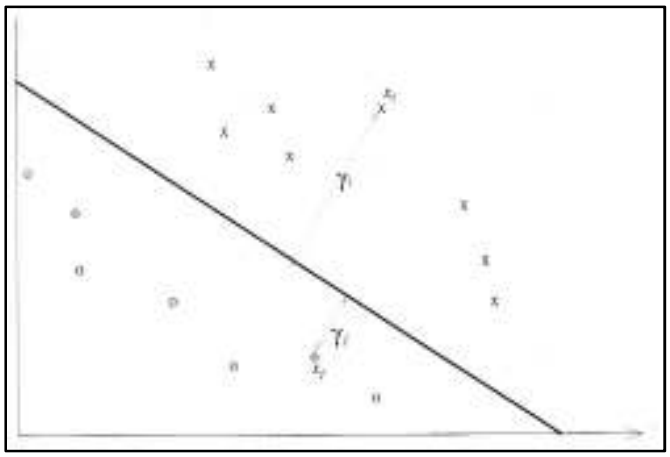


Figure 2.2 Geometric Margin Representation in SVM (Shawe-Taylor and Cristianini 2000)

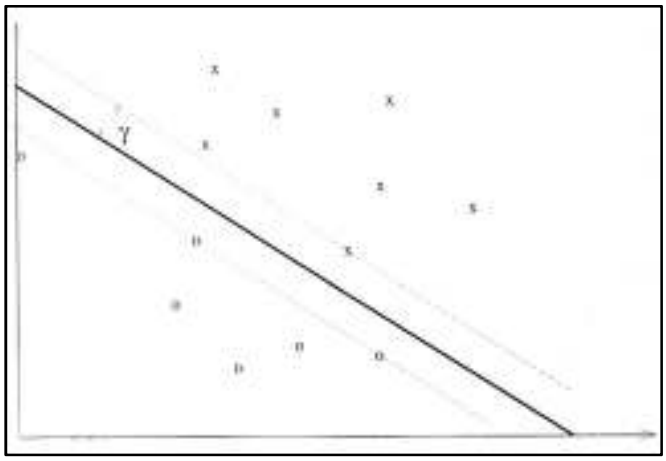


Figure 2.3 Hyperplane Representation in SVM (Shawe-Taylor and Cristianini 2000)

The first version of the algorithm that was the foundation for learning linear classification was introduced in 1956 by Frank Rosenblatt (Shawe-Taylor and Cristianini 1999). Rosenblatt's algorithm has proven guaranteed performance provided that there exists a hyperplane that separates the data set (Shawe-Taylor and Cristianini 2000). In this case the data are said to be linearly separable. However, a problem manifests itself if the data are not linearly separable. In the 1960s, Misky and Papert highlighted the limited computational ability of a linear

learning machine (Misky and Papert 1990). As stated by Shawe-Taylor and Cristianini (2000), complex real life problems are rarely linearly separable. In other words, they cannot be represented by a simple linear combination of given attributes. Consequently, a more sophisticated higher dimension space is needed for the representation of such problems in order for these complex problems to be linearly separable. As the literature in this field instigate, Kernel representation provides a solution to this problem by transforming the data into a higher dimensional feature space to enhance the computational power of linear machine learning (Shawe-Taylor and Cristianini 2000, Shawe-Taylor and Cristianini 1999, Platt 1999, and Mangasarian and Musicant 1999). Kernel machines have been initially devised for the binary setting. However, extensions to the multiclass case have been promptly proposed (e.g. Vapnik, 1998, Weston and Watkins 1999, and Crammer and Singer 2003). As shown earlier in equation 2.1, the representation of any data set in a feature space for linear machine learning is achieved as a dot product of the feature vector (***x***) and the weight vector (**w**). By introducing the appropriate Kernel function, one can map the data set to higher feature space (equation 2.3 and figure 2.4) transforming it from linearly inseparable to linearly separable. In this manner, the input space **X** is mapped into a new higher feature space **F** = $\{\emptyset(\mathbf{x})|\mathbf{x}\in X\}$.

$$\mathbf{x}=(x_1,\ldots,x_n)\rightarrow\emptyset(\mathbf{x})=\left(\emptyset_1(\mathbf{x}_1),\ldots,\emptyset_n(\mathbf{x}_n)\right)\text{or } k(\mathbf{x},\mathbf{y})=[\emptyset(\mathbf{x}).\emptyset(\mathbf{y})] \qquad 2.3$$

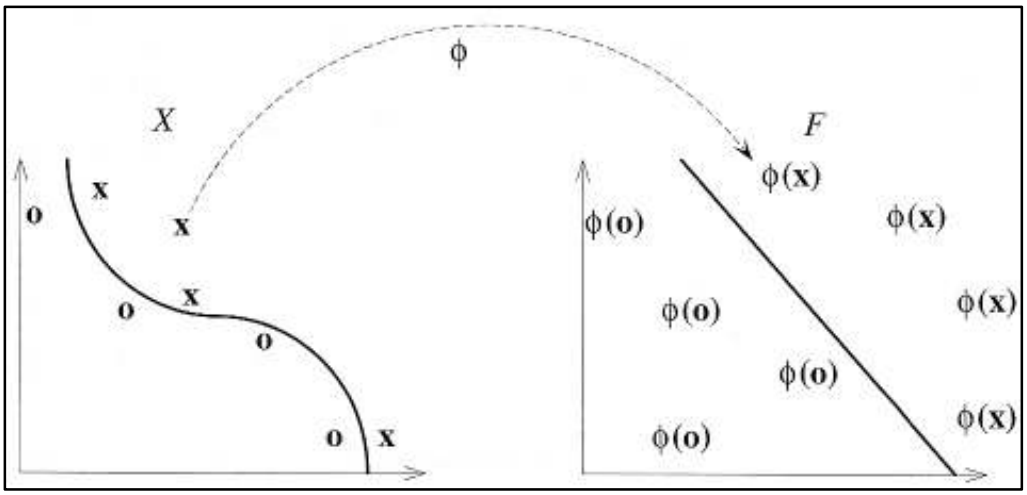


Figure 2.4 Kernel Transformation (Shawe-Taylor and Cristianini 2000)

2.6.3 Naïve Bayes Classifiers

The name Naïve Bayes is derived from two parts. The former relates to an assumption that is inherited in the performance of the classifier. Naïve Bayes Classifiers assumes that the values of attributes are irrespective of each other. That is effect of an attribute on the prediction is independent from the effect of others as will be discussed later. The latter relates to the name of the pioneering mathematician that is credited for its initial use. Reverend Thomas Bayes (1702 – 1761) was an English Presbyterian and Mathematician that is considered to be the first to apply *Probability Theory*, the basis of Naïve Bayes Classifiers, in an inductive manner.

Naïve Bayes Classifiers is a type of classifiers that do not implement rules to derive the classification, unlike rule induction classifiers that will be discussed later. The classification methodology adopted by Naïve Bayes Classifiers is based on the probability theory. In other words, it finds the most likely classification for an instance

among all available classes taking into consideration the presence of prior knowledge of other pieces of information. For example, a classifier calculates the odds of a case being classified to an Owner or a Contractor class while having prior knowledge of the significant legal factors occurrence. A decision is made based on the highest calculated probability for both classes. Figure 2.5 illustrates the mathematical bases of Naïve Bayes Classifiers. For more elaboration an illustrative example is adopted from Max Bramer's book Principles of Data Mining (2007). Table 2.1 includes 20 instances for the 6:30 pm train from London to a certain local station. Each instance records four attributes (namely day of the week, season of the year, wind status, and rain status) and a classification (either the train was on time, late for less than 10 minutes, very late beyond 10 minutes, or cancelled). As mentioned earlier, a prediction of a newly unseen instance would be decided as the highest probability for that instance to fall into one of the above mentioned four classes. Consequently, Naïve Bayes assumes that each instance is mutually exclusive and exhaustive. In other words, it only falls into one class and cannot be classified to more than one. Table 2.2 defines the conditional and prior probabilities of all attributes and classes. A conditional probability as given in equation 2.4, is read as the probability of attribute (a) happening with the prior knowledge of a classification falling in class (x). However, a prior probability means the probability of a certain class (x) happening based on the 20 instances recorded.

$$P(\text{attribute}=a|\text{class}=x) \qquad 2.4$$

Naïve Bayes Classification

Given a set of k mutually exclusive and exhaustive classifications $c_1, c_2, \ldots, c_k$, which have prior probabilities $P(c_1), P(c_2), \ldots, P(c_k)$, respectively, and n attributes $a_1, a_2, \ldots, a_n$ which for a given instance have values $v_1, v_2, \ldots, v_n$ respectively, the posterior probability of class c_i occurring for the specified instance can be shown to be proportional to

$$P(c_i) \times P(a_1 = v_1 \text{ and } a_2 = v_2 \ldots \text{ and } a_n = v_n \mid c_i)$$

Making the assumption that the attributes are independent, the value of this expression can be calculated using the product

$$P(c_i) \times P(a_1 = v_1 \mid c_i) \times P(a_2 = v_2 \mid c_i) \times \ldots \times P(a_n = v_n \mid c_i)$$

We calculate this product for each value of i from 1 to k and choose the classification that has the largest value.

Figure 2.5 Naive Bayes Classifiers Algorithm (Bramer 2007)

Table 2.1 Train Data for Naive Bayes Classifier

Day	Season	Wind	Rain	Class
Weekday	Spring	None	None	On time
Weekday	Winter	None	Slight	On time
Weekday	Winter	None	Slight	On time
Weekday	Winter	High	Heavy	Late
Saturday	Summer	Normal	None	On time
Weekday	Autumn	Normal	None	Very late
Holiday	Summer	High	Slight	On time
Sunday	Summer	Normal	None	On time
Weekday	Winter	High	Heavy	Very late
Weekday	Summer	None	Slight	On time
Saturday	Spring	High	Heavy	Cancelled
Weekday	Summer	High	Slight	On time
Saturday	Winter	Normal	None	Late
Weekday	Summer	High	None	On time
Weekday	Winter	Normal	Heavy	Very late
Saturday	Autumn	High	Slight	On time
Weekday	Autumn	None	Heavy	On time
Holiday	Spring	Normal	Slight	On time
Weekday	Spring	Normal	None	On time
Weekday	Spring	Normal	Slight	On time

Following the classifier algorithm given in figure 2.5 and data provided in table 2.1, a newly unseen instance with attributes day of the week = weekday, season of the year = winter, wind status = high, and rain status = heavy would be classified as very late based on the following calculations.

P(Class = on time) = 0.70 x 0.64 x 0.14 x 0.29 x 0.07 = 0.0013

P(Class = late) = 0.10 x 0.50 x 1.00 x 0.50 x 0.50 = 0.0125

P(Class = very late) = 0.15 x 1.00 x 0.67 x 0.33 x 0.67 = 0.0222

P(Class = cancelled) = 0.05 x 0.00 x 0.00 x 1.00 x 1.00 = 0.0000

Table 2.2 Naive Bayes Probability Calculations for Train Data Example

	Class = On time	**Class = Late**	**Class = Very late**	**Class = Cancelled**
Day = Weekday	9/14 = 0.64	1/2 = 0.5	3/3 = 1	0/1 = 0
Day = Saturday	2/14 = 0.14	1/2 = 0.5	0/3 = 0	1/1 = 1
Day = Sunday	1/14 = 0.07	0/2 = 0	0/3 = 0	0/1 = 0
Day = Holiday	2/14 = 0.14	0/2 = 0	0/3 = 0	0/1 = 0
Season = Spring	4/14 = 0.29	0/2 = 0	0/3 = 0	1/1 = 1
Season = Summer	6/14 = 0.43	0/2 = 0	0/3 = 0	0/1 = 0
Season = Autumn	2/14 = 0.14	0/2 = 0	1/3 = 0.33	0/1 = 0
Season = Winter	2/14 = 0.14	2/2 = 1	2/3 = 0.67	0/1 = 0
Wind = None	5/14 = 0.36	0/2 = 0	0/3 = 0	0/1 = 0
Wind = High	4/14 = 0.29	1/2 = 0.5	1/3 = 0.33	1/1 = 1
Wind = Normal	5/14 = 0.36	1/2 = 0.5	2/3 = 0.67	0/1 = 0
Rain = None	5/14 = 0.36	1/2 = 0.5	1/3 = 0.33	0/1 = 0
Rain = Slight	8/14 = 0.57	0/2 = 0	0/3 = 0	0/1 = 0
Rain = Heavy	1/14 = 0.0.07	1/2 = 0.5	2/3 = 0.67	1/1 = 1
Prior Probability	14/20 = 0.7	2/20 = 0.1	3/20 = 0.15	1/20 = 0.05

2.6.4 Rule Based Induction Classifiers

Decision trees, ADTrees, and Rules Classifiers are types of ML classifiers that adopt decision rules automatically generated from training examples or data sets to classify a newly unseen instance (Bramer 2007). Decision tree classifier is a special case in which the generated decision rules are fitted into a form of a tree, where each leaf represents a decision state (figure 2.6).

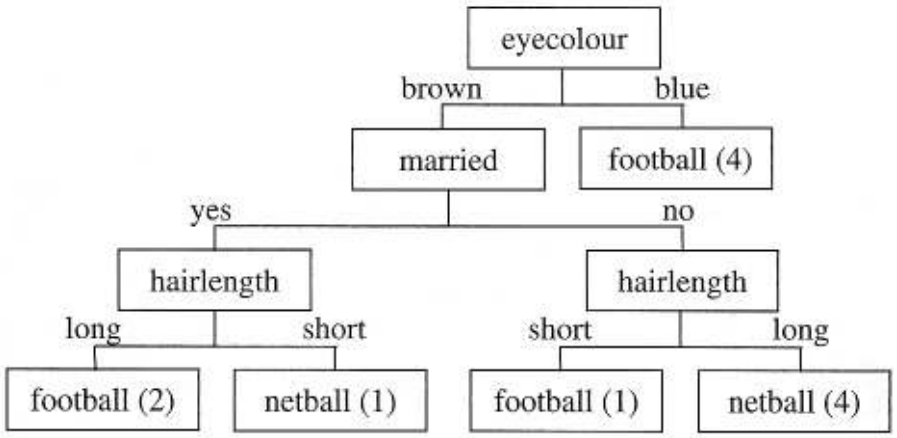


Figure 2.6 Decision Tree Representation (Bramer 2007)

For a given training data set, decision rules are derived based on a process known as *splitting on the value of attributes* or for short *splitting on attributes*. In such a process each attribute within a training set is tested for all of its possible values. For a discrete attribute, a rule (branch) is generated for each possibility. However, continuous attributes are branched normally at values like “less than or equal to a value”, “greater than or equal to a value”, “less than a value”, “greater than a value” … etc. A defined value for branching is defined as the split value. The splitting mechanism is continued until all attributes are tested and each rule is titled with just one classification. For more illustration, a widely used example by many authors to

illustrate the application of decision rules is adopted from Quinlan (1993), Witten and Frank (2000), and Bramer (2007). The data set illustrated in table 2.3 represents the decision (classification) of a golfer to play golf each day based on 4 attributes namely outlook, temperature, humidity, and wind status. The table provides information about 14 instances. Figure 2.7 illustrates the decision tree derived from the given data set based on the previously discussed research design and implementation.

Table 2.3 Golfer Data

Outlook	Temperature (^{o}F)	Humidity (%)	Wind Status	Class
Sunny	75	70	True	Play
Sunny	80	90	True	Don't play
Sunny	85	85	False	Don't play
Sunny	72	95	False	Don't play
Sunny	69	70	False	Play
Overcast	72	90	True	Play
Overcast	83	78	False	Play
Overcast	64	65	True	Play
Overcast	81	75	False	Play
Rain	71	80	True	Don't play
Rain	65	70	True	Don't play
Rain	75	80	False	Play
Rain	68	80	False	Play
Rain	70	96	False	Play

Rule decision algorithms, especially decision trees, were developed in the mid 1960s (Manning & Scheutze 1999). TDIDT short for Top-Down Induction of Decision Trees is a very powerful algorithm that initiated the application of decision trees for many classification systems (Bramer 2007). As stated by Bramer 2007 "Decision trees are widely used as a mean of generating classification rules because of the existence of a simple and powerful algorithm called TDIDT". TDIDT is an

algorithm that is applied in a recursive manner, keeps iterating till terminated, as shown in figure 2.8.

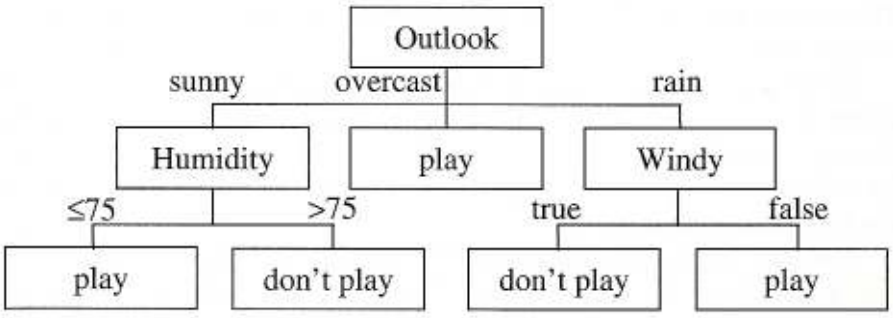


Figure 2.7 Decision Tree Representation for Golfer Example (Bramer 2007)

TDIDT: BASIC ALGORITHM

IF all the instances in the training set belong to the same class
THEN return the value of the class
ELSE (a) Select an attribute A to split on^{+}
(b) Sort the instances in the training set into subsets, one for each value of attribute A
(c) Return a tree with one branch for each *non-empty* subset, each branch having a descendant subtree or a class value produced by applying the algorithm recursively

$^{+}$ Never select an attribute twice in the same branch

Figure 2.8 The TDIDT Algorithm (Bramer 2007)

The simplicity of the implementation of decision trees led to its use in a variety of applications in the construction domain. In one of the most recent researches, Li and Lui (2008) implemented decision trees for the analysis of procurement strategies and task allocation between public and private sectors for infrastructure projects. Dogan et al. (2008) utilized decision trees for the determination of attribute weights in CBR models related to early cost prediction. Hegab and Nassar (2005)

implemented a decision tree methodology for the development of an expert system for commencement delay analysis. In addition, Arditi and Pulket (2005) implemented boosted decision trees for the development of litigation prediction model for the construction industry. Lee et al. (2004) implemented decision trees for the classification of change orders impact on productivity in construction projects. All of the above studies provide a strong support for the potentials of using rule based induction classifiers for the current research.

2.6.5 Latent semantic Analysis (LSA)

"Latent Semantic Analysis (LSA) is a theory and method for extracting and representing the meaning of words" (Landauer et al 2007). In a variety of AI techniques, the meaning of a word is determined through statistical computations applied on a large corpus of text. However, from human experience, a language can be learned by immersion without being explicitly taught. Consequently, the ability to understand the meaning of an expression composed of words can be acquired by humans through being surrounded by a certain language users. That directs to the belief that there exists a mechanism by which such a phenomenon takes place. The LSA theory attempts to model the mechanism of exactly how words and passage meanings can be constructed from experience with language. A corpus of related text imposes constraints on the meaning and semantic similarities of a word. For example, a word like "bank" can mean "a river side" or "an institution for financial transactions" based on the constraints imposed by the rest of words within a body of text. The theory of LSA hypothesizes that the meaning of a text is conveyed by the

words from which it is composed. Therefore, LSA is based on determining the meaning of a word by solving these constraints in a mathematical form by utilizing linear algebra, particularly, singular value decomposition (SVD). In other words, the meaning of a word is acquired by solving an enormous set of simultaneous equations that capture the contextual usage of words. It is not concerned with word position or segments.

Landauer et al. (2007) highlights the superiority of LSA over other machine learning techniques with respect human knowledge simulation. LSA has shown to reflect human knowledge in a variety ways (1) its measures highly correlates to humans' scores on standard vocabulary and subject matter tests; (2) it resembles humans' word sorting and category judgment; and (3) it accurately estimates passage coherence. Furthermore, it has proven outstanding results in inter-sentence similarity measurements (Choi et al. 2001). LSA has been extensively used in linguistic researches. Landauer et al. (2003a and 2003b) tested LSA in multiple-choice vocabulary tests and the task of determining the adequacy of expository essays contents. LSA scored in the high school student level. Foltz et al. (1998) researched the use of LSA to measure paragraph to paragraph coherence where it scored better than human coding. In other studies, LSA successfully modeled several laboratory findings in cognitive psychology (Howard et al 2007; Landauer 2002; Landauer and Dumais 1997; and Lund et al. 1995). It detected improvement in student knowledge level from before to after reading as well as human judges (Rehder et al. 1998; and Wolfe et al. 1998). In the medical field, LSA was used to

diagnose schizophrenia from patients' descriptions. It scored as well as experienced psychiatrists (Elvevag et al. 2005).

LSA is based on the concept of Vector Space Model implemented by SVM. However, the main advantage in LSA is that it utilizes a truncated space in which the number of features is reduced. LSA represents word and passage meanings in a form of mathematical averages. Word meanings are formulated as average of the meaning of all the passages in which it appears, and the meaning of a passage as average of the meaning of all the words it contains. LSA methodology applies SVD for the reduction of dimensionality in which all of the local word context relations are simultaneously represented. LSA, unlike many other methods, employs a preprocessing step in which the overall distribution of a word over its usage contexts, is first taken into account independent of its correlations with other words. This step improves LSA's results considerably. LSA then implements three well defined steps. Firstly, text document within a training corpus are represented in a form of matrix (figure 2.9). Each row of the developed matrix demonstrates a specific word in the training corpus. Each column of the matrix stands for a text document. Each cell contains the frequency with which the word of its row appears in the passage denoted by its column (Landauer et al. 2007). Consequently, a document collection including *n* documents and *m* features, which could be words, phrases, sentences, paragraphs … etc., are represented by an *m* by *n* matrix. Often, the number of features *m* is much higher than the number of documents *n* within the collection. Removal of stop words before performing matrix representation is not a necessity, due to the mathematical nature of the SVD, but it enhances its

performance by removing excess noise. The developed *m* by *n* matrix will contain zero and nonzero elements. Generally, a weighing function is applied to nonzero element to give lower weights to high frequency features that occur in many documents and higher weights to features that occur in some documents but not all (Salton and Buckley, 1991). Weighing functions are of two types namely local and global. The former relates to increasing or decreasing a nonzero element with respect to each document. The latter relates to increasing or decreasing a nonzero element across the whole collection of documents.

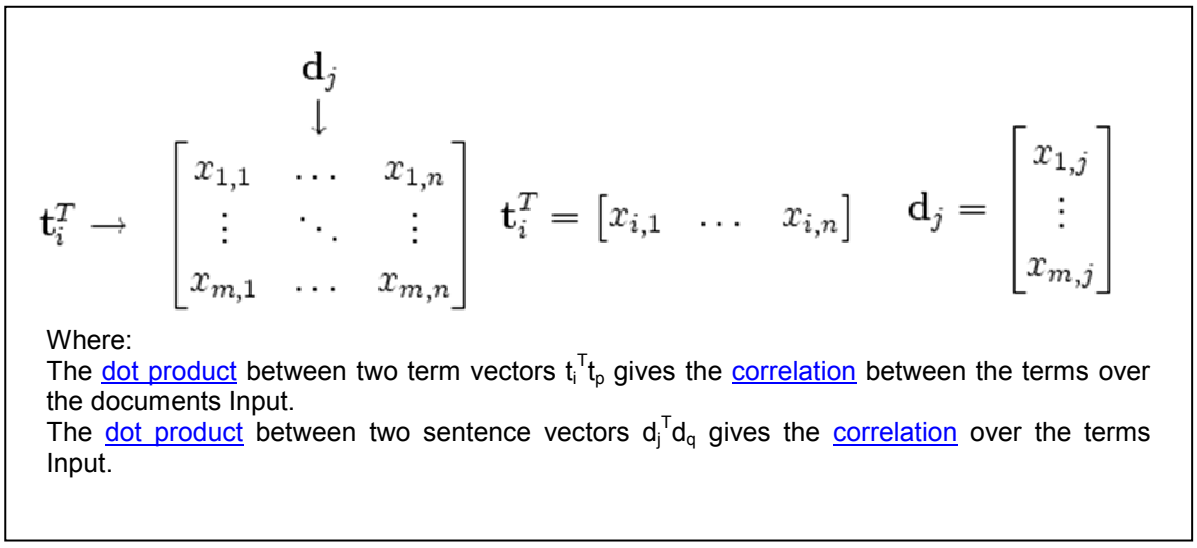


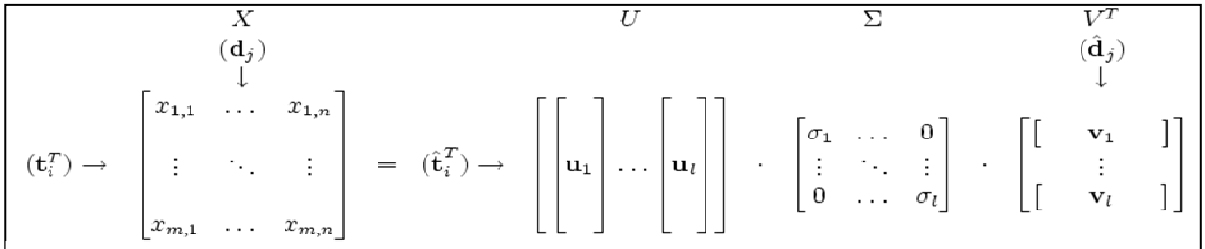
Figure 2.9 Matrix representation in LSA (Landauer et al. 2007)

Secondly, SVD is applied to the developed matrix to achieve an equivalent representation in a smaller dimension space (Choi et al. 2001). With SVD, a rectangular matrix is decomposed into the product of three other matrices (figure 2.10). One component matrix describes the original row entities as vectors of derived orthogonal factor values, another describes the original column entities in the same way, and the third is a diagonal matrix containing scaling values such that

when the three components are matrix-multiplied, the original matrix is reconstructed (Hofmann 1999).

Thirdly, the number of features adopted for analysis is determined (Truncation). Since the singular value matrix is organized in an ascending order based on the weight of each term, it is easy to decide on a threshold singular value below which terms significance is negligible, refer to (figures 2.10 and 2.11), (Dumais 1990). For an original matrix **A** with rank *k*, a newly truncated matrix $\mathbf{A_k}$ can be formulated by the dot product illustrated in equation 2.5. As stated by Landauer et al. (2007), truncating the SVD and creating $\mathbf{A_k}$ is what captures the important underlying semantic structure of words and documents. Words that are similar in meaning are near to each other in *k dimensional space*.

$$A_i = \sum_{i=1}^{k} u_i \sigma_i v_i^T \rightarrow A_k = U_k \Sigma_k V_k^T \qquad 2.5$$



Where:

- The term-document matrix *X* represents how important a term is in a given document.
- Σ is a diagonal matrix representing the weights of the concepts. Usually SVD algorithm produces a Σ with $\sigma_1 > \sigma_2 > ... > \sigma_k > 0$.
- The columns of *U* are the terms in concept space, the rows of *V* are the documents in the concept space. Since SVD is used, *U* and *V* are unitary matrices, that is, the rows of *U* and the columns of *V* are of unit length and are pairwise orthogonal.
- Not all concepts are necessarily used. Only those with sufficiently large singluar values (i.e. σ-s) may be selected. The result matrix is the best low-rank approximation to *X* in Frobenius norm.

Figure 2.10 SVD Matrix Representation in LSA (Dumais 1990)

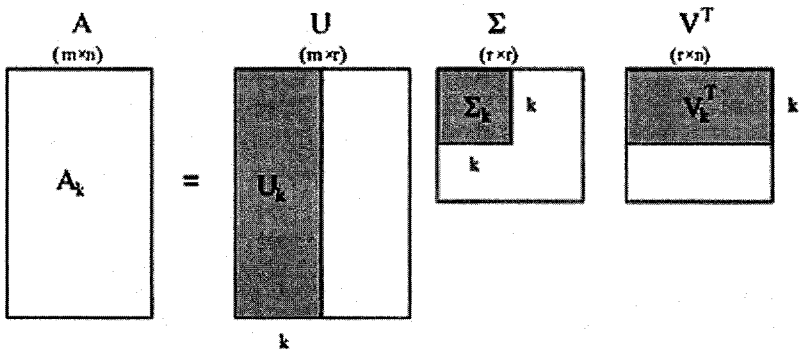


Figure 2.11 K Dimensional Space Representation in LSA (Dumais 1990)

By representing any document in the generated concept space, it is then possible to calculate "distance" (metric) on the set of such document representations thus computing whether two such representations are close which usually implies that the documents themselves are related. This notion makes LSA a very strong tool for document classification.

For more elaboration, an example is adopted from Landauer et al (2007). Figure 2.12 provides titles for topics on music and baking. Figures 2.13 and 2.14 illustrate the developed word by document matrix for the topics collection. Figure 2.15 shows the SVD of the example word by document matrix reduced to 2 features (*k*=2). Figure 2.16 shows a plot of words represented by squares and documents represented by rectangles after truncation. The (*x,y*) pairs of each point is defined as

x = first dimension or column of matrix **U** or **V** multiplied by first singular value.

y = second dimension or column of matrix **U** or **V** multiplied by second singular value.

Similarities between words and documents can be determined based on angles between vectors. Consequently, from figure 2.12, it can be deduced that document M4 “A Perspective of Rock Music in the 90’s” and M1 “Rock Music in the

1960's" are the closest documents to M3 "Drum and Base Composition". In addition, the word "Music" is most similar to "Rock" and "Composition" in the document collection.

Label	*Titles*
M1	*Rock* and *Roll Music* in the 1960's
M2	Different *Drum Rolls*, a Demonstration of Techniques
M3	*Drum* and Bass *Composition*
M4	A Perspective of *Rock Music* in the 90's
M5	*Music* and *Composition* of Popular Bands
B1	How to Make *Bread* and *Rolls*, a *Demonstration*
B2	*Ingredients* for Crescent *Rolls*
B3	A *Recipe* for Sour*dough Bread*
B4	A Quick *Recipe* for Pizza *Dough* using Organic *Ingredients*

Figure 2.12 Titles for Topics on Music and Baking (Landauer et al. 2007)

LSA implementation includes another fold. Once a truncated space of a dataset is produced, queries can be performed. Query in LSA can be defined as finding features or documents within the generated space similar to newly introduced ones. Deerwester et al. (1990) refers to representing a query in a truncated vector space as *a pseudo-document*. "A query is the weighted sum of its feature vector scaled by the inverse of the singular values, this individually weights each dimension in the *k*-dimensional feature-document vector space" (Landauer et al. 2007). A newly introduced query to the truncated feature space can be represented as per equation 2.6, where $\mathbf{q}^T$ is a vector containing zero and nonzero weighted frequency values of features in the newly introduced document. Similarity measures can then be implemented based on angles between vectors as mentioned earlier (Letsche and Berry 1997).

$$query = \boldsymbol{q}^T U_k \Sigma_k^{-1} \quad 2.6$$

Types	*Documents*								
	M1	M2	M3	M4	M5	B1	B2	B3	B4
Bread	0	0	0	0	0	1	0	1	0
Composition	0	0	1	0	1	0	0	0	0
Demonstration	0	1	0	0	0	1	0	0	0
Dough	0	0	0	0	0	0	0	1	1
Drum	0	1	1	0	0	0	0	0	0
Ingredients	0	0	0	0	0	0	1	0	1
Music	1	0	0	1	1	0	0	0	0
Recipe	0	0	0	0	0	0	0	1	1
Rock	1	0	0	1	0	0	0	0	0
Roll	1	1	0	0	0	1	1	0	0

Figure 2.13 The 10X9 Word by Document Matrix with Word Frequencies Corresponding to the Titles in Figure 2.12 (Landauer et al. 2007)

Types	*Documents*								
	M1	M2	M3	M4	M5	B1	B2	B3	B4
Bread	0	0	0	0	0	.474	0	.474	0
Composition	0	0	.474	0	.474	0	0	0	0
Demonstration	0	.474	0	0	0	.474	0	0	0
Dough	0	0	0	0	0	0	0	.474	.474
Drum	0	.474	.474	0	0	0	0	0	0
Ingredients	0	0	0	0	0	0	.474	0	.474
Music	.347	0	0	.347	.347	0	0	0	0
Recipe	0	0	0	0	0	0	0	.474	.474
Rock	.474	0	0	.474	0	0	0	0	0
Roll	.256	.256	0	0	0	.256	.256	0	0

Figure 2.14 The 10X9 Weighted Word by Document Matrix Corresponding to the Titles in Figure 2.12 (Landauer et al. 2007)

Matrix U-Type Vectors									
Bread	**.42**	**–.09**	–.20	.33	–.48	–.33	.46	–.21	–.28
Composition	**.04**	**–.34**	.09	–.67	–.28	–.43	.02	–.06	.40
Demonstration	**.21**	**–.44**	–.42	.29	.09	–.02	–.60	–.29	.21
Dough	**.55**	**.22**	.10	–.11	–.12	.23	–.15	.15	.11
Drum	**.10**	**–.46**	–.29	–.41	.11	.55	.26	–.02	–.37
Ingredients	**.35**	**.12**	.13	–.17	.72	–.35	.10	–.37	–.17
Music	**.04**	**–.35**	.54	.03	–.12	–.16	–.41	.18	–.58
Recipe	**.55**	**.22**	.10	–.11	–.12	.23	–.15	.15	.11
Rock	**.05**	**–.33**	.60	.29	.02	.33	.28	–.35	.37
Roll	**.17**	**–.35**	–.05	.24	.33	–.19	.25	.73	.22
Matrix Σ-Singular Values									
	1.10	0	0	0	0	0	0	0	0
	0	**.96**	0	0	0	0	0	0	0
	0	0	.86	0	0	0	0	0	0
	0	0	0	.76	0	0	0	0	0
	0	0	0	0	.66	0	0	0	0
	0	0	0	0	0	.47	0	0	0
	0	0	0	0	0	0	.27	0	0
	0	0	0	0	0	0	0	.17	0
	0	0	0	0	0	0	0	0	.07
	0	0	0	0	0	0	0	0	0
Matrix V-Document Vectors									
M1	**.07**	**–.38**	.53	.27	.08	.12	.20	.50	.42
M2	**.17**	**–.54**	–.41	.00	.28	.43	–.34	.22	–.28
M3	**.06**	**–.40**	–.11	–.67	–.12	.12	.49	–.23	.23
M4	**.03**	**–.29**	.55	.19	–.05	.22	–.04	–.62	–.37
M5	**.03**	**–.29**	.27	–.40	–.27	–.55	–.48	.21	–.17
B1	**.31**	**–.36**	–.36	.46	–.15	–.45	.00	–.32	.31
B2	**.19**	**–.04**	.06	–.02	.65	–.45	.41	.07	–.40
B3	**.66**	**.17**	.00	.06	–.51	.12	.27	.25	–.35
B4	**.63**	**.27**	.18	–.24	.35	.10	–.35	–.20	.37

Figure 2.15 The SVD of the Weighted Word by Document Matrix Corresponding to the Titles in Figure 2.12 (Landauer et al. 2007)

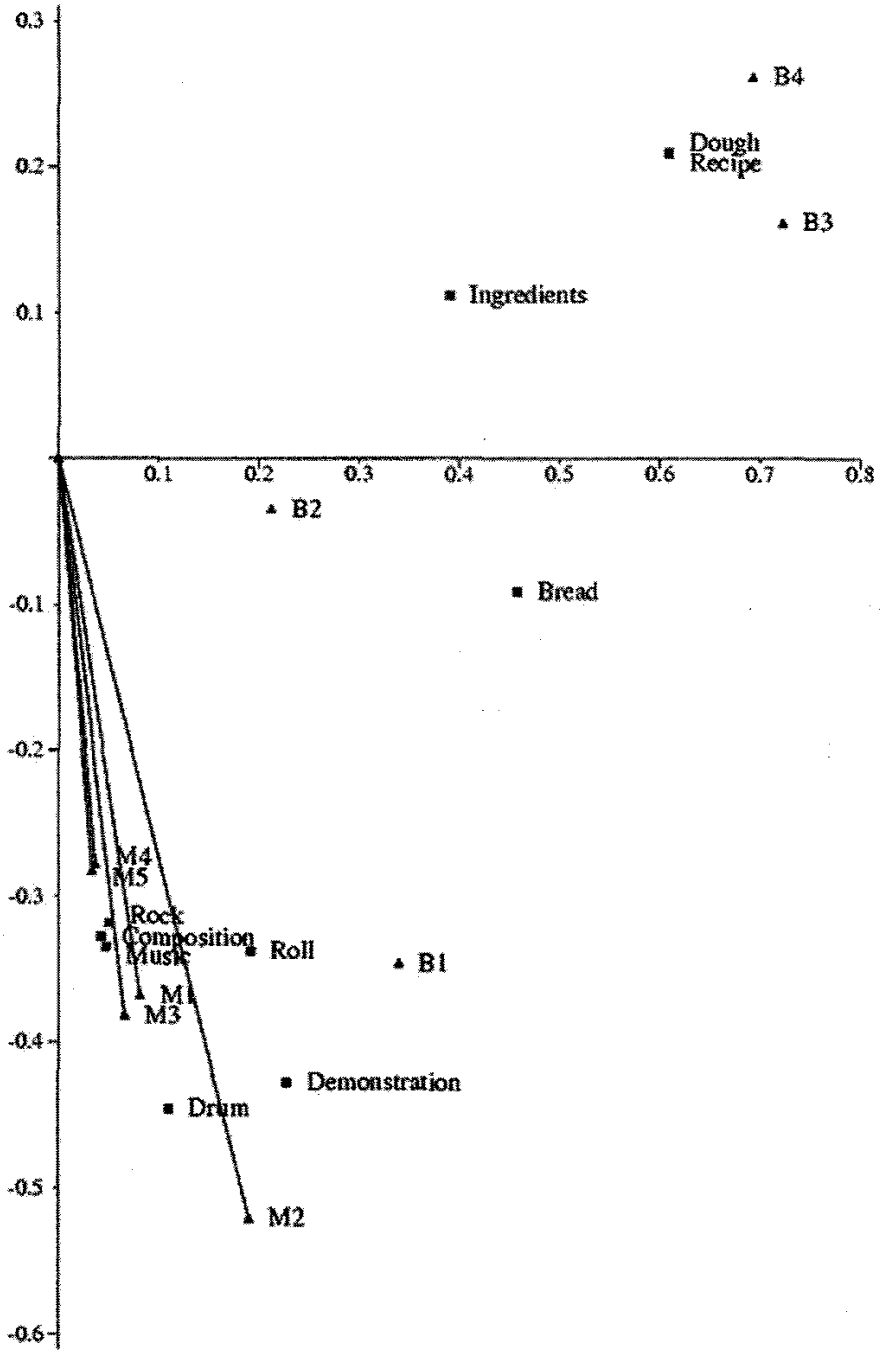


Figure 2.16 The Rank-2 LSA Vector Space for the Music/Baking Titles Collection (Landauer et al. 2007)

2.7 Differing Site Conditions (DSC)

The focus of this section of the chapter is to provide an overview of the definition of DSC in the construction industry, background information about the implementation of DSC clauses in construction contracts, the contractual context of DSC clauses, and the types of DSC.

One of the major and most commonly encountered disputes that had raised a lot of questions and enforced alterations on the way Owners and Contractors perceive risk allocation in construction projects is DSC (El-Saadi 1998). Originally, owners' approach to handling risk entailed allocating most risk on contractors (Levin 1988). As a rule of thumb, a directly proportional relation exists between the risk assumed by the contractor and the contingency imposed on his bid (Krol 1993). In other words, not including a DSC clause in the contract leads each party to take extreme measures. Faced with the burden of most DSC risk, contractors tend to include larger contingencies in their bid prices as a method for protecting themselves against the many uncertainties of construction projects. This consequently leads owners who allocate these risks contractors, to incur higher values for the performed works in the case that no DSCs encountered. Nevertheless, by agreeing to share DSC risk and allowing for the reimbursement of costs incurred by contractors due to DSC, contractors would reasonably price their bids by including a smaller contingency, and would not claim damages under breach of contract. In the latter case, "The owner, whether public or private, minimizes the risk of being held in breach of contract for failing to adequately describe the physical conditions at the job site" (Levin 1988).

2.7.1 Differing Site Conditions Clauses

Differing Site Conditions (DSC) clauses have many forms since contracts allow for different degrees of variability in site conditions. Some clauses are restricted to handling conditions which vary from those described in the contract documents irrespective of any unexpected conditions encountered that were not referred to. Others cover under their scope only materially different conditions from those expected in similar projects. Each of these categories allocates different level of risk on both contracting parties. However, there are agreed upon concepts that are represented in standard forms of contracts like FAR (Federal Acquisition Regulations), AIA (American Institute of Architects), FIDIC (Fédération Internationale Des Ingénieurs-Conseils, French for the International Federation of Consulting Engineers), and the Engineers Joint Contract Committee. The concepts can be utilized to formulate a definition for DSC as “physical site conditions at the job site which differ materially from the conditions represented in a construction contract or the condition that normally could be expected in a job of similar nature” (Levin 1988).

The definition of DSC must be integrated with an understanding of the characteristics of DSC clauses to comprehend its application. DSC clauses have unique characteristics, and do not lead to any implied rights. A DSC clause must be present in a contract for the contractor to have the right to any additional payment under the contract. Once construction begins on a project under a contract that is silent about the risk of unforeseen conditions, a contractor bears the risk of running into conditions that were not expected at the time they submitted their bid even though they significantly increase the cost of performance (Iacobelli 1994). This

draconian rule had always been the governing factor when court decisions are required. However, judges were confronted with cases that triggered their sense of fair judgment with regards to whether the misrepresentation of the physical conditions was either intentional or caused by neglect on part of the owner. However, judges also made judgments that question the foreseeability of the conditions and the level of prudency of the contractor in interpreting the contract documents that did not mention DSC.

2.7.2 History of Differing Site Conditions Clause (DSC)

The literature shows that the federal government was a pioneer in using DSC clauses. November 22nd, 1921 is recoded as the first date a DSC clause titled “changed conditions”, which was later titled ”Differing Site Conditions”, was implemented (Tarkoy, unpublished book, 2008). On August 20th, 1926 the first standard general conditions for construction that includes a “changed conditions” clause was approved by the president of the United States for use by the federal government in their contracts (Tarkoy, unpublished book, 2008). From that date on, Federal Regulations made the use of DSC clause compulsory in all U.S. Government Contracts. It was incorporated as part of the Federal Acquisition Regulations (FAR) to prevent contractors from bidding on a worst-case-scenario basis (North Slope 1988). This clause allocates to the government the risks for conditions that the contract documents fail to disclose, but leaves upon the contractor the costs of encountering those conditions described in the contract (Erickson-Shaver 1985). Because the DSC clause alleviates the need for contractors

to insert speculative contingency costs in their bids, it reduces inflated bidding, and the government presumably saves money by getting lower bids (Weeks Dredging 1987, and North Slope 1988). Therefore, for over a half a century the DSC clause has been used in government contracts and has been interpreted by the courts. The purpose of the clause has been to shift the risk of adverse subsurface or latent physical conditions from the contractor, who normally bears such risk under a fixed-price contract, to the government. While it is recognized that the DSC clause is a risk shifting mechanism, it does not shift all unanticipated risk in a project's site conditions to the government. The Federal Circuit Court articulated the purpose of the DSC clause as follows: The government bears only those risks that encourage "more accurate bidding." Those risks are shifted to the government so that contractors will not add to their bids the cost of assessing whether adverse subsurface conditions exist or the cost of confronting such conditions if and when they are encountered.

The standard DSC clause defines a differing site condition and provides the procedures and requirements a contractor must follow before it is able to recover an equitable adjustment to the contract. It provides that when a contractor encounters a DSC, it must promptly notify the contracting officer (CO) in writing before the conditions are disturbed. The clause also defines the two types of DSC as follows: (Type 1) subsurface or latent physical conditions at the site which differ materially from those indicated in this contract, or (Type 2) unknown physical conditions at the site, of an unusual nature, which differ materially from those ordinarily encountered and generally recognized as inhering in work of the character provided for in the

contract. It also grants authority to the CO to make an equitable adjustment to the contract if the CO determines the alleged DSC satisfies the definition provided in the clause.

After the incorporation of the DSC clauses by the FAR, similar clauses have been included in other standard contract forms like AIA (American Institute of Architects), FIDIC (Fédération Internationale Des Ingénieurs-Conseils, French for the International Federation of Consulting Engineers), and the Engineers Joint Contract Committee.

2.7.3 Type of Differing Site Conditions (DSC)

As mentioned earlier, analysis of the language of the DSC clause of the federal government contracts (FAR) addresses two types of differing site conditions (Type 1 and Type 2). The former relates to physical conditions which differ materially from those indicated in the contract documents. The latter authorizes compensation "equitable adjustment" for unknown conditions which differ materially from those that would normally be encountered in projects of similar nature. As stated by Levin (2008), Type 2 Differing Site Conditions are rarely considered by both private and public owners in their contracts. Construction law literature explains that in order, to prevail on a Type 1 DSC claim, a plaintiff must show: (1) the contract documents affirmatively indicate subsurface conditions; (2) she acted as a reasonably prudent contractor in interpreting the contract documents; (3) she reasonably relied on the indications of subsurface conditions in the contract; (4) the subsurface conditions actually encountered differed materially from those indicated in the contract; (5) the

actual subsurface conditions were not reasonably foreseeable; and (6) her damage was attributable to the materially different subsurface conditions (Weeks Dredging 1987). Consequently, the threshold issue of whether a contractor is eligible for an equitable adjustment for a Type 1 DSC at a project site depends on the soil conditions indicated in the contract. The United States Court of Appeals for the Federal Circuit has made it clear that a contractor cannot be eligible for an equitable adjustment for DSC unless the contract indicated what those conditions would supposedly be (Weeks Dredging 1987). Courts that have addressed Type 1 DSC have found *indications* of the site conditions in the contract in order to consider that the contractor encountered this type of DSC. In the context of Type 1 DSC, while it is true that a contract *indication* need not be explicit or specific, the contract documents must still provide sufficient grounds to justify a bidder's expectation of latent conditions materially different from those actually encountered. In other words, the difficulty in Type 1 DSC inquiry is showing whether the condition differed materially from the affirmative representations in the contract. Contract indications may be implicit, but there must be sufficient indications of the condition to induce a reasonable reliance in the bidder that subsurface conditions would be more favorable than those encountered (Weeks Dredging 1987). As a consequence, determining whether a contract contained indications of a particular site condition is a matter of contract interpretation and thus presents a question of law. As illustrated in Travelers Casualty, v. the United States of America (2007), unlike traditional contract interpretation, in a differing site condition claim, a contractor is permitted to make inferences from a contract's implications. Interpretation of contract indications

requires the United States Court of Federal Claims to place itself into the shoes of a reasonable and prudent contractor. The implications in the contract need only be sufficient to impress a reasonable bidder. When a contract's language is unambiguous, it must be given its *plain and ordinary* meaning. When determining the plain meaning of a contract, a court must first determine what documents are actually parts of that contract. Documents will be considered part of a contract only when the intention to include it is clearly manifested. The key distinction between patent and latent ambiguity is in the way the law treats them and the corresponding effect on the contracting parties' rights and obligations. In common law, ambiguities are generally interpreted against the drafter. In the context of federal contracts, contractors are required to inquire about patent ambiguities before making bids. The purpose of requiring pre-bid inquiry is to prevent contractors from taking advantage of ambiguities in government contracts by adopting narrow interpretations in preparing its bids and then, after the award, seeking equitable adjustments to perform additional work the government actually wanted. The Federal Circuit, of the Court Appeals however, has not given the patent ambiguity doctrine broad application (Travelers 2007). Because the doctrine has the effect of relieving the government from consequences of its own poorly drafted contracts, the doctrine has been applied only to contract ambiguities that are judged so patent and glaring that it is unreasonable for a contractor not to discover and inquire about them. A court's finding of a latent ambiguity, however, does not automatically mean a favorable result for the plaintiff. The court will only adopt the contractor's interpretation of a latent ambiguity if its interpretation is reasonable.

On the other hand, to prevail on a Type 2 DSC claim, a plaintiff must show: (1) the encountered subsurface conditions were not reasonably foreseeable; (2) she did not have prior knowledge of the existence of the subsurface conditions; (3) the encountered conditions vary from the norms in similar construction projects (Levin 1988). In order for a contractor to recover for a Type 2 DSC, the condition must have existed at the time the contract was executed (North Slope 1988). Analogous to the rule that a DSC must exist before the execution of the contract, a contractor typically cannot recover for a post-award phenomenon considered an act of God. Generally, the government, under the standard DSC clause, does not assume an obligation to compensate a contractor for additional costs or losses it incurs resulting solely from weather conditions, which neither party expected or could anticipate and not from any act or fault of the government. Weather conditions generally are considered to be acts of God (North Slope 1988). The general rule is that the risk of severe weather in a particular region is not shifted to the government via the DSC clause. For example excessive rainfall is not in and of itself a DSC for which price and time adjustments are to be made under the DSC clause. Likewise excessive rainfall is not in and of itself a suspension of work nor is the CO under a duty to suspend merely because of such rainfall. But when excessive rainfall in interaction with a drainage area makes specified performance impossible a DSC does exist and the CO, if he wants work done, must change the specifications so as to make it possible. Within the context of a Type 2 DSC, where the Government has elected not to pre-survey and represent the subsurface conditions with the result that a contractor must demonstrate that he has encountered something materially different from the *known*

and the *usual*. This is necessarily a stiffer test because of the wide variety of materials ordinarily encountered when excavating in the earth's crust (North Slope 1988). Consequently, in determining whether a particular condition is unusual, the encountered condition is judged against the normal conditions for the area. Legally, unusual conditions with respect to a DSC claim are judged by the normal conditions for the area. The condition must significantly deviate from the norm for the area and the type of work (Servidone 1990). For example, difficulties caused by the combination of expansive clay soils and precipitation are the usual and reasonable problems encountered when expansive clay soils interact with moisture and do not constitute a Type 2 DSC.

2.8 Chapter summary:

Case-Based Reasoning has showed to be a very powerful tool in the implementation and utilization of previous knowledge learned from experience. It has been implemented as a potential solution to variety of problems in the construction domain including litigation outcomes prediction. However, a crucial aspect of the use of CBR models is the extraction of previous knowledge to form the cases of the case base. Since this knowledge includes significant amount of textual material expressed in human language, the need for tools that are capable of effectively analyzing textual material and efficiently retrieving pertinent information from them has become a necessity. As mentioned earlier, the accuracy of the output of a CBR system is largely dependent on the availability of reliable information about the attributes used to define the training cases. As Arditi and Pulket 2005 state “Finding a complete and

reliable set of training examples is difficult in construction litigation cases". The use of natural language processing techniques NLP can enhance and facilitate the use of construction litigation prediction models. Automatic case classification and knowledge extraction can be improved through NLP techniques (Bruninghause and Ashley 2001). This notion is greatly supported by the use of NLP approaches as a solution to different problems related to enhancing information models, document integration, and inter-organizational systems in construction. Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques have been employed extensively through a variety of automated and semi-automated tools (Labidi 1997). Text mining methodologies, document clustering techniques, controlled vocabulary schemes, and web based models were some of the techniques utilized to perform the above mentioned tasks (Caldas and Soibelman 2003). However, its use to enhance construction litigation outcomes prediction has not yet been attempted. The highly sophisticated electronic information storage and retrieval systems available for researching the law and legal precedent are extremely complex and time consuming. Sometimes this complexity creates problems for information seekers and can limit their access to relevant information. Consequently, accurate legal decisions within the construction domain are exceedingly time consuming and may require knowledgeable professionals that are on very high demand to provide the needed decision support.

Investigation of ML techniques showed the superiority of the induction reasoning over other reasoning methodologies. This investigation further highlighted

the high potential for using SVM, Naïve Bayes, and Rule Induction Classifiers for extracting novel information hidden within textual representations.

DSC clauses were introduced by the Federal Government to lower contingency measures adopted by contractors and in return lower bid prices. These clauses provide a measure of assurance for contractors to recover from extra costs due to unanticipated site conditions. These clauses created some problems due to their abuse by some contractors. For one, claims for DSC have become a custom tactic to be followed by contractors to recover from cost overruns, misinterpretation of anticipated conditions, and poor project coordination. This has led owners and engineers to hold a hard position when reviewing contractors' legitimate claims related to unforeseen conditions and associated costs. Furthermore, the literature in this area illustrates that the process of proving a DSC requires tremendous time and effort for factual examining. Consequently, the presence of an automated legal support for DSC in the construction industry that utilizes standardized methodology for (1) automated identification of significant legal factors that affects litigation outcomes of DSC disputes; (2) automated prediction of litigation outcomes of DSC Disputes; and (3) automated extraction of precedent DSC cases similar to newly un-encountered ones will reduce the time required and costs incurred by construction firms and improve overall project control.

CHAPTER 3
A STATISTICAL ANALYSIS OF FACTORS AFFECTING LITIGATION OUTCOMES IN DIFFERING SITE CONDITIONS DISPUTES

3.1 Introduction

The overall objective of this chapter is to analyze the main legal factors that govern litigation outcomes in DSC disputes. This objective is undertaken as a first step in the development of a construction legal decision support methodology based on statistical modeling and machine learning. The focus of this chapter, therefore, is to illustrate the design implementation of discrete choice prediction models for identifying the legal factors governing DSC disputes. The developed statistical models will aim to (1) detect the effect of each identified legal factor on the prediction of the winning party; (2) identify the best combination of legal factors with the highest significance on the prediction model; and (3) prioritize the identified legal factors according to their importance to DSC disputes.

As claims and disputes increase, the construction industry struggles to find ways to equitably and economically resolve them. As illustrated earlier in chapter 2 “Literature review”, a number of researchers in AI fields have developed tools and methodologies for modeling judicial reasoning and predicting the outcomes of construction litigation cases. However, their success was always bound by the input parameters they consider. In an attempt to provide an outcome prediction system for Differing Site Condition (DSC) claims in the construction industry, this chapter provides as a first step, a statistical analysis of a number of differing site condition cases from the Federal Court of New York in an endeavor to derive a set of

significant legal factors that governs litigation outcomes prediction concerned with this type of claims. The following sections of this chapter will therefore explain the implementation of the developed Statistical Models, as well as the results of these models and discussion of the main findings of the implementation of the models.

3.2 Design and Implementation of Statistical Models

The objective of this chapter is to identify, quantify, and measure the impact of significant legal factors on the prediction of outcomes of DSC claims in the construction industry. Consequently, this chapter provides a statistical analysis of set of 60 precedent cases from the Federal Court of New York in an effort to derive a comprehensive set of significant legal concepts that govern litigation outcomes of DSC claims. To this end, the main steps of the design and implementation of the proposed statistical model include (Figure 1.1): (1) developing a corpus of construction DSC cases; (2) identifying a set of legal factors that constitute the bases of judgments in construction DSC cases; and (3) developing statistical models that relate the likelihood of a DSC cases being judged in favor of one party over the other to the identified set of legal factors. It is important to note here that the developed prediction models are used mainly as a vehicle for determining the significant factors in DSC claims rather than as a decision support tool. The proposed statistical modeling approach will create and compare the outputs of Discrete Binary Probit Choice Model and Discrete Binary Logistic Regression Model (a) to identify the effect of each extracted factor on the prediction of the winning party; (b) to identify the best combination of factors with the highest significance on

the prediction model; and (c) to perform a sensitivity analysis to priorities the most significant legal factors. The statistical modeling approach will therefore be composed of three main stages:

1. Data acquisition and preparation;
2. Binary Probit model implementation; and
3. Binary Logistic model implementation.

3.2.1 Data Acquisition and Preparation

Corpus based approaches have become increasingly important in providing the basic data for prediction model (Robinson 2004). The scope of work under this research utilizes data from a web legal case retrieval engine (LexisNexis) for the statistical analysis of legal factors in DSC conflicts and disputes. LexisNexis provides access to over 32,000 legal, news, and business sources. Furthermore, it clusters legal cases in subdivisions based on states (LexisNexis 2008). An initial corpus composed of 60 DSC precedent cases was collected. The gathered corpus, which covers a time interval from 1912 to 2005, was collected from the Federal Court of New York due to the large number of construction precedent cases in this jurisdiction. Out of the gathered 60 cases there are: (1) 32 cases judged in favor of Owner versus 28 cases judged in favor of Contractor; (2) 28 cases (46.67%) of the cases are first, second, or third appeals; and (3) 32 cases (53.33%) of the cases are non-appeals. Out of the 28 appeal cases:

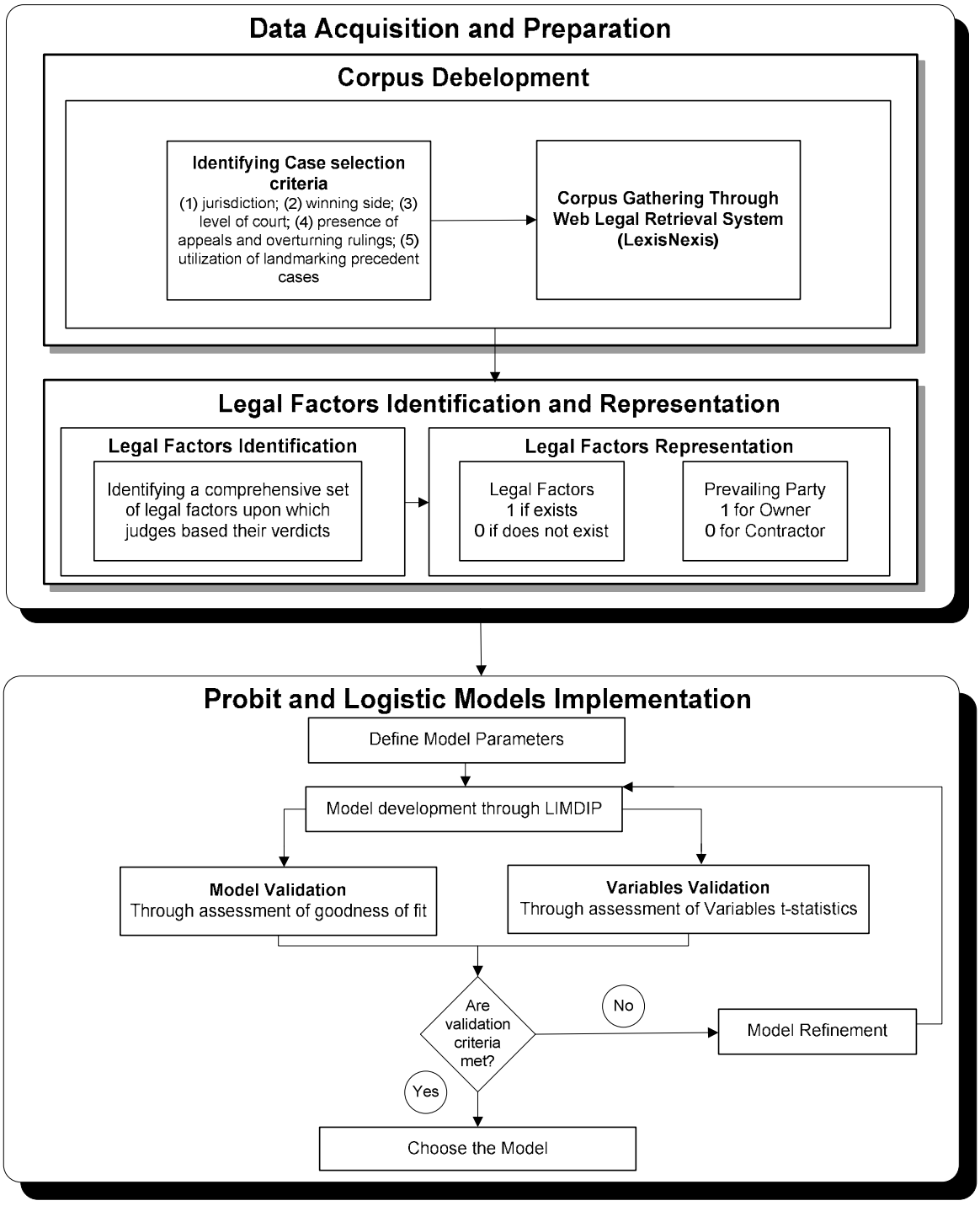


Figure 3.1 Statistical Modeling Approach

(1) 14 cases were judged in favor of Owner in comparison to 14 cases judged in favor of Contractor; (2) 9 cases were originally judged in favor of Owner and judgments were affirmed; (3) 10 cases were originally judged in favor of Owner and judgments were reversed; (4) 4 cases were originally judged in favor of Contractor and judgments were affirmed; and (5) 5 cases were originally judged in favor of Contractor and judgments were reversed. Out of the 32 non-appeal cases, 18 cases were judged for Owner and 14 cases were judged for Contractor.

For each of the collected precedent cases, a detailed analysis is performed to extract legal factors that are hypothesized to have led to the decisions on those cases. Within the legal domain, cases are judged after detailed and through analysis of surrounding circumstances. Consequently, judgments are based on concrete facts that are always stated within the body of each case. The factors related to this analysis are extracted from the stated opinions of judges. For example, in the case of All County Paving Corp., Doing Business as Collins Construction Co., Appellant, v Suffolk County Water Authority, Respondent, judges Anita R. Florio, J.P., Robert W. Schmidt, Thomas A. Adams, and William F. Mastro stated in their opinion “Indeed, the specifications stated that there was "no guarantee that unknown, adverse, conditions [did] not exist underground in the vicinity of the drill site." Thus, under the terms of the parties' contract, the plaintiff bore the risk of encountering unexpected subsurface soil conditions, and “since the defendant made no misrepresentations and withheld no information, the plaintiff was not entitled to extra compensation”. These facts are considered as the bases of including two factors namely SpecWarn (Whether the specifications warn against the possibility of DSC existence or not) and

MMistake (Whether the mistake was a mutual one and no ill intent was meant from any party). In that regard, a total of 53 factors were extracted. An analysis of the existence of each of these factors in all cases was then performed in the form of binary indicator variables [not existing (0) or existing (1)]. As a measure of choice, an indicator variable for the final judgment was recorded [owner (1) or contractor (0)]. A list of all extracted factors is provided in appendix A.

3.2.2 Binary Probit Model Implementation

This study is concerned with finding factors, out of the generated list, that are statistically significant for the prediction of construction litigation outcomes related to DSC claims. Since the analysis is pertinent to only two outcomes, Discrete Binary Models were implemented using the statistical modeling software LIMDEP (Greene 1998). The present stage of the statistical modeling approach implements a binary probit model. In statistics, a probit model is a popular specification of generalized linear models that was introduced by Chester Ittner in 1935. This stage implements probit regression, which is the application of probit models to the data set created in the previous stage of this statistical modeling approach. In this regression the likelihood of an outcome of a case (either in favor of the owner or contractor) follows a binary distribution. For illustration, Let *Y* be a binary outcome variable representing whether the owner prevail or not and having the value of 1 or 0 respectively. Also let *X* be a vector of regressors defining the legal factors in each case. The probit model developed will therefore be given by equation 3.1.

$$P(Y=1|X=x)=\emptyset(x'\beta) \qquad 3.1$$

where Φ is the cumulative distribution function of the standard normal distribution, *x* is a legal factors, *x'* is the standardized form of the legal factors, and β is a vectors of estimable parameters obtained from the regression. The Probit model is derived under the assumption that disturbance terms ε within the generated model are normally distributed. In this case the probability of owner prevailing (Y=1) occurring for case *n* is computed using equation 2 (Washington et al. 2003).

$$P_n(1)=P(\beta_1 X_{1n}-\beta_2 X_{2n}\geq\varepsilon_{2n}-\varepsilon_{1n}) \qquad 3.2$$

Where: β_1 and β_2 are vectors of estimable parameters for the owner or contractor prevailing respectively. X_1 and X_2 are vectors of legal factors that determine the outcome for case *n*. ε_{1n} and ε_{2n} are normally distributed disturbance terms with mean=0, variance σ^2_1 and σ^2_2 respectively, and covariance σ_{12}. Due to the normality assumption, $(\varepsilon_{1n} - \varepsilon_{2n})$ is normally distributed with mean=0 and variance = $\sigma^2_1 + \sigma^2_2 - \sigma_{12}$. It could be implied from above that the cumulative normal function for the probability of owner prevailing is given by equation 3 where $\sigma = (\sigma^2_1 + \sigma^2_2 - \sigma_{12})^{0.5}$ and the term $1/\sigma$ is a scaling of the function determining the case outcome (Washington et al. 2003)..

$$P_n(1)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{(\beta_1 X_{1n}-\beta_2 X_{2n})/\sigma} EXP\left[-\frac{1}{2}\omega^2\right] d\omega \qquad 3.3$$

In probit model the vector of estimable parameters *β* is readily estimated using standard Maximum Likelihood Estimation method (MLE). The principle of MLE is that different statistical populations generate different samples; any one sample is more likely to come from some populations rather than others. For example, if we have a sample of cases Y1, Y2, ... , Yn, the target is to find the value of β most likely

to generate this sample based on their legal factors X. Assuming that Yi is normally distributed with mean $\beta_0 + \beta_i x_i$ and variance σ^2, where β_0 and β_i are scalars representing the estimable parameter of Y intercept and each legal factor respectively. Therefore, the probability distribution can be written as (Washington et al. 2003):

$$P(Y_1)=\frac{1}{\sqrt{2\pi\sigma^2}}EXP\left[-\frac{1}{2\sigma^2}\left(Y_i-\beta_0-\sum\beta_i x_i\right)^2\right] \quad 3.4$$

Consequently, the likelihood function can be written as (Washington et al. 2003):

$$L\left(Y_1,Y_2,\ldots,Y_N,\beta_0,\beta_i,\sigma^2\right)=P(Y_1)P(Y_2)\ldots P(Y_N) \quad 3.5$$

$$=\prod_{i=1}^{N}\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)EXP\left[-\frac{1}{2\sigma^2}\left(Y_i-\beta_0-\sum\beta_i x_i\right)^2\right]$$

Where Π is the product of N factors. For simplicity, work is done with the algorithm form of *L*. this is statistically acceptable since *L* is always non-negative. Maximizing LN(L), LL with respect to β_0, β_1, and σ^2 results in:

$$\frac{\partial(LL)}{\partial\beta_0}=\frac{1}{\sigma^2}\sum\left(Y_i-\beta_0-\sum\beta_i x_i\right)=0 \quad 3.6$$

$$\frac{\partial(LL)}{\partial\beta_1}=\frac{1}{\sigma^2}\sum\left[x_i\left(Y_i-\beta_0-\sum\beta_i x_i\right)\right]=0 \quad 3.7$$

$$\frac{\partial(LL)}{\partial\sigma^2}=-\frac{1}{2\sigma^2}+\frac{1}{2\sigma^4}\sum\left(Y_i-\beta_0-\sum\beta_i x_i\right)=0 \quad 3.8$$

$$\beta_i=\frac{\sum(x_i-\bar{X})(Y_i-\bar{Y})}{\sum(x_i-\bar{X})^2} \quad 3.9$$

$$\beta_0=\bar{Y}-\sum\beta_i\bar{X} \quad 3.10$$

The basic functional form adopted for this analysis is the linear form. Due to the nature of the model specification and to legal factors being modeled using

indicator variable that have values 0 and 1, pseudo elasticity were observed. In this case, the sign of the estimated parameters relates the presence of a statistically significant factor and its increasing or decreasing effect (+ or – sign) on the probability of owner prevailing.

Table 3.1 illustrates the dependent and independent variables for a sample of five cases. Implementing the statistical modeling approach yielded β_{DSCC}=0.50, β_{DCS}=-0.667, $\beta_{N\&C}$=-0.50, $\beta_{Conraise}$=-0.50, $\beta_{ComImpossible}$=-1.00, $\beta_{Ochange}$=-1.00, $\beta_{Mmistake}$=-6.667, β_{Ocause}=0.00, $\beta_{SpecWarn}$=1.00, $\beta_{SpecRep}$=0.00, $\beta_{CNoExtra}$=-0.50, $\beta_{Ofalsely}$=0.50, $\beta_{OAdjust}$=0.50, β_0=2.50. A positive estimable parameter means that that the related factor increases the probability of the outcome, while a negative estimable parameter means that the related factor decreases the probability of that outcome; a large estimable parameter means that that the related factor strongly influences the probability of that outcome; while a near-zero estimable parameter means that that the related factor has little influence on the probability of that outcome. From the above example, a factor like SpecRep will have no effect on the outcome; whereas, Specwarn will have the highest effect on increasing the probability of the outcome. From the above equations, if all other estimable parameters are equal to zero, β_0 (also called y intercept) will represent the general trend of the outcome. As all other estimable parameters, the sign and value inferences are applicable to the interpretation of β_0. In the above example, there is a general trend for the outcome to occur. Since the target of this research is to find and determine the significance of the defined legal factor on the prediction of the winning party; the validation process is twofold. The first is the determination of the

best probit model through the measure of fit ρ^2 and over all model fit R^2. Second, significance of each factor is determined through its t-statistics (equation 11), which is a representation of any parameter to be significantly different than 0. At a confidence interval of 0.1, a t-statistics above 1.3 is considered significant (please refer to tables 3.2 and 3.3). The above described modeling steps are repeated iteratively till a model is found that best satisfies the aforementioned validation criteria (please refer to figure 3.1).

$$t_i = \frac{\beta_i - 0}{standard\ error(\beta_i)} \qquad 3.11$$

3.2.3 Binary Logistic Model Implementation

Non-linear modeling, Logistic Regression (LR), is another alternative for analyzing data of binary nature that is implemented in this stage to verify the significance of the legal factors affecting DSC disputes that were identified in the first stage of this statistical modeling approach (Tabachnick and Fidell, 1996). The LR model was derived similar to the probit model but under the assumption that disturbance terms ε within the generated model follow Gumbel distribution. The adopted form of the model is represented in equation 3.12.

Table 3.1 Sample Example of 5 Cases

DSSC	DSC	N&C	Conraise	ComImp	Ochange	Mmistake	Ocause	SpecWarn	SpecRep	CNoExtr	Ofalse	Oadj	Outcome
0	1	1	0	0	0	1	0	1	1	1	0	0	1
0	1	1	1	1	1	1	0	0	1	1	0	0	0
0	1	1	1	1	1	1	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0	1	1	0	0	1	1
1	0	0	1	0	0	0	0	1	1	1	1	0	1

Table 3.2 Relevant statistics of Probit Model at Confidence Interval = 0.1

Relevant statistics	Value
Dependent variable	OUT
Weighting variable	None
Number of observations	60
Log likelihood function	-7.733669
Number of parameters	12
Info. Criterion: AIC	0.743720
Finite Sample: AIC	0.858000
Info. Criterion: BIC	1.105050
Info. Criterion:HQIC	0.878420
Restricted log likelihood	-31.091550
McFadden Pseudo R-squared	0.751261
Chi squared	46.715760
Degrees of freedom	11
Prob[ChiSqd > value]	0.000000
Hosmer-Lemeshow chi-squared	1.623930
P-value (with deg.fr. = 1)	0.202540

$$f(Y)=\frac{1}{1+e^{-X}} \rightarrow y=\log\left(\frac{P_i}{1-P_i}\right)=\beta_0+\beta_i \cdot x_i \qquad 3.12$$

Where β_0 and β_i are estimable parameters for the Y intercept and legal factor i respectively and x_i is the value of the legal factor that determine the outcome for any individual case *i*. The variable y represents the exposure to some set of legal factors x_i, while *f*(Y) represents the probability of a particular outcome of a case, given that set of legal factors. The intercept is the value of β_0 when the value of all legal factors is zero. The individual value of each estimable parameter determines the significant effect of its corresponding legal factor on the probability of a particular outcome of a case. Similar to Probit, the estimable parameter values are estimated using MLE. The significance of each factor is determined through its t-statistics

(equation 3.11), which is an indicator of any parameter to be significantly different than 0. At a confidence interval of 0.1, a t- statistics above 1.3 is considered significant.

Table 3.3 Relevant statistics of Logistic Model at Confidence Interval = 0.1

Relevant statistics	Value
Dependent variable	OUT
Weighting variable	None
Number of observations	60
Log likelihood function	-12.04719
Number of parameters	10
Info. Criterion: AIC	0.93543
Finite Sample: AIC	1.04972
Info. Criterion: BIC	1.29676
Info. Criterion:HQIC	1.07013
Restricted log likelihood	-31.09155
McFadden Pseudo R-squared	0.6125253
Chi squared	38.08872
Degrees of freedom	9
Prob[ChiSqd > value]	0.7248771E-05
Hosmer-Lemeshow chi-squared	11.97723
P-value (with deg.fr. = 4)	0.01752

In this form of logistic regression equation 1.13, the owner prevailing outcome is the logarithm of the ratio of the probability of the *owner* prevailing (P_i) to the probability that this outcome does not occur ($1-P_i$). Taking the exponential of both sides of the above equation yields (Washington et al. 2003):

$$\left(\frac{P_i}{1-P_i}\right) = e^{\beta_0+\beta_i \cdot x_i} = e^{\beta_0} \cdot e^{\beta_i \cdot x_i} \qquad 3.13$$

It is clear from equation 3.13 that when a legal factor x_i increases by 1(i.e. exist in the case), with all other factors remaining unchanged, then the odds of an

outcome will increase by a factor e^{β_i}, known as the odds ratio (OR). The OR quantifies the relative change by which the odds of the outcome increase or decrease when the value of the predictor is increased by 1.

The application of this three stage statistical modeling approach yielded a number of very useful insights about the main legal factors that impact DSC disputes in the construction industry. The results and the insights obtained are detailed in the following section of the chapter.

3.3 Results and Discussion

The results of the application of the aforementioned statistical modeling approach are presented in tables 3.5, 3.6, and 3.7, respectively. The following is closer examination and discussion of these results that highlights: (1) the independent variable estimation; (2) the prediction models; and (3) the sensitivity analysis.

3.3.1 Independent Variables Estimation

The independent variables (legal factors) under investigation represent factual conditions upon which the entitlements of 60 DSC litigation cases in the construction industry were decided in the Federal Court of New York. These variables include “the presence of DSC clause”, “the presence of factual aspects illustrating the presence of Type 1 and\ or Type 2 DSC”, among others. While performing a study of the influence of each variable on the prediction of owner prevailing, it was noticed that 18 of the extracted variables were constant over all observations (cases). Although these factors were constant the outcomes of their

related cases were not. Consequently, it was obvious that they had no direct impact on case outcome. Statistical models built utilizing these variables lead to estimable parameters $\beta_i=0$. As a result, they were not included in the scope of this analysis. Eliminating these variables from the analysis yielded 35 variables to be tested. Furthermore, some of the variables were grouped and were represented by new indicator variables yielding a total of 23 variables for testing. Grouping of variables was based on their similarity. For example, three variables related to work stoppage (Stoppage of work due to the encountered matter, Stoppage of work due to the Owner, and Stoppage of work due to the Contractor) were grouped under one variable namely Wstop. If any of the newly developed indicator variables was proven to be statistically significant by the best developed models, a detailed analysis of their components was to be performed. The two developed models yielded consistent results with respect to the effect of the tested legal factors. The remaining factors after this process are shown in Table 3.4. In addition the Table illustrates whether the existence of each of these factors increase, or decrease the prediction of the model.

3.3.2 Prediction Models

One of the very promising findings of the developed statistical modeling approach is the prediction rates of the developed Probit and Logistic models that reached 88.9% and 93.3%, respectively. Results reported in Tables 3.5 and 3.6 illustrate a number of interesting insights about the legal factors and their outcomes. The following is a discussion of these findings:

1. It can be deduced from the coefficients of the constants in table 3.5 and 3.6 that generally, cases in which the Federal Government is a concerned party of the dispute, judgments are in favor of the government (owner) over contractor. This is expected due to the fact that N.Y. Const. art. 3, § 28 it is stated that the legislature shall not, nor shall the common council of any city, nor any board of supervisors, grant any extra compensation to any public officer, servant, agent or contractor (Ralph S. Keep 1930).
2. Further examination of the developed models demonstrate consistency with regards to "the presence of evident facts that the encountered conditions caused a change in the nature and cost of the contract" to decrease the prediction of an owner winning a case, reference is made to the coefficients of the N&C parameter in table 3.5 and 3.6. During bidding, contractors specify their prices based on decisions concerning methods and resources needed for performing the works. A change in works causing a variation in the nature of these methods may have a great impact on increasing the contractors' costs. Consequently, it is not fair to burden contractors with that increase in cost, leading to a decrease in planed profit or even loss, without equitably adjusting them. Supporting this notion, Judge Goldman, J. W. states that where an operation is not within the original plans and the contractor is forced to use a more expensive operation to perform the work than was originally anticipated and contemplated by the contract, the claimant shall be compensated for this extra work. However, there are two scenarios that should be discussed in this case. The first, if an owner

compensates a contractor for his\ her direct or direct and indirect costs, reference is made to the coefficient of the OADJUST parameter in table 3.6. In this case, owner has rectified a mistake on his side. As a result, the developed models predicted that the presence of evidence of this nature in a case as a factor that increases the prediction in favor of owner. The second is when the contract included a clause giving the owner the right to make changes to the project until final completion and acceptance without invalidating the contract provided that it was made due to a necessity; reference is made to the coefficients of the SPECWARN parameter in tables 3.5 and 3.6 (Tony 1919). The presence of similar clause in a contract was interpreted by both models to increase the prediction in favor of the owner.

3. It was also found from the coefficients of the COMIMPOS parameter in tables 3.5 and 3.6 that the prediction of owner winning a construction litigation case concerning DSC is decreased due to the presence of evident facts that the encountered matter rendered the project impossible to be completed. For example, if the DSC experienced in a project required a redesign that caused the elimination of a major part of the contractor's scope of work, which intern affects his method of pricing and profit allocation to the extent that he/she cannot perform the works as specified, he\ she is entitled to be compensated for that loss (Kinser 1912). In addition, the contractor raising the faced incident as per the contract documents and in due time was found to decrease the prediction of owner winning a DSC related litigation case, reference is made to the coefficients of the CRAISE parameter in tables 3.5

and 3.6. Construction Contracts place a responsibility on the contractor to inform the owner with any unexpected matters encountered in the project lifetime. This responsibility allows the owner and contractor enough time to analyze the situation and decide on counter measures. Consequently, if a contractor fulfilled his/her contract requirement, he\she will have a better chance proving his case.

Table 3.4 Significance of Individually Tested Variables

Factor Symbol	Factor Meaning	Influence on Prediction
TYPEP	Type of project: the higher is the sophistication of the construction project the higher is the variable	Increase
DSC	The presence of factual facts demonstrating Type 1 or Type 2 DSC	Decrease
WSTOP	Stoppage of work due to the encountered matter, Owner, or Contractor	Not significant
DSCC	The presence of DSC Clause in the Contract	Decrease
REDESIGN	Whether the encountered matter required redesign	Not significant
N&C	Whether the encountered matter imposed changes on the nature and costs of the Contract or not	Decrease
CRAISE	Whether the contractor raised his claim as per the contract clauses or not	Decrease
COMIMPOS	Whether the encountered matter made the project completion impossible or not	Decrease
OCHANGE	Whether the contract clauses allow the owner to perform changes at any time of the project duration without the consent of the contractor or not	Increase
CNPROFIT	Whether the contractor under the conditions of the contract waived his right for profit due to changes or extras or not	Increase
MMISTAKE	Whether the mistake was a mutual one and no bad intentions was meant from any party or not	Increase
VCHANGES	Whether various changes were implemented through the life time of the project or not	Not significant

Table 3.4 (Continued)

Factor Symbol	Factor Symbol	Factor Symbol
OCAUSE	Whether the incurred damages were caused due to the owners negligence or any of his representatives or not	Decrease
SPECWARN	Whether the specifications warn against the possibility of DSC existence or not	Increase
SPECREPR	Whether the specifications had a representation of the actual site conditions or not	Decrease
CNEXTRA	Whether the contractor under the conditions of the contract waived his right for compensation due to extras or not	Increase
OFALSELY	Whether The Owner\ Owner Rep. falsely state that the matter encountered in hand, so far as known, was shown in the Contract documents?	Decrease
LUMPUNIT	Whether the contract is a unit price or not	Not significant
OADJUST	Whether the owner equitable adjusted the contractor against extra works performed or not	Decrease
BENEFIT	Whether the contractor benefits from the work done or not	Not significant
NOTIME	The presence of enough evidence demonstrating that there was no time for the Contractor to perform his own investigations	Decrease
WTEMP	Whether the extra works were performed as temporary works or not	Increase
WAPPEAL	In case of appeals, in favor of whom did the court originally rule	Not significant

4. Furthermore, both developed models pointed out that the presence of evident facts that there was a mutual mistake from both sides in examining the site and contract documents increases the prediction of judgment in favor of owner, reference is made to the coefficients of the MMISTAKE parameter in tables 3.5 and 3.6. In this case, there is no bad faith, concealment or misrepresentation on the side of the owner; therefore, no responsibility for the loss resulting from a difference between estimated quantities of material affecting work conditions and those actually found at a job site (Drake 1965). On the other hand, it can be inferred from the coefficients of the SPECREPE

parameter in tables 3.5 and 3.6 that the presence of evident facts that the contract documents included accurate representation of the site conditions decreases the prediction in favor of the owner. In this case, the existence of DSC depends upon a comparison of the site conditions actually encountered with the affirmative representations of conditions contained in the bid and contract documents. To the extent that the conditions described in the contract materialize, the contractor bears the risk, while the owner assumes the risk for conditions that the contract documents fail to disclose.

Table 3.5 Probit Model Results at a Confidence Interval = 0.1

Independent variable	Coeff.	t-stat.	Elasticity	% Change in Prediction
Constant	4.83	1.80		
DSCC	-0.33	-1.31	-0.37	0.00
N&C	-2.69	-2.02	-0.68	-17.77
CRAISE	-2.10	-1.52	-0.66	-11.11
COMIMPOS	-1.17	-1.17	-0.47	-11.11
OCHANGE	4.04	1.75	0.22	17.78
MMISTAKE	2.06	1.53	0.67	17.78
OCAUSE	-1.14	-1.47	-0.22	-11.11
SPECWARN	2.46	1.80	0.49	55.56
SPECREPR	-3.07	-1.50	-0.80	0.00
CNEXTRA	1.38	1.41	0.40	0.00
OFALSELY	-1.04	-1.55	-0.12	0.00

5. Additionally, the presence of evident facts that the specifications included a warning against the presence of DSC from those conveyed in the contract documents increases the prediction of judgment in favor of owner, reference is made to the coefficients of the SPECWARN parameter in tables 3.5 and

3.6. If the specifications stated that there was no guarantee that adverse conditions did not exist underground at the construction site, the contractor is required to familiarize himself\ herself with the site conditions. In this case, the contractor is held responsible for damages that he\ she might incur due to encountering DSC (All County Paving 2005).

Table 3.6 Logistic Model Results at a Confidence Interval = 0.1

Independent variable	Coeff.	t-stat.	OR
Constant	2.36	0.25	
TYPEP	4.23	1.66	32.314
N&C	-5.69	-2.00	0.035
CRAISE	-7.54	-1.86	0.053
COMIMPOS	-3.04	-1.46	0.047
OCHANGE	9.65	2.13	15.604
MMISTAKE	2.57	1.05	13.732
SPECWARN	3.80	2.19	44.740
SPECREPR	-5.11	-1.58	0.061
OADJUST	5.67	1.84	2.353

6. As can be deduced from the coefficient of the DSCC parameter in table 3.5, the Probit model pointed out the presence of a DSC clause in the contract as a crucial factor that decreases prediction in favor of owner. As mentioned earlier, once construction begins on a project under a contract that is silent about the risk of unforeseen conditions, a contractor bears the risk of running into conditions that were unforeseen at the time he\ she submitted his\ her bid even though they significantly increase the cost of performance (Iacobelli 1994). As a result, the presence of a DSC clause in a contract allows the

contractor to reimburse additional incurred costs due to DSC. A contractor with legal right to extra compensation would always have a better chance winning a case in the presence of such a clause. Furthermore, the model highlighted that the prediction of owner winning a construction litigation case concerning DSC is decreased due to the presence of evident facts that the damage incurred by the contractor was due to negligence on the side of the owner, reference is made to the coefficient of the OCAUSE parameter in table 3.5. For more illustration, if an owner, by its own act, causes the work to be done by a contractor to be more expensive than it otherwise would have been according to the terms of the original contract, it is liable to him\her for the increased cost or extra work (William 1899). Similarly, the model indicated that the prediction is also decreased by the presence of evident facts that the Owner or their representative falsely state that the matter encountered in hand, so far as known, was shown in the contract documents, reference is made to the coefficient of the OFALSELY parameter in table 3.5. In Faber (1918), the contractor recovered damages that he has incurred due to DSC on the grounds that there was an express warranty from the project engineer that the contract documents constitute an accurate representation of the site sub-surface conditions. However, the model predicted that prediction in favor of the owner increases if the contractor agreed: (1) to waive his right for any extra compensation; and (2) that all work shall be solely at the his risk until it has been finally inspected and accepted by the owner (Kinser 1912).

7. In comparison, the Logistic model demonstrated that as the complexity of public projects increases, the prediction in favor of owner is increased, reference is made to the coefficient of the TYPEP parameter in table 3.6. Projects under analysis vary between Excavation projects, Sanitary projects, and Water related projects like Dams. The nature in which this variable was integrated in the model is prioritized with Excavation projects being the lowest in complexity to Water related projects being the most complex. Costs associated with performing a construction project is directly related to its level of complexity and size.

Table 3.7 Analysis of Binary Choice Models Prediction (Threshold = 0.5)

Prediction Success	**Probit**	**Logit**
Sensitivity = actual 1s correctly predicted	85.714%	90.476%
Specificity = actual 0s correctly predicted	91.667%	95.833%
Positive predictive value = predicted 1s that were actual 1s Negative predictive value = predicted 0s that were actual 0s	90.000%	95.000%
Negative predictive value = predicted 0s that were actual 0s	88.000%	92.000%
Correct prediction = actual 1s and 0s correctly predicted	88.889%	93.333%
Prediction Failure		
False pos. for true neg. = actual 0s predicted as 1s	8.333%	4.167%
False neg. for true pos. = actual 1s predicted as 0s	14.286%	9.524%
False pos. for predicted pos. = predicted 1s actual 0s	10.000%	5.000%
False neg. for predicted neg. = predicted 0s actual 1s	12.000%	8.000%
False predictions = actual 1s and 0s incorrectly predicted	11.111%	6.667%

3.3.3 Sensitivity Analysis

The obtained results shown in Figure 3.2 illustrate the outcomes of the sensitivity analysis performed on the developed models. The sensitivity analysis is used to determine how different values of an independent variable (significant legal concepts) will impact a particular dependent variable (owner winning a case) under a given set of assumptions. This analysis is very useful when attempting to determine the impact the actual outcome of a particular variable will have if it differs from what was previously assumed. To that end, the sensitivity of each variable is tested by increasing the variable by 1 while maintaining the rest fixed at their mean value. The outcomes of the analysis were consistent between both developed models and demonstrated the following.

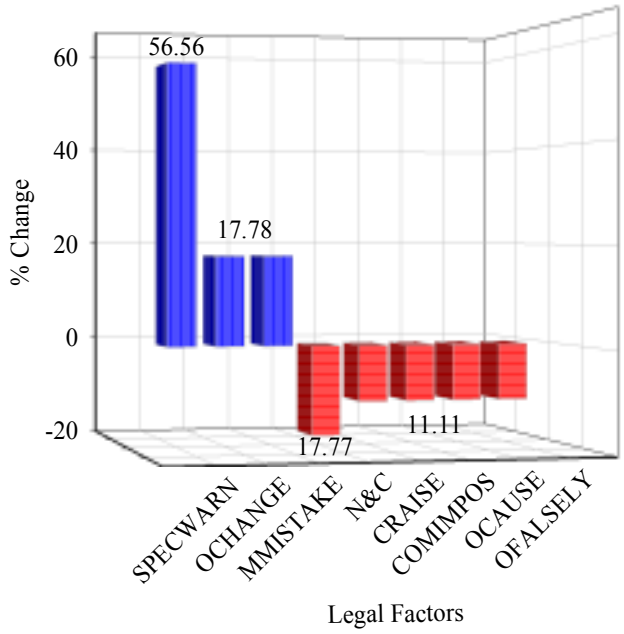


Figure 3.2 Outcomes of Sensitivity Analysis

1. The presence of evident facts that the encountered conditions caused a change in the nature and cost of the contract had the highest impact among variables causing a decrease in the prediction of judgment in favor of owner. The Probit model indicated that it caused an increase in prediction in favor of contractor from 55.56% under the base case to 73.33% under imposed scenario (refer to figure 3.3). Consistent with that finding, the Logistic model indicated that an increase of 1 results in reducing the odds of an owner winning approximately by a factor of 29 (OR=0.035). Reference is made to the percentage change in prediction and OR values of the N&C parameter in table 3.5 and 3.6.
2. The presence of evident facts that the specifications included a warning against the presence of DSC from those conveyed in the contract documents had the highest increases in the prediction of judgment in favor of owner (refer to figure 3.4). It caused an increase in prediction on favor of owner from 44.44% under base case to 100.00% under imposed scenario. Consistent with that finding, the Logistic model indicated that an increase of 1 results in increasing the odds of an owner wining approximately by a factor of 45. Reference is made to the percentage change in prediction and OR values of the SPECWARN parameter in table 3.5 and 3.6.
3. The presence of a clause in a contract giving the owner the right to make changes to the project until final completion and acceptance without invalidating the contract provided that it was made due to a necessity (reference is made to the percentage change in prediction and OR values of

the OCHANGE parameter in table 3.5 and 3.6), and the presence of evident facts that the mistake was a mutual one and no bad faith was intended by any party (reference is made to the percentage change in prediction and OR values of the MMISTAKE parameter in table 3.5 and 3.6) caused the lowest increase on the prediction of judgment in favor of owner. It caused an increase in prediction on favor of owner from 44.44% under base case to 62.22% under imposed scenario. Consistent with that finding, the Logistic model indicated that an increase of 1 results in increasing the odds of an owner wining approximately by factors of 16 and 14 respectively.

4. The presence of evident facts that the contractor raised his claim as per the contract clauses, and that the encountered matter rendered the project completion impossible caused the lowest decrease on the prediction of judgment in favor of owner. Reference is made to the percentage change in prediction and OR values of the CRAISE and COMIMPOS parameter respectively in table 3.5 and 3.6. It caused an increase in prediction in favor of the contractor from 55.56% under base case to 66.67% under imposed scenario. Consistent with that finding, the Logistic model indicated that an increase of 1 results in reducing the odds of an owner wining approximately by factors of 19 (OR=0.053) and 21 (OR=0.047) respectively
5. The developed Probit model predicted that the presence of evident facts that the damage incurred by the contractor was due to negligence on the side of the owner had an impact of decreasing the prediction of judgment in favor of owner by 11.11. Reference is made to the percentage change in prediction

values of the OCAUSE parameter in table 3.5. They caused an increase in prediction on favor of contractor from 55.56% under base case to 66.67% under imposed scenario. However, increasing the following factors by 1 unit had no effect on the odds of prediction: (1) the presence of a DSC clause in a contract; (2) the presence of evident facts that the specifications had a representation of the actual site conditions; (3) the presence of evident facts that the Owner\Owner Rep. falsely state that the matter encountered in hand, so far as known; and (4) whether the contractor under the conditions of the contract waived his right for compensation due to extras or not. Reference is made to the percentage change in prediction values of the DSCC, SPECREPR, OADJUST, OFALSELY, and CNEXTRA parameter respectively in table 3.5.

6. From the OR value of the TYPEP parameter in table 3.6 it can be deduced that the developed Logistic model predicted that increasing the complexity of the project by 1 unit results in increasing the odds of an owner winning approximately by a factor of 32.

3.4 Summary and Conclusion

This chapter provides an initial step in this research methodology that attempts to create a construction legal decision support system through statistical analysis and machine learning techniques. Consequently, the aim of this chapter was to statistically analyze the significant legal factors that govern litigation outcomes in DSC dispute. The chapter, therefore, implemented three main stage that: (1)

collected significant number of DSC cases and extracted the legal factors on which they were judged; and (2) the main findings from the implementation of this three stage statistical modeling approach include:

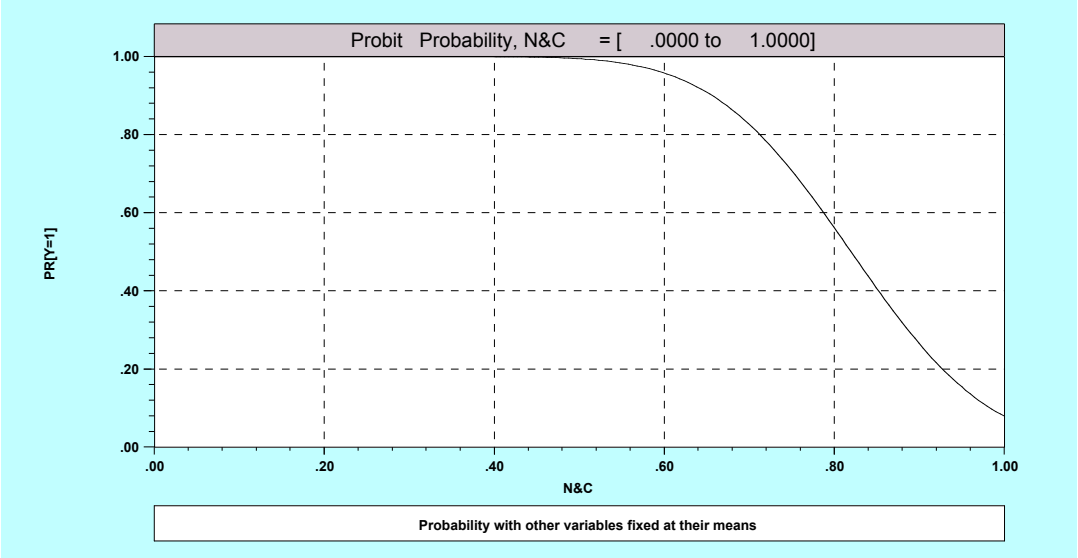


Figure 3.3 N&C Variation V. Prediction of Outcome 1 Occurring

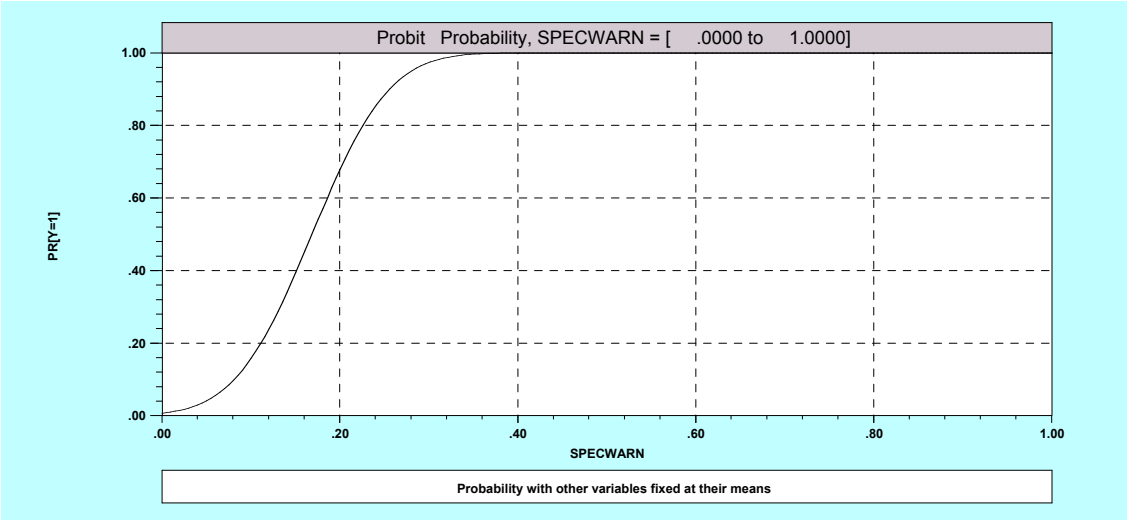


Figure 3.4 SPECWARN Variation V. Prediction of Outcome 1 Occurring

1. Generally, cases in which the Federal Government is a concerned party of the dispute, judgments are in favor of the government (owner) over contractor.
2. “The presence of DSC Clause in the Contract”, “Whether the encountered matter imposed changes on the nature and costs of the Contract or not”, “Whether the encountered matter made the project completion impossible or not”, “Whether The Owner\ Owner Rep. falsely state that the matter encountered in hand, so far as known, was shown in the Contract documents”, and “Whether the incurred damages were caused due to the owners negligence or any of his representatives or not” are factors that increase the probability of judgment in favor of contractors.
3. “Whether the contract clauses allow the owner to perform changes at any time of the project duration without the consent of the contractor or not”, “Whether the owner equitable adjusted the contractor against extra works performed or not”, and “Whether the specifications warn against the possibility of DSC existence or not” are factors that increase the probability of judgment in favor of owner.
4. “The presence of evident facts that the encountered conditions caused a change in the nature and cost of the contract” had the highest impact among variables causing a decrease in the prediction of judgment in favor of owner. It caused an increase of 17.77% in prediction on favor of contractor.
5. “The presence of evident facts that the specifications included a warning against the presence of DSC from those conveyed in the contract

documents" caused the highest increases in the prediction of judgment in favor of owner. It caused an increase of 56.56% in prediction on favor of owner.

These findings provide very useful insight on this important type of construction disputes. In case of a DSC dispute, an owner and/or a contractor can assess the strength of their situation based on the identified factors if resolving through litigation is decided. This assessment would allow disputing parties to take a more assured decision about other resolution mechanism like amicable settlement, mitigation, and/or arbitration. Furthermore, some of the identified factors are related the wording of contracts and technical specifications in the construction industry. Therefore, the current research provides knowledge to contractors about factors to which emphasis should be given while bidding for new projects and upon which control should be maintained while performing a project. The developed models, however, do not take into consideration precedent cases cited within the body of each case. Because rules alone are insufficient, judges employ analogical reasoning with precedent cases in their decision-making process (Ashley and Rissland 1988). Precedence, or the reliance of a court on the decisions of previous relevant cases, is an important aspect of the Anglo-Saxon legal system (Elhadi 2001), the dominant legal system in the judicial system of the United States. These enhancements and others are, therefore, incorporated under the research tasks illustrated in the following chapters of this dissertation.

CHAPTER 4

DSC LITIGATION PREDICTION MODEL DEVELOPMENT FOR THE CONSTRUCTION INDUSTRY

4.1 Introduction

One of the fields of AI that is becomingly a topical issue in computing research is Machine Learning (ML). ML is that field of AI that deals with developing tools and algorithms allowing a computer to build up knowledge about problems and applying it to solve newly encountered ones of similar nature (Shawe-Taylor and Cristianini 2000). As illustrated earlier in chapter 2, ML algorithms address complex problems that do not lend themselves to solution using traditional computing techniques. In the present chapter a number of ML algorithms are used for building models that provide decision support capabilities in DSC disputes. As illustrated in chapter 2, capturing all legal rules as well as human thinking and perception of facts has proven to be a very complex undertaking. Researchers in the field of litigation decision support and natural language processing (NLP) demonstrated that developing a model to mimic the cognitive ability of the human mind, resembling its ability to acquire knowledge by the use of reasoning, intuition, and perception, is impossible with the current state of science (Cobb and Diekmann 1986). However, the required human knowledge about solving a problem exists implicitly in precedent cases of similar nature (Arditi and Pulket 2005). As a result, the problem is simplified to a matter of extracting the knowledge rather than building it cognitively. Consequently, ML tools and algorithms devised new strategies that attempt to solve

problems by utilizing the computers' ability to extract knowledge from tagged input\output data sets (Nilsson 2008). These tools have been extensively utilized in building CBR systems for the construction industry as discussed in chapter 2. In general, two types of learning are widely applied: inductive, and deductive. Inductive machine learning methods extract rules, patterns, and information automatically out of massive data sets by computational and statistical methods in an attempt to attain the required computer knowledge (Jurafsky and Martin 2000).

The main objective of this chapter is to develop ML model for construction legal decision support in DSC disputes. In order to achieve this goal, the performance of different ML tools will be evaluated, including: (1) Support Vector Machine (SVM) algorithms; (2) Naïve Bayes (NB) algorithms; and (3) Rule Induction Learning. The aforementioned algorithms will be evaluated using the significant legal factors identified in chapter 3. The evaluation process utilizes 120 DSC cases from The Federal Court of New York that were filled in the period between 1912 and 2007. The research approach adopted for the current stage includes (Figure 4.1): (1) data preparation; (2) ML model development and analysis; and (3) ML model implementation.

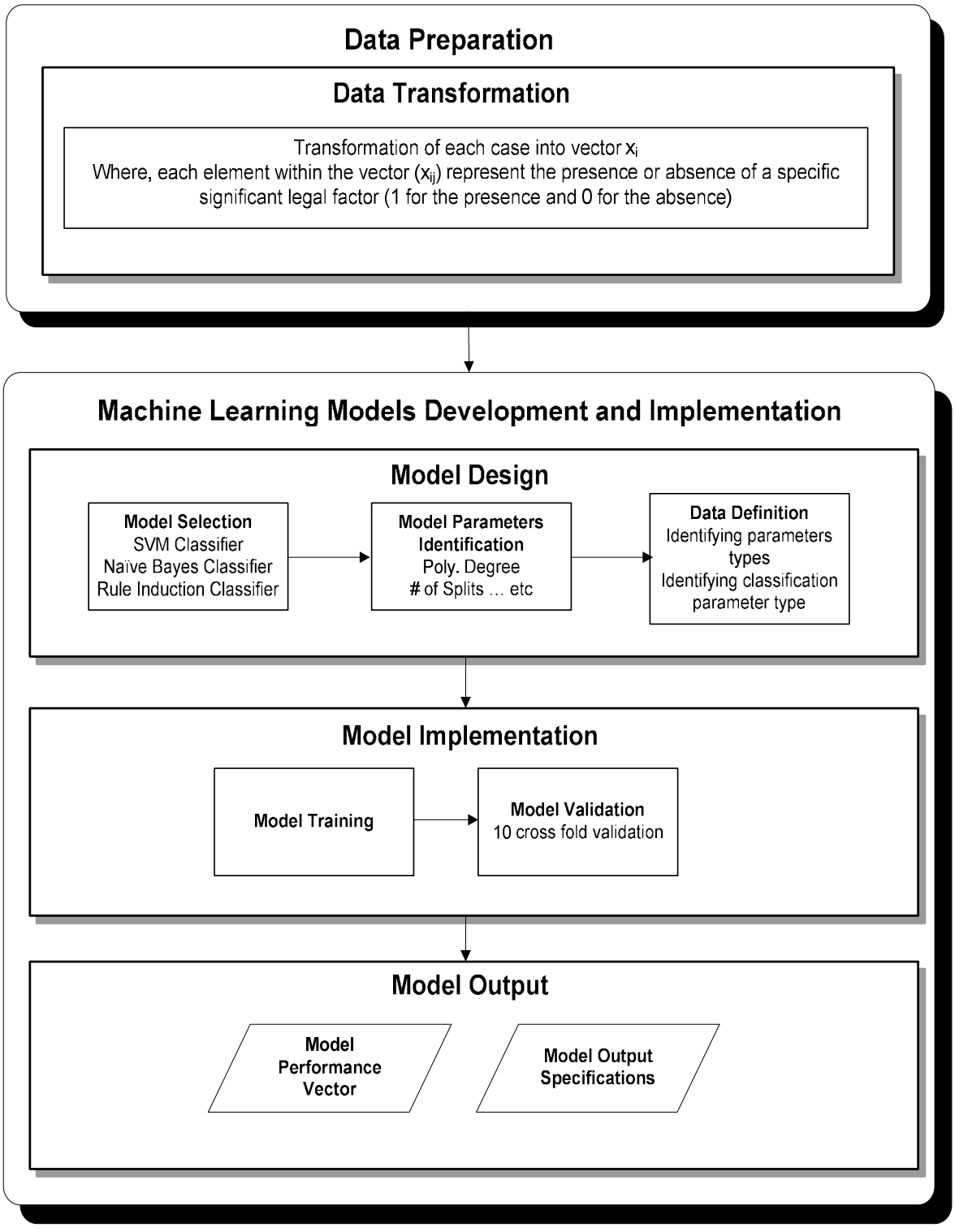


Figure 4.1 Research Approach

4.2 Data Preparation

As mentioned earlier, the work under this chapter represents a continuation for chapter 3. Consequently, the identified significant legal factors namely Ptype, DSCC, DSC, N&C, Conraise, ComImpossible, Ochange, Mmistake, Year, Ocause, SpecWarn, SpecRep, CNoExtra, Ofalsely, and OAdjust are adopted as the learning parameters for the models to be developed. The input data for the models are developed in the form of vectors in which each case (instance) has a designated input vector x_i and each element within the vector (x_{ij}) represent the presence or absence of a specific significant legal factor (1 for the presence and 0 for the absence). However, two variables do not follow this representation namely the type of the project (Ptype) and the year of filing the case with the Federal Court of New York (Year). The analysis of the former is based on the complexity of the project and falls into one of four categories listed as follows based on the complexity assumption. Water related works are given a value of 4 and assumed to be the most complex type of projects due to the high uncertainty and difficulty in predicting site conditions. This category includes projects like dams and river stream maintenance and protection projects. The following category includes land related works like roads or traditional excavation works and is given a value of 3. Sanitary works are assumed to be less complex and are given a value of 2. All other types of works like housing projects are given a value of 1. As for the representation of the year, cases are categorized based on 5 years intervals as shown in the table 4.1 below. In addition, as a measure of choice, an indicator variable for the final judgment was

recorded [owner (1) or contractor (0)] and was inputted as an element in the case vectors.

Table 4.1 Representation of the Year Factor

Year of Deciding a case	Factor
Year>2002	0
1997<Year>2002	1
1992<Year>1997	2
1987<Year>1992	3
1982<Year>1987	4
1977<Year>1982	5
1972<Year>1977	6
1967<Year>1972	7
1962<Year>1967	8
1957<Year>1962	9
1952<Year>1957	10
1947<Year>1952	11
1942<Year>1947	12
1937<Year>1942	13
1932<Year>1937	14
1927<Year>1932	15
1922<Year>1927	16
1917<Year>1922	17
1912<Year>1917	18
Year ≤1912	19

4.3 ML Model Development and Analysis

The objective of this chapter is to develop a construction legal decision support model for DSC disputes based on statistically significant legal factors predefined in chapter 3. To this end, the present stage will aim at developing: (1) Kernel SVM Models; (2) Naïve Bayes (NB) models; and (3) Induction Learning Models including Decision Tree (DT), Boosted decision Trees (BDT), and PART models for DSC litigation outcome prediction in the construction industry.

4.3.1 Support Vector Machines (SVM)

In the current task, the SVM Classification algorithm aims at separating the 120 training cases into two classes (Owner and Contractor) based on the 15 statistically significant legal factors identified in chapter 3. In its simplest linear form, a support vector machine finds a hyperplane that separates a set of positive examples (cases judges in favor of Owner) from the set of negative examples (cases judges in favor of Contractor) with maximum margin as shown in figure 4.2. Binary classification is performed by using a real-valued hypothesis function, equation 4.1, where input x (case) is assigned to the positive class (Owner) if $f(x) \geq 0$; otherwise, it is assigned to the negative class (Contractor).

$$y = \langle \mathbf{w}.\mathbf{x} \rangle + b \qquad 4.1$$

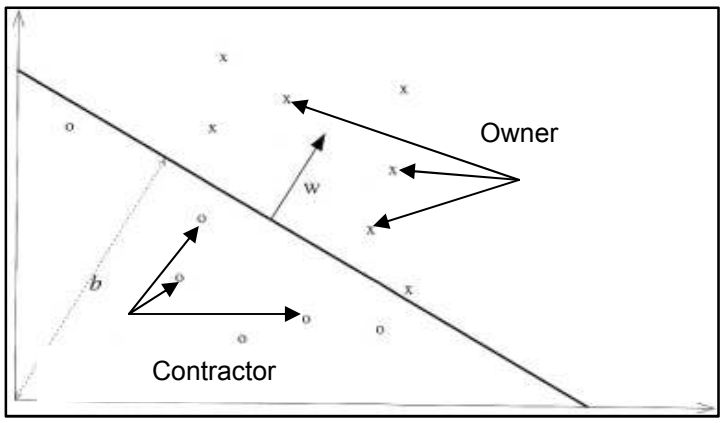


Figure 4.2 SVM Classification

As illustrated in chapter 2, kernel mapping is a widely used transformation method for solving nonlinear classification problems. Many kernel mapping functions can be used – probably an infinite number (DTREG 2008). But a few kernel functions have been found to work well for a wide variety of applications. The default

and recommended kernel function is the Radial Basis Function (RBF) and Polynomial Kernel (POLY) (Aiolli and Sperduti 2005, and DTREG 2008). Consequently, the work performed under this task investigates the use of Kernel SVMs (RBF and Polynomial) for developing a DSC litigation outcome prediction model.

4.3.2 Naïve Bayes Classifiers (NB)

In addition to the above described SVM models, this research task is concerned with finding the best outcome prediction model for construction cases related to DSC disputes utilizing Naïve Bayes Classifiers. Since the analysis is pertinent to only two outcomes, and due to the presence of high support in the ML domain in favor of the performance of Naïve Bayes (Bramer 2007, Manning and Schutze 2003) it was adopted for the current analysis.

Naïve Bayes is a type of classifier that does not implement rules to derive the classification, unlike rule induction classifiers that will be discussed later. The classification methodology adopted by NB Classifiers is based on the probability theory. It finds the most likely possible classification for an instance among all available classes taking into consideration the presence of prior knowledge of other pieces of information. Pertinent to the current research, NB classifier is build to estimate the probability of each class (Owner and Contractor) given the training set of 120 cases and prior knowledge of the existence of the significant legal factors. The classifier is trained based on conditional and prior probabilities of the existing set. A conditional probability as given in equation 4.2 is read as the probability of

case with legal factor values of (a) happening with the prior knowledge of a classification falling in class (x), Owner or Contractor. However, a prior probability means the probability of a certain class (x), Owner or Contractor happening based on the 120 cases recorded.

$$P(\text{case legal factors}=a|\text{class}=x) \quad 4.2$$

Since the 120 cases C_1, C_2, …, C_{120} are conditionally independent, the probability of an outcome of a newly un-encountered case is calculated based on equation 4.3.

$$P(\text{class}=\mathbf{x}|C=c_k)=\prod_i^{120} P(x_i|C=c_k) \quad 4.3$$

4.3.3 Rule Induction Classifiers

This sub-section is concerned with finding the best outcome prediction model for construction DSC cases utilizing Rule Induction Classifiers. DT, BDT, and PART are types of ML classifiers that adopt decision rules automatically generated from training examples or data sets to classify a newly unseen instance (Bramer 2007). DT classifier is a special case in which the generated decision rules are fitted into a form of a tree, where each leaf represents a decision state. For the 120 cases, decision rules were derived based on binary decision at each node and not class probability (Witten and Frank 2000). The models were developed with a splitting mechanism of a minimum of 2 instances per leaf and a confidence threshold of 0.25. Weka algorithm J48, ADTree, and PART were utilized for developing the DT, BDT, and PART models respectively.

4.4 Model Testing and Validation

Testing and validation of the developed models was performed using RapidMiner (formerly known as Yale) version 4.1 (Rapid-I 2008). Validation of the best developed model was based on prediction accuracy, precision, recall, F-measures, and the relation between true positive and false positive predictions illustrated by a value known as Area Under Curve (AUC). Outputs of the developed models were compared to a base line prediction of 50%. Model accuracy is defined as the proportion of the total number of correctly predicted cases to the total number of tested cases. Model precision is defined as a measure of the proportion of selected cases that the developed model predicted correctly out of the total set of cases the model referred to that class of prediction, whether true of false (equation 4.4). Model Recall is defined as the proportion of the cases pertinent to a specific class of prediction that the proposed model selected right (equation 4.5). It should be noted that there is always a tradeoff between precision and recall. For more illustration, a full set of cases could be selected attaining a 100% recall but with a very low precision. Consequently, an overall performance combining precision and recall can be reported by F-measure (equation 4.6).

$$\text{Precision} = \frac{tp}{tp+fp} \qquad 4.4$$

$$\text{Recall} = \frac{tp}{tp+fn} \qquad 4.5$$

$$\text{F- Measure} = \frac{2PR}{(R+P)} \qquad 4.6$$

Where t_p is the true positive prediction of the model, f_p is the false positive prediction of the model, f_n is the false negative prediction of the model, P is the precision of the model, and R is the recall of the model.

The testing and validation of the model is performed on a 10 fold scheme. The theory behind this training method also known as *cross-validation* was pioneered by Seymour Geisser (Shawe-Taylor and Cristianini 2000; Rapid-I 2008). This training methodology is a statistical practice of partitioning a sample of data into subsets such that the analysis is initially performed on a single subset, while the other subsets are retained for subsequent use in confirming and validating the initial analysis. Consequently, within the tested data set, the developed model is trained in a rotational manner. In each rotation, the model is trained over 90% of the cases and tested over the remaining 10%. This process is repeated till the model is trained and tested over all cases. Performance measures are reported for each developed model after the cross-validation stage is finished.

4.5 ML Model Implementation

The following is a description of the performance of the algorithm implemented for ML model development. The algorithm starts with identification of the model parameters (i. e. the degree of the SVM model or the number of splits of DT model). The algorithm iterates through the training data separating it into folds based on the cross validation mechanism (i.e. 10, 20, or 100 cross fold validation). The algorithm is trained over 90% of the data and tested over the other 10%. The algorithm performance vector parameter, accuracy, precision, and recall for each

fold are reported. The algorithm iterates in the above manner until it is trained and tested over the entire training set. The performance vector parameters, accuracy, precision, recall, and AUC averages over all folds are reported before the algorithm terminates.

RapidMiner (formerly known as Yale) version 4.1, developed by Rapid-I, was utilized for the implementation of ML models described in this chapter. RapidMiner is an environment for machine learning and data mining processes that has already been applied for ML and knowledge discovery tasks in a various domains like feature generation and selection (Klinkenberg 2002, Ritthoff et al. 2003, and Ritthoff et al. 2002), concept drift handling (Klinkenberg, 2004, Klinkenberg 2003, Klinkenberg and Rűping 2003, and Klinkenberg and Joachims 2000), transduction (Daniel et al. 2002, Klinkenberg 2001), pre-processing of and learning from time series (Mierswa and Morik 2005(a), Mierswa and Morik 2005(b), and Mierswa 2004), meta learning (Mierswa and Wurst 2005(a), and Mierswa and Wurst 2005(b)), clustering, and text processing and classification.

The research approach for the present task developed 10 ML models that related the likelihood of a DSC case being judged in favor of one party over the other to the identified set of legal factors and provided predictions for newly introduced cases. First, due to the presence of high support in favor of the performance of Polynomial and Radial Base Function (RBF) Kernel SVM (Aiolli and Sperduti 2005), 4 ML models namely Polynomial 1st degree, Polynomial 2nd degree, polynomial 3rd degree, and RBF Kernel models were developed. Second, the proposed research approach developed and compared the outputs of 2 NB models while implementing

and not implementing kernel estimators as a parameter of the model. Third, 4 rule induction models namely DT, PART, BDT with 10 Boosts, and BDT with 15 Boosts models were develop. It worth noting at this point that the boost number was increased to 20 and 25; however, no enhancement in the performance of the model was achieved.

4.6 Results

This section presents the testing and validation results for the 10 ML models developed in the previous section of this chapter. The section will present, for each type of ML algorithm the best model obtained.

4.6.1 Support Vector Machines (SVM)

The results of the testing and validation of the developed SVM algorithms are presented in tables 4.2, figures 4.3, 4.4, 4.5, 5.6, and appendix B respectively. The following is closer examination and discussion of these results. As can be noted from table 4.2, the 2nd and 3rd degree Polynomial Kernel SVM models achieved the highest performance measures while 1st degree Polynomial Kernel SVM achieved the lowest. The overall accuracy of the Polynomial degree 1, 2, and 3, and RBF models were 94%, 98%, 98%, and 96% respectively.

The observed superiority of the 2nd and 3rd polynomial models extended to cover all validation criteria. A closer look into the achieved measures illustrates a slighter higher performance of the 3rd degree polynomial kernel model over the 2nd degree one. It can be seen from table 4.2; the statistical properties (namely the Mean absolute Error, Root mean squared error, Relative absolute error, and Root

relative squared error) of the 3rd degree model are slightly less than those of the 2nd degree one giving it the upper hand when deciding on the best model. Further examination of figure 4.8 (AUC), which defines the relation between true and false positive predictions, further highlights the superiority of the 3rd degree Polynomial model. A true positive prediction of 99.6% can be made with 0% false positive predictions. In other words, the model achieves right classification (assigning a case to its right class) at a rate of 99.6% without making mistakes.

Table 4.2 Results of Kernel SVM Implementation

Property	Polynomial Degree			RBF
	1	2	3	
Accuracy	94.00%	98.00%	98.00%	96.00%
Precision	93.83%	98.00%	98.00%	86.48%
Recall	93.50%	98.00%	98.00%	93.00%
F-Measure	93.66%	98.00%	98.00%	89.62%
AUC	95.40%	99.60%	99.60%	94.30%
Contractor's class precision	97.73%	97.83%	97.83%	86.00%
Contractor's class recall	93.48%	97.83%	97.83%	93.48%
Owner's class precision	94.64%	98.15%	98.15%	94.00%
Owner's class recall	98.15%	98.15%	98.15%	87.04%
Contractor's class F-Measure	95.56%	97.83%	97.83%	89.58%
Owner's class F-Measure	96.36%	98.15%	98.15%	90.39%
Kappa statistics	0.9195	0.9597	0.9597	0.9397
Mean absolute Error	0.0709	0.02	0.1315	0.0707
Root mean squared error	0.2165	0.1414	0.1214	0.1857
Relative absolute error	0.1425	0.4023	0.0373	0.1422
Root relative squared error	0.4339	0.2834	0.2636	0.3721

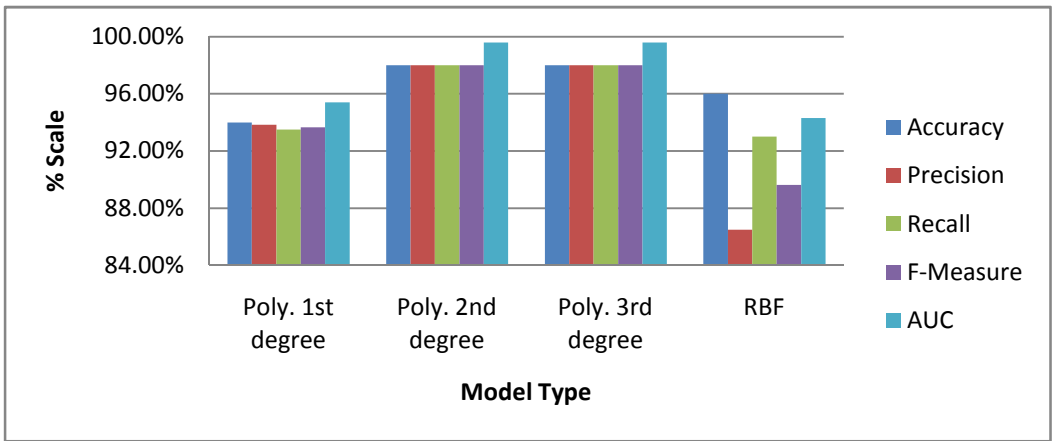


Figure 4.3 Accuracy, Precision, Recall, F-Measure, and AUC Results of SVM Modeling

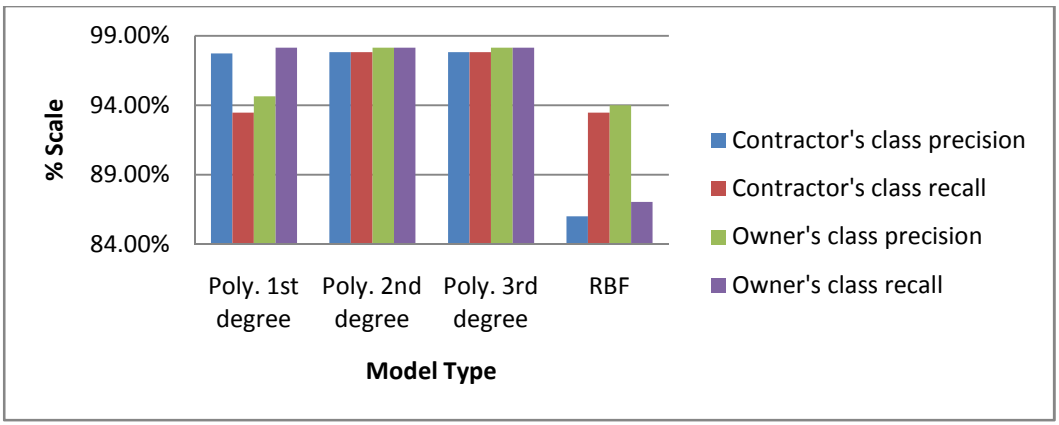


Figure 4.4 +Ve and -Ve Class Results of SVM Modeling

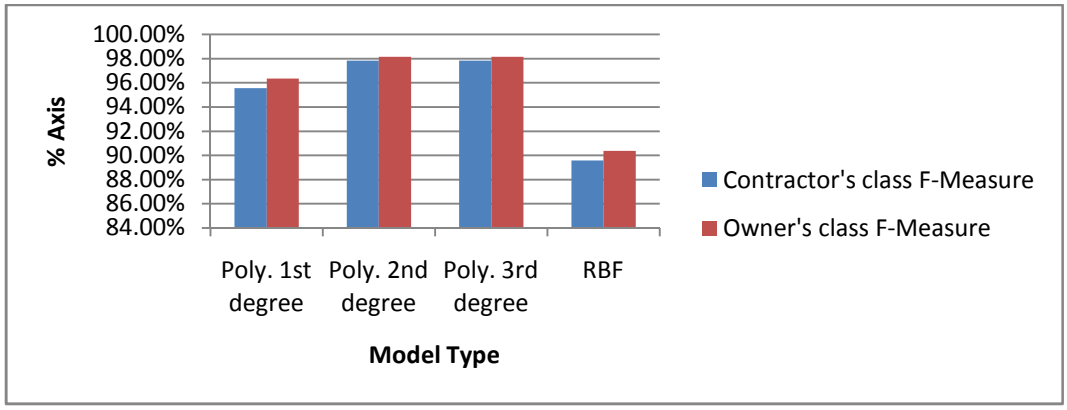


Figure 4.5 Class F-Measure Results of SVM Modeling

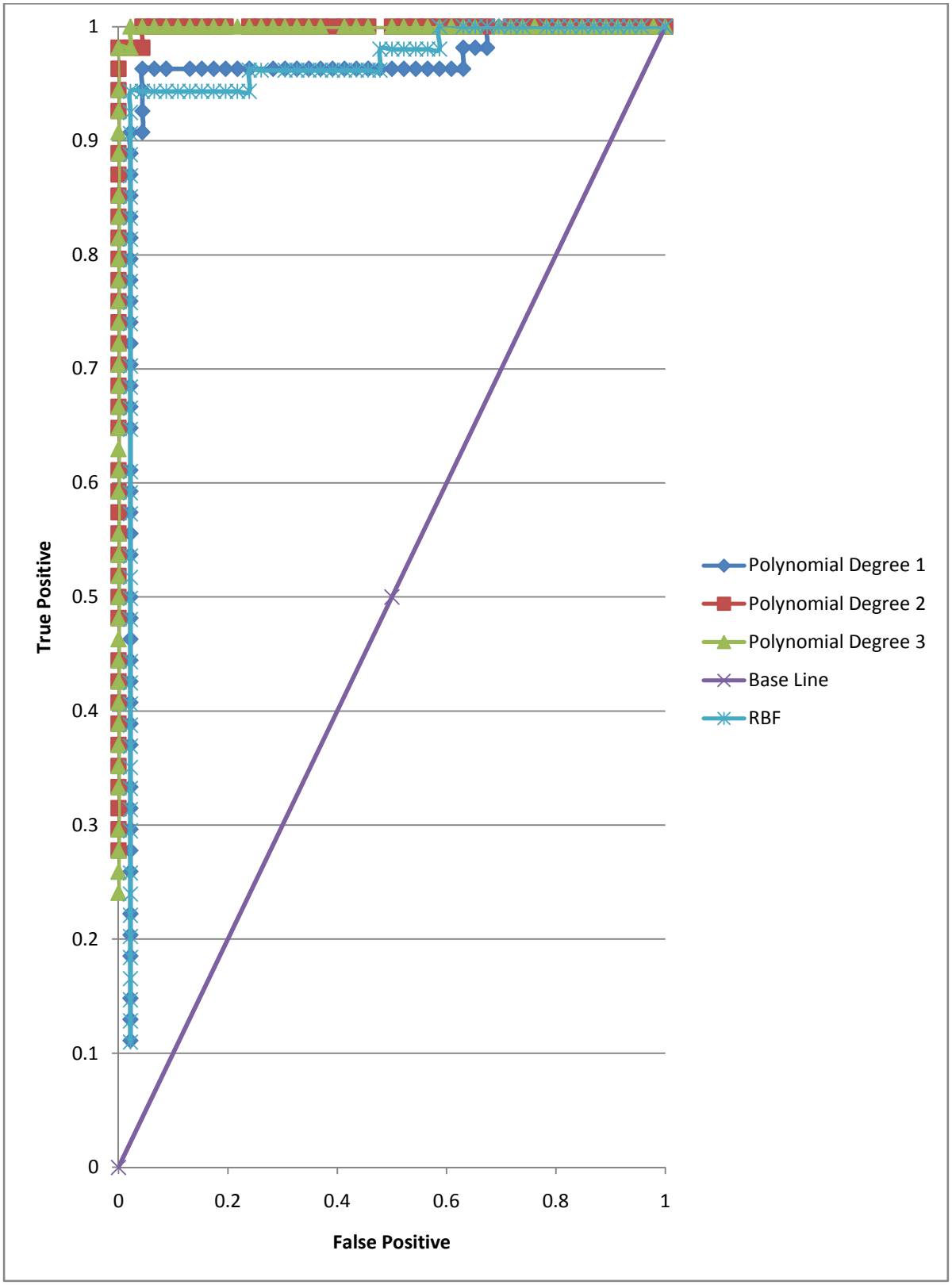


Figure 4.6 Area Under Curve (AUC) Results of SVM Modeling

4.6.2 Naïve Bayes Classifiers

The results of the testing and validation of the developed NB models are presented in tables 4.3, figures 4.7, 4.8, 4.9, 4.10, and appendix C respectively. The following is a closer examination and discussion of these results. As can be noted from table 4.3, both models have achieved similar results. Comparing the two models yields the followings:

- The Naïve Bayes classifier without the kernel estimators (Model 1) has its accuracy decreased by 1.00% over Naïve Bayes classifier with kernel estimators (Model 2).
- The precision of model 1 was higher than that attained by model 2 by a value of 1.94%.
- The recall of model 1 was less than that attained by model 2 by a value of 0.80%.
- The AUC of model 1 was higher than that attained by model 2 by a value of 5.00%.
- The Contractor's class precision of model 1 was less than that attained by model 2 by a value of 5.66%; while the class recall was increased by a value of 4.35%.
- The Owner's class precision of model 1 was higher than that attained by model 2 by a value of 3.22%; while the class recall was decreased by a value of 5.56%.

Table 4.3 Results of Naive Bayes Implementation

Property	Model #	
	NB with Kernel	NB without Kernel
Accuracy	93.00%	94.00%
Precision	92.94%	91.00%
Recall	93.20%	94.00%
F-Measure	93.07%	92.48%
AUC	94.30%	89.30%
Contractor's class precision	89.80%	95.45%
Contractor's class recall	95.65%	91.30%
Owner's class precision	96.08%	92.86%
Owner's class recall	90.74%	96.30%
Contractor's class F-Measure	92.63%	93.33%
Owner's class F-Measure	93.33%	94.55%
Kappa statistics	0.8598	0.8788
Mean absolute Error	0.095	0.1093
Root mean squared error	0.2251	0.2366
Relative absolute error	0.4512	0.4741

From the above information, it is clear that the performance of both models is nearly similar. Consequently, the basis of adopting one as being better than another will be based on the AUC measure. As mentioned earlier, AUC relates the true positive prediction rate of a model to its false positive prediction. As shown in table 4.3 and figure 4.10, model 1 and model 2 have achieved an AUC of 94.30% and 89.30% respectively. As a result, model 1 is estimated to classify a case to its appropriate class 94.30% of the times without making a mistake. On the other hand, model 2 is estimated to classify a case to its appropriate class 89.30% of the times without making a mistake. From the above, it is concluded that model 1, Naïve Bayes Classifier without implementing kernel estimators, is the best NB model.

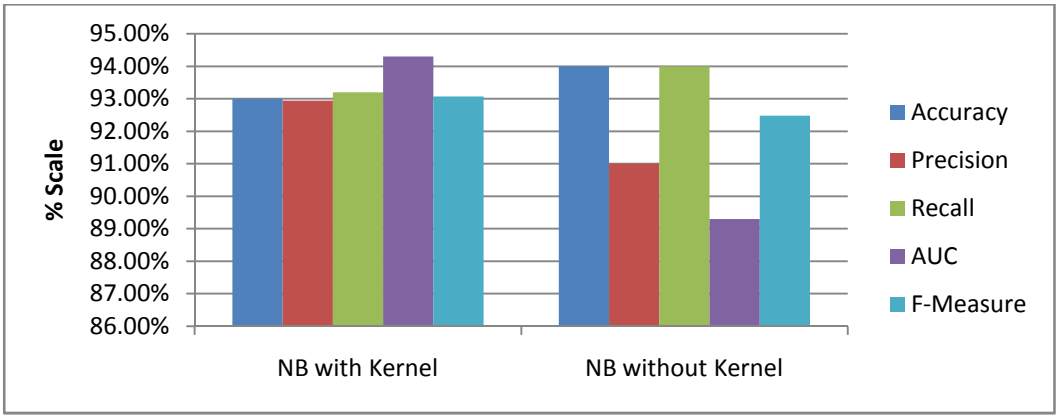


Figure 4.7 Accuracy, Precision, Recall, F-Measure, and AUC Results of Naive Bayes Modeling

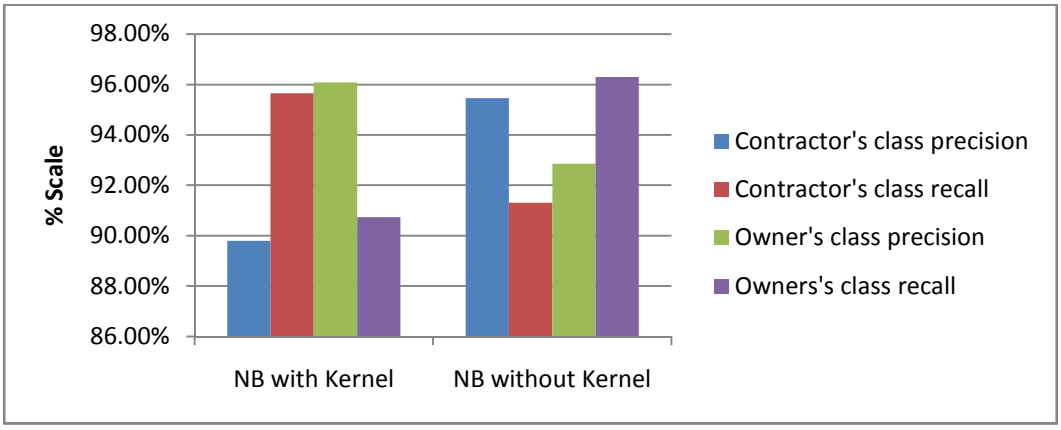


Figure 4.8 +Ve and -Ve Class Results of Naive Bayes Modeling

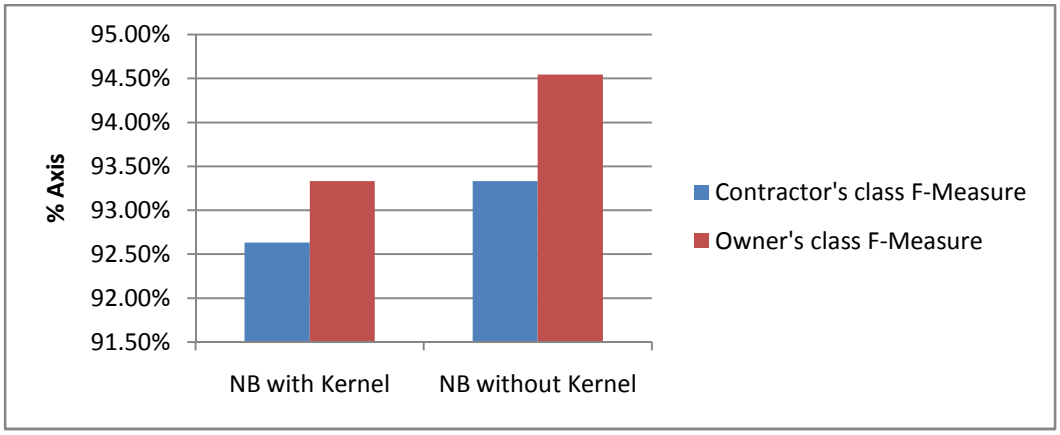


Figure 4.9 Class F-Measure Results of Naive Bayes Modeling

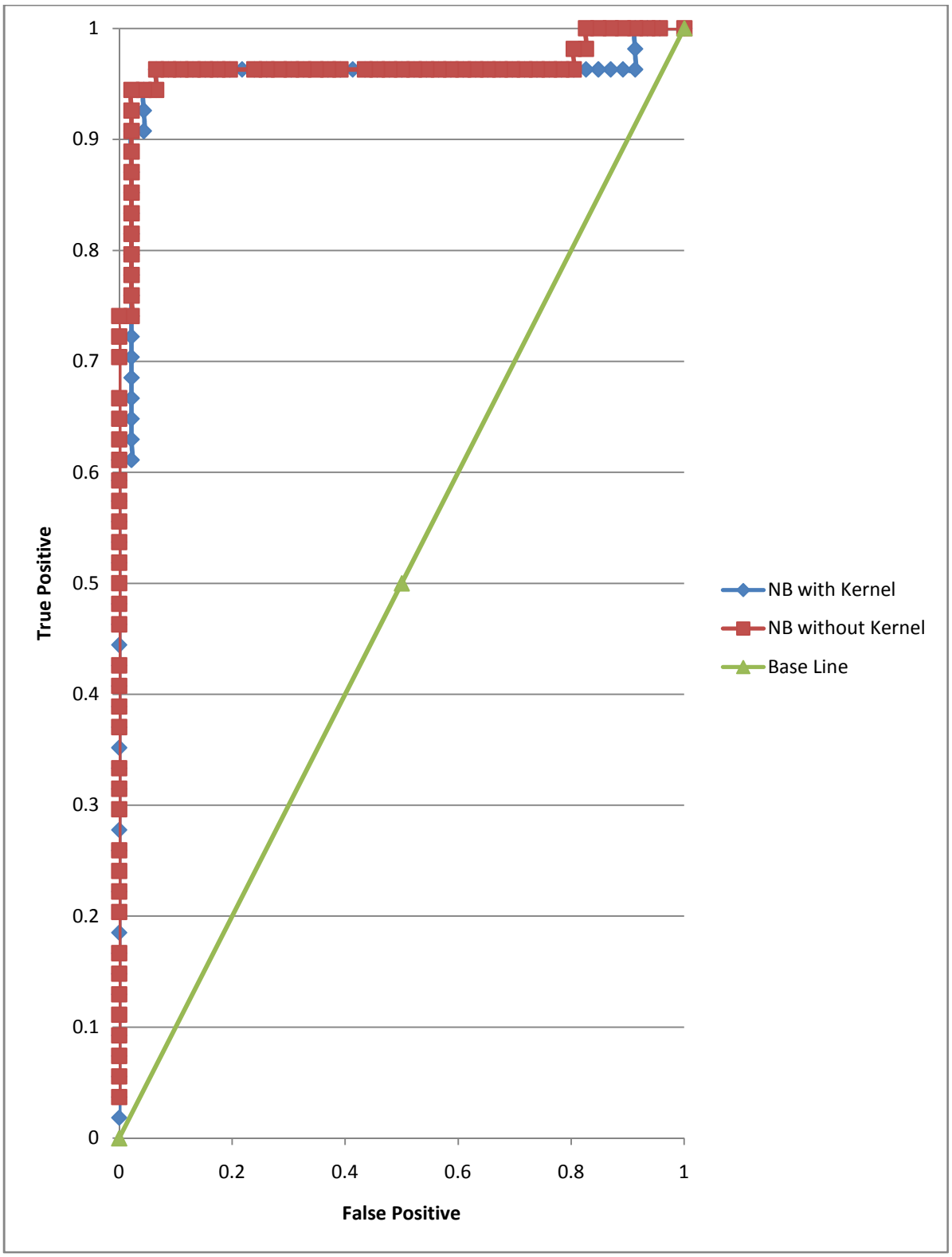


Figure 4.10 Area Under Curve (AUC) Results of Naive Bayes Modeling

4.6.3 Rule Induction Classifiers

The results of the testing and validation of the developed rule Induction models are presented in table 4.4, figures 4.11, 4.12, 4.13, 4.14, and appendix D respectively. The following is closer examination and discussion of these results. As can be noted from table 4.4, 15 boosts ADTree model performed the best. As shown in table 4.7, that the 15 boosts AD tree achieved higher performance with respect to all performance measures except the model recall. The 15 boost ADTree model achieved:

- An overall accuracy increase of 3.8%, 2.8%, and 2.8% over decision tree, 10 boosts ADTree, and PART respectively (please refer to figure 4.11).
- An increase in the precision of 4.03%, 2.66%, and 3.66% over decision tree, 10 boosts ADTree, and PART, respectively (refer to figure 4.11).
- A decrease in the recall of 0.00%, 0.88%, and 2.00% over decision tree, 10 boosts ADTree, and PART, respectively. As mentioned earlier in section 4.2.2.4, there is always a tradeoff between precision and recall. Consequently, F-measure is adopted to perform realistic comparison. The 15 boosts ADTree model achieved an increase in F-measure of 1.97%, 0.85%, and 0.79% over decision tree, 10 boosts ADTree, and PART, respectively (refer to figure 4.11).
- An increase in the AUC of 6.8%, 4.8%, and 2.4% over decision tree, 10 boosts ADTree, and PART, respectively (refer to figures 4.11 and 4.14).

- An increase in the Contractor's class precision of 4.35%, 0.15%, and 4.21% over decision tree, 10 boosts ADTree, and PART, respectively (refer to figure 4.11).
- An increase in the Contractor's class recall of 4.35%, 6.52%, and 2.17% over decision tree, 10 boosts ADTree, and PART, respectively (refer to figure 4.12).
- An increase in the Owner's class precision of 3.70%, 5.17%, and 1.92% over decision tree, 10 boosts ADTree, and PART, respectively (refer to figure 4.12).
- An increase in the Owner's class recall of 3.70%, 0.00%, and 3.70% over decision tree, 10 boosts ADTree, and PART, respectively (refer to figure 4.12).
- An increase in the Contractor's class F-measure of 4.35%, 3.44%, and 3.20% over decision tree, 10 boosts ADTree, and PART, respectively (refer to figure 4.13).
- An increase in the Owner's class F-measure of 3.70%, 2.65%, and 2.82% over decision tree, 10 boosts ADTree, and PART, respectively (refer to figure 4.13).
- An increase in the Kappa Statistics of 10.06%, 4.03%, and 10.06% over decision tree, 10 boosts ADTree, and PART respectively.

Table 4.4 Results of Rule Induction Classifiers Implementation

Property	Decision Tree	AD Tree (10 Boosts)	AD Tree (15 Boosts)	PART
Accuracy	94.00%	95.00%	97.80%	95.00%
Precision	93.96%	95.33%	97.99%	94.33%
Recall	94.00%	94.88%	94.00%	96.00%
F-measure	93.98%	95.10%	95.95%	95.16%
AUC	91.20%	93.20%	98.00%	95.60%
Contractor's class precision	93.48%	97.67%	97.83%	93.62%
Contractor's class recall	93.48%	91.30%	97.83%	95.65%
Owner's class precision	94.44%	92.98%	98.15%	96.23%
Owner's class recall	94.44%	98.15%	98.15%	94.44%
Contractor's class F-Measure	93.48%	94.38%	97.83%	94.62%
Owner's class F-Measure	94.44%	95.50%	98.15%	95.33%
Kappa statistics	0.8792	0.9397	0.9798	0.8792
Mean absolute Error	0.0662	0.0915	0.0727	0.0662
Root mean squared error	0.2352	0.1563	0.1356	0.2204
Relative absolute error	0.1331	0.1839	0.1462	0.1251
Root relative squared error	0.4715	0.3132	0.2719	0.4417

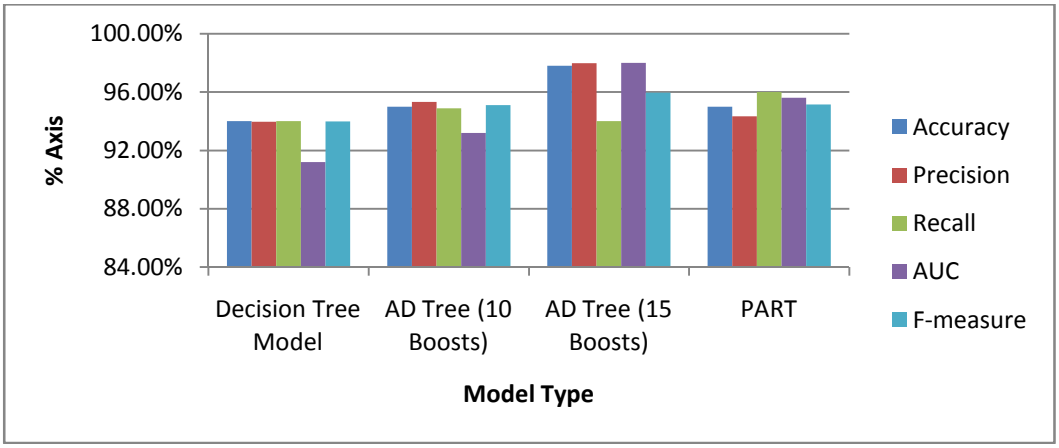


Figure 4.11 Accuracy, Precision, Recall, F-Measure, and AUC Results of Rule Induction Modeling

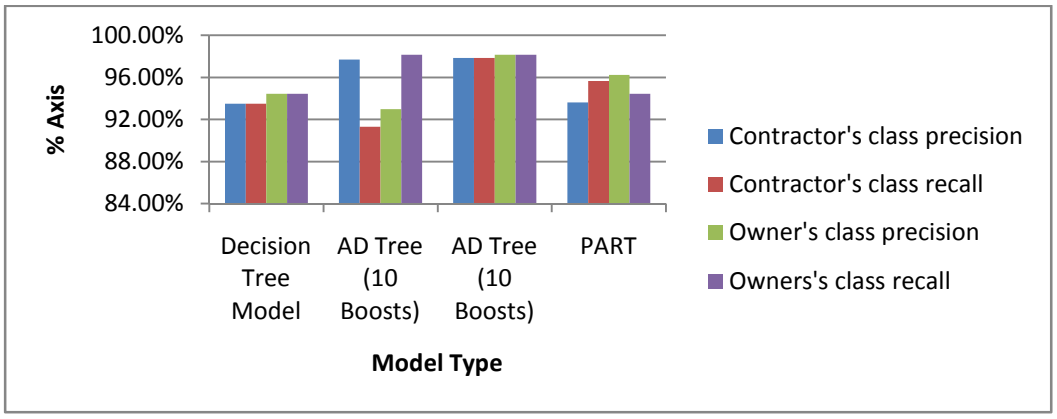


Figure 4.12 +Ve and -Ve Class Results of Rule Induction Modeling

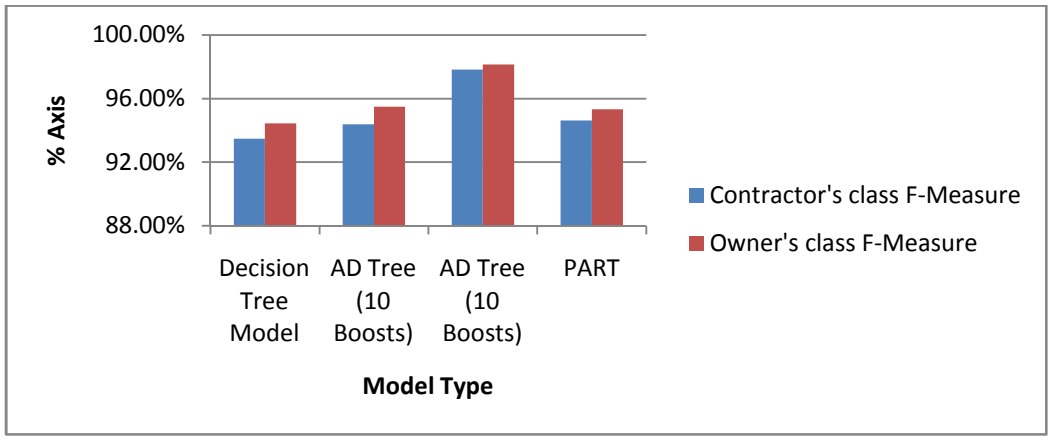


Figure 4.13 Class F-Measure Results of Rule Induction Modeling

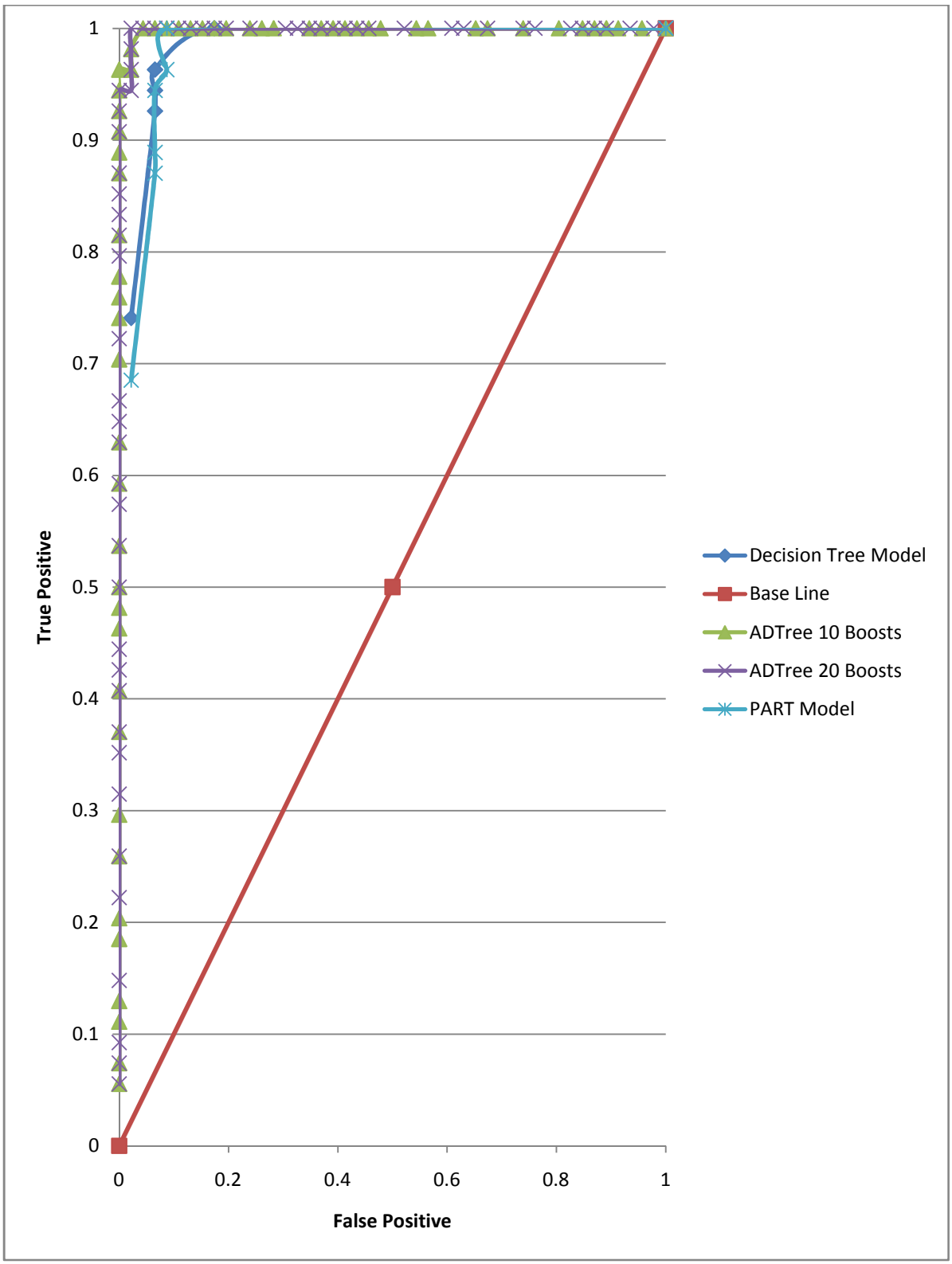


Figure 4.14 Area Under Curve (AUC) Results of Rule Induction Modeling

A comparison of the four models with respect to the derived decision rules was performed. The Decision Tree model generated a model with tree size of 13 and number of decision leaves 7 (Figure 4.15). The model derived the following rules:

```
DSC <= 0
|   Ocause <= 0: OWNER (43.0)
|   Ocause > 0: CONTRACTOR (3.0/1.0)
DSC > 0
|   SpecWarn <= 0
|   |   CNoExtra <= 0: CONTRACTOR (36.0)
|   |   CNoExtra > 0
|   |   |   DSCC <= 0
|   |   |   |   Conraise <= 0: OWNER (4.0/1.0)
|   |   |   |   Conraise > 0: CONTRACTOR (7.0)
|   |   |   DSCC > 0: OWNER (2.0)
|   SpecWarn > 0: OWNER (5.0)
```

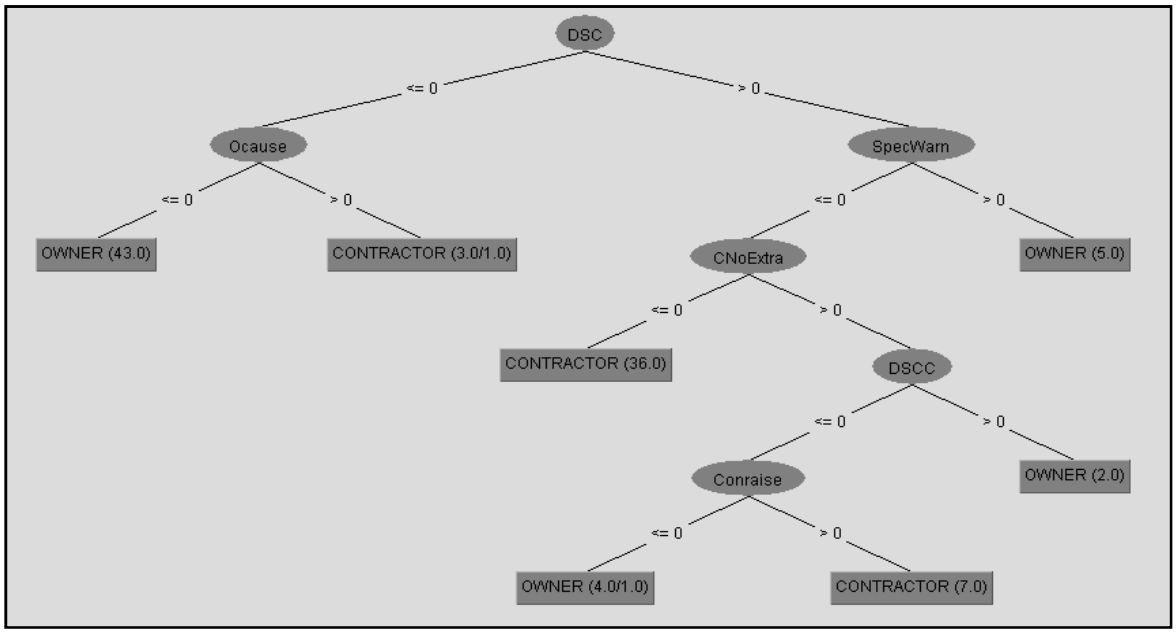


Figure 4.15 Decision Tree Model Output

The 10 boosts ADTree generated a tree size of 25 with 17 decision nodes (Figure 4.16). The model derived the following rules:

| (1)DSC < 0.5: -1.289
| | (3)ComImpossible < 0.5: -1.793
| | (3)ComImpossible >= 0.5: 1.469
| (1)DSC >= 0.5: 0.778
| | (2)CNoExtra < 0.5: 2.141
| | (2)CNoExtra >= 0.5: -0.902
| | | (7)DSCC < 0.5: 0.177
| | | | (8)Conraise < 0.5: -0.492
| | | | (8)Conraise >= 0.5: 0.55
| | | (7)DSCC >= 0.5: -0.754
| | (5)N&C < 0.5: -0.355
| | (5)N&C >= 0.5: 0.725
| (4)SpecWarn < 0.5: 0.373
| (4)SpecWarn >= 0.5: -1.002
| (6)Ocause < 0.5: -0.44
| (6)Ocause >= 0.5: 0.474
Legend: -ve = OWNER, +ve = CONTRACTOR

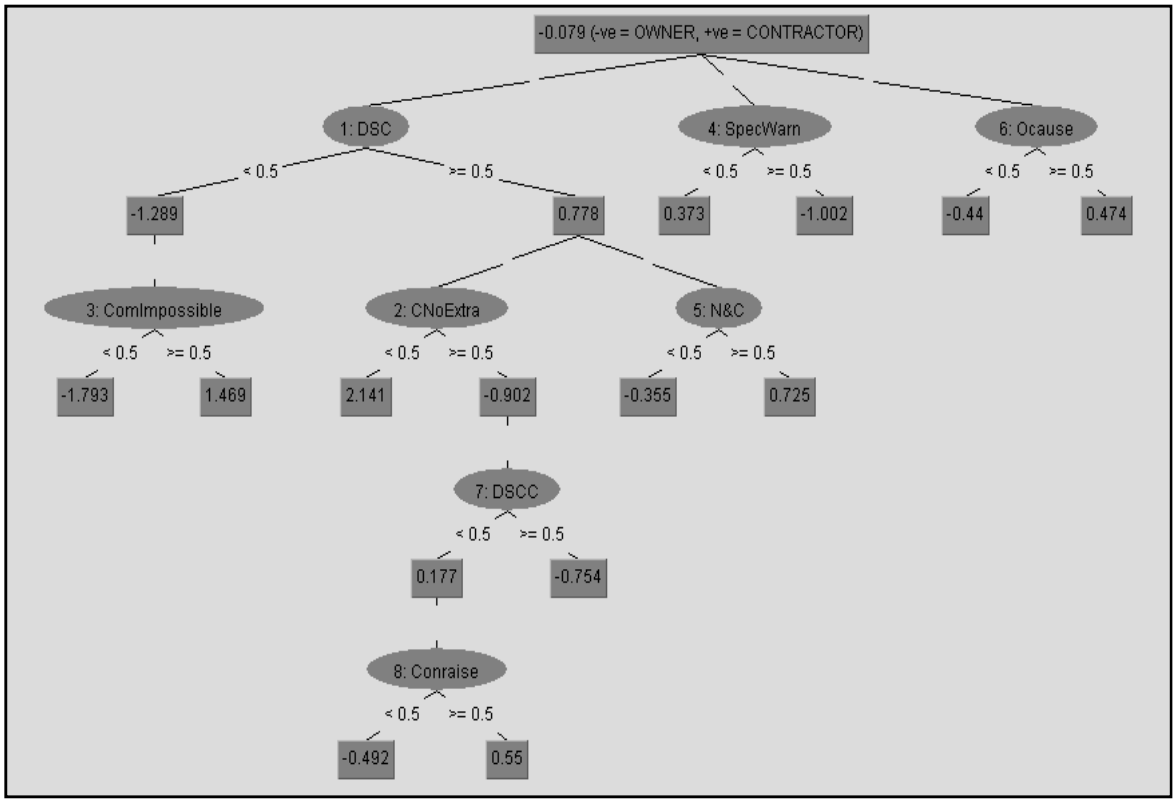


Figure 4.16 ADTree Model Output

Finally, the 15 boosts ADTree generated a tree size of 37 with 25 decision nodes (Figure 4.17). The model derived the following rules:

```
|  (1)DSC < 0.5: -1.289
|  |  (3)ComImpossible < 0.5: -2.173
|  |  (3)ComImpossible >= 0.5: 1.756
|  (1)DSC >= 0.5: 0.778
|  |  (2)CNoExtra < 0.5: 2.141
|  |  (2)CNoExtra >= 0.5: -0.902
|  |  |  (7)DSCC < 0.5: 0.177
|  |  |  |  (8)Conraise < 0.5: -0.492
|  |  |  |  (8)Conraise >= 0.5: 0.55
|  |  |  |  |  (9)SpecWarn < 0.5: 0.579
|  |  |  |  |  (9)SpecWarn >= 0.5: -0.223
|  |  |  (12)Ocause < 0.5: -0.217
|  |  |  (12)Ocause >= 0.5: 0.403
|  |  (7)DSCC >= 0.5: -0.754
|  |  (10)N&C < 0.5: -0.397
|  |  (10)N&C >= 0.5: 0.285
|  (5)N&C < 0.5: -0.355
|  (5)N&C >= 0.5: 0.725
|  |  (11)SpecWarn < 0.5: 0.476
|  |  (11)SpecWarn >= 0.5: -0.164
|  (4)SpecWarn < 0.5: 0.373
|  (4)SpecWarn >= 0.5: -1.002
|  (6)Ocause < 0.5: -0.44
|  (6)Ocause >= 0.5: 0.474
Legend: -ve = OWNER, +ve = CONTRACTOR
```

In addition, the PART model generated 3 decision rules as follows.

```
DSC <= 0 AND Ocause <= 0: OWNER (43.0)
SpecWarn <= 0 AND CNoExtra <= 0: CONTRACTOR (38.0)
SpecWarn <= 0 AND DSCC <= 0 AND Conraise > 0: CONTRACTOR (7.0)
```

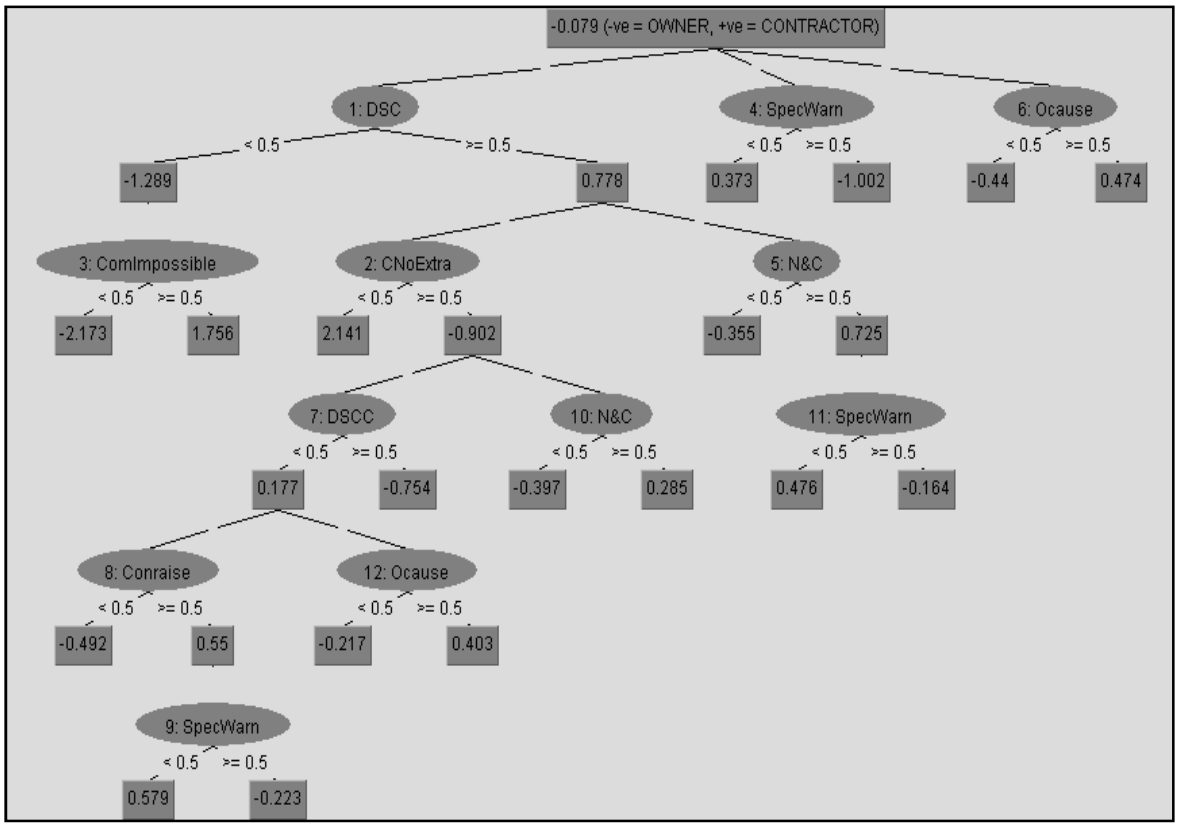


Figure 4.17 Pictorial Representation of the 15 Boost ADTree Model Output

From the above information, it is clear that the performance of the four developed models under this sub-task had achieved a higher performance than the base line since they have achieved an accuracy higher than 50%. In addition, comparing the four developed models namely Decision Tree, ADTree with 10 boosts, ADTree with 15 boosts, and PART yielded the ADTree model with 15 boosts with the best performance under the adopted research approach.

4.7 Analysis and Discussion

As can be noted from the above results, the Kernel Polynomial 3rd degree, Naïve Bayes without Kernel estimators, and ADTree with 15 boosts models attained the best performance measures within the studied SVM, Naïve Bayes, and Inductive

Rule classifiers respectively. Table 4.5, and figures 4.18, 4.19, 4.20, and 4.21 illustrate comparisons between these models. Comparing the outcomes of the three models one deduces that the SVM Kernel Polynomial 3rd degree achieved higher performance measure over the other two. It had attained the followings.

- An increase in the overall accuracy of 5.00% and 0.2% from the Naïve Bayes without Kernel estimators and ADTree with 15 boosts models, respectively (refer to figure 4.18).
- An increase in the precision of 5.06% and 0.01% from the Naïve Bayes without Kernel estimators and ADTree with 15 boosts models, respectively (refer to figure 4.18).
- An increase in the recall of 4.80% and 4.00% from the Naïve Bayes without Kernel estimators and ADTree with 15 boosts models, respectively (refer to figure 4.18).
- An increase in the F-measure of 4.93% and 2.05% from the Naïve Bayes without Kernel estimators and ADTree with 15 boosts models, respectively (refer to figure 4.18).
- An increase in the AUC of 5.30% and 3.65% from the Naïve Bayes without Kernel estimators and ADTree with 15 boosts models, respectively (refer to figures 4.18 and 4.21).
- An increase in the Contractor's class precision of 8.03% from the Naïve Bayes without Kernel estimator model. However, a minor decrease of 0.17% from ADTree with 15 boosts model was noticed (refer to figure 4.19).

Table 4.5 Output Analysis of the Best Models

Property	Model Type		
	SVM (Poly. 3rd Degree)	Naive Bayes	ADTree (15 Boosts)
Accuracy	98.00%	93.00%	97.80%
Precision	98.00%	92.94%	97.99%
Recall	98.00%	93.20%	94.00%
F-Measure	98.00%	93.07%	95.95%
AUC	99.60%	94.30%	95.95%
Contractor's class precision	97.83%	89.80%	98.00%
Contractor's class recall	97.83%	95.65%	97.83%
Owner's class precision	98.15%	96.08%	97.83%
Owner's class recall	98.15%	90.74%	98.15%
Contractor's class F-Measure	97.83%	92.63%	98.15%
Owner's class F-Measure	98.15%	93.33%	97.83%
Kappa statistics	95.97%	85.98%	98.15%
Mean absolute Error	13.15%	9.50%	97.98%
Root mean squared error	12.14%	22.51%	7.27%
Relative absolute error	3.73%	19.11%	13.56%
Root relative squared error	26.36%	45.12%	14.62%

- An increase in the Contractor's class recall of 2.17% from the Naïve Bayes without Kernel estimator model was noticed. However, no improvement over ADTree with 15 boosts model was detected (refer to figure 4.19).
- An increase in the Owner's class precision of 2.07% and 0.32% from the Naïve Bayes without Kernel estimators and ADTree with 15 boosts models respectively (refer to figure 4.19).

- An increase in the Owner's class recall of 7.41% from the Naïve Bayes without Kernel estimator model was noticed. However, no improvement over ADTree with 15 boosts model was detected (refer to figure 4.19).
- An increase in the Contractor's class F-Measure of 5.19% from the Naïve Bayes without Kernel estimator model. However, a minor decrease of 0.32% from ADTree with 15 boosts model was noticed (refer to figure 4.20).
- An increase in the Owner's class F-Measure of 4.81% and 0.32% from the Naïve Bayes without Kernel estimators and ADTree with 15 boosts models, respectively (refer to figure 4.20).
- An increase in the Kappa of 9.99% from the Naïve Bayes without Kernel estimator model. However, a decrease of 2.18% from ADTree with 15 boosts model was detected.

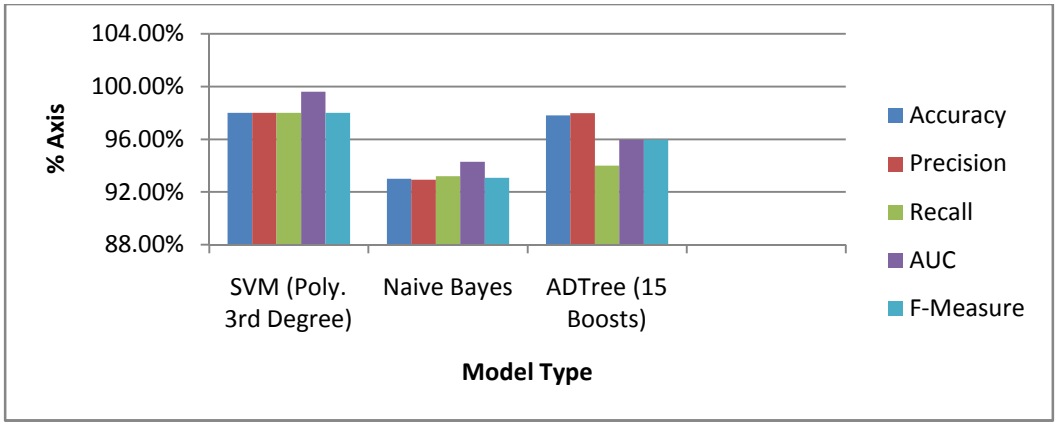


Figure 4.18Accuracy, Precision, Recall, F-Measure, and AUC Results of the Best Developed Models

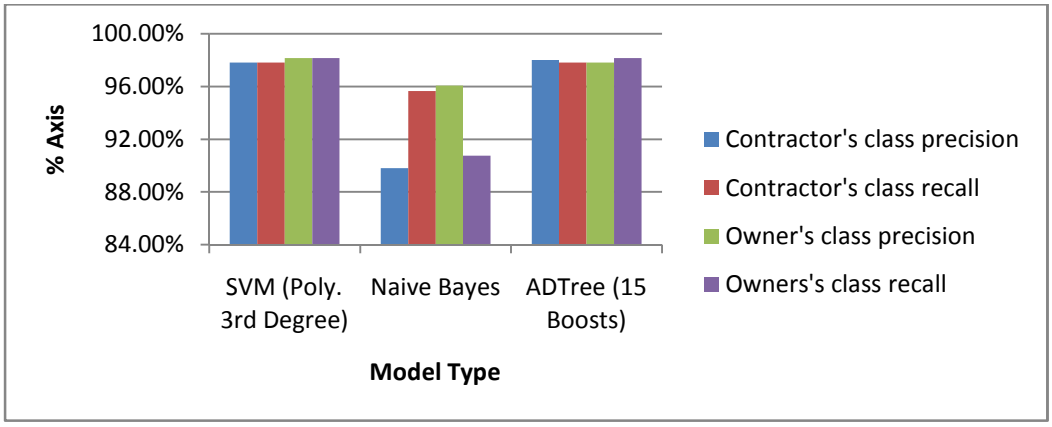


Figure 4.19 +Ve and -Ve Class Results of the Best Developed Models

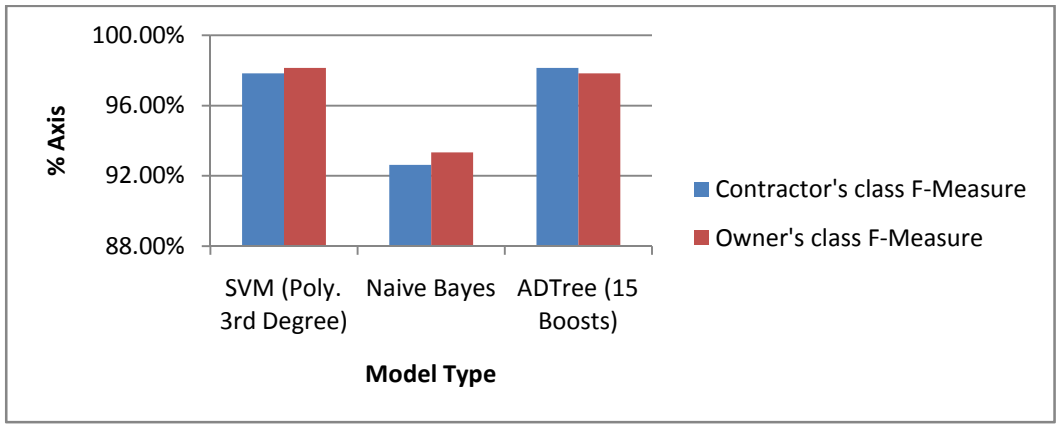


Figure 4.20 Class F-Measure Results of the Best Developed Models

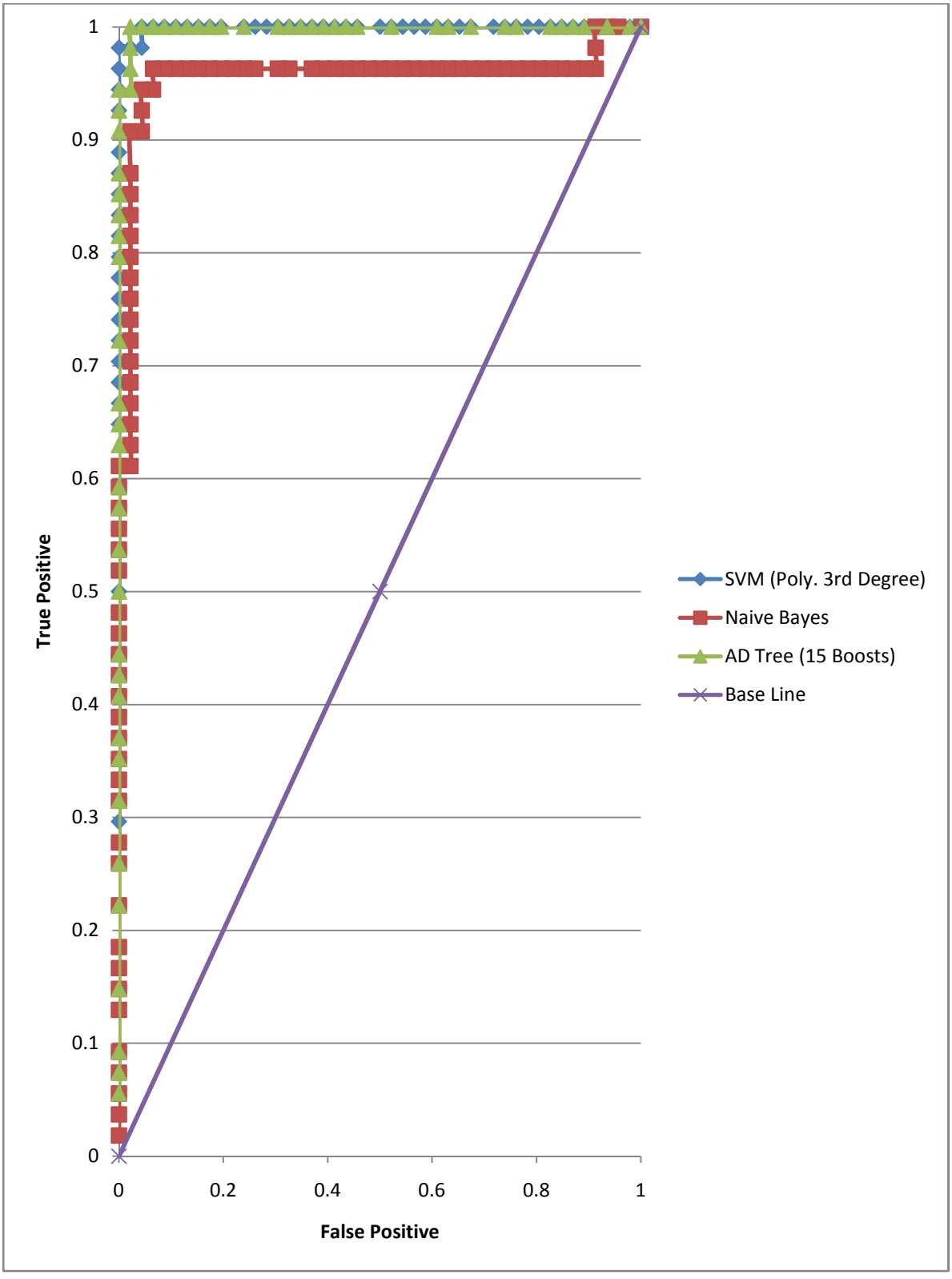


Figure 4.21 Area Under Curve (AUC) Results of the Best Developed Models

Comparing the best three developed models namely SVM Kernel Polynomial 3rd degree, Naïve Bayes without Kernel estimators, and ADTree with 15 boosts yielded the SVM Kernel Polynomial 3rd degree model with the best performance under the adopted research design and implementation.

The achieved superiority of the SVM Kernel Polynomial 3rd degree is supported by outcomes of previous research studies in the literature review. However, it provides very significant insight on the nature of the problem being investigated as follows:

(1) The problem analyzed is a real life complex one in which simple prediction tools like NB and Rule Induction classifiers cannot analyze its extent fully. The classification of a legal case in terms of whether it is to be judged in favor of one party over the other integrates a lot of factors that are not linearly separable in nature. Consequently, simple classifiers are not suitable for the task. However, Support Vector Machine (SVM) is a state-of-the-art classification and regression algorithm, which implements strong regularization techniques, that is, the optimization procedure maximizes predictive accuracy while automatically avoiding over-fitting of the training data (Cannon et al. 2007). Furthermore, the transformation of the data into a higher dimension space through Kernel estimation provides the strength of the SVM model in solving this complex problem. On the other hand, NB makes predictions using Bayes' Theorem, which derives the probability of a prediction from the underlying evidence. This theory and the inherent assumption that cases are mutually exclusive limit the performance of the model.

(2) The analysis utilizes 120 cases and is considering 15 features. The fact that the number of cases is more than twice the number of features makes SVM a stronger tool for the analysis of the current problem due to its active learning feature (Oracle 2009). “SVM models grow as the size of the training data set increases…. Active learning forces the SVM algorithm to restrict learning to the most informative training examples and not to attempt to use the entire body of data” (Oracle 2009). Furthermore, SVM is not dependent on general rules. In rule dependent classifiers and NB, the number of collected rules is dependent on the size of the dataset. Consequently, the lower performance in NB and Rule Induction Classifiers could be attributed to the number of features analyzed. The number of features selected may not be enough to accurately differentiate the cases for those algorithms.

4.8 Chapter summary

The objective of this chapter was to investigate the feasibility of ML use for the development of a DSC litigation prediction model for the construction industry. To that end, SVM, Naïve Bayes, and Induction Rule classifiers were adopted for the study. 10 models were developed in the following manner 4 SVM, 2 Naïve Bayes, and 4 Induction rule models. The highest prediction rate of 98% within the first category was attained by Kernel Polynomial 3rd degree model. Models developed under the second category yielded a highest rate of prediction of 93% attained by the Naïve Bayes model without implementing kernel estimators. A prediction rate of 97.8% was the highest attained within the third category by ADTree model with 15 boosts. Comparing the outputs of all developed models yields a great advancement

in this area when compared to a base line of 50% and previously performed researches (Arditi and Tokdemir 1999, and Chau 2006) as discussed in chapter 2. In addition, after performing the aforementioned analysis, it could be concluded that SVM Kernel Polynomial 3rd degree model has achieved the best performance among all developed models.

CHAPTER 5

AUTOMATED EXTRACTION OF SIGNIFICANT LEGAL FACTORS

5.1 Introduction

Researchers have highlighted knowledge integration and knowledge management as two of the major problems that affect the efficiency of the construction industry (Caldas et al. 2002, Wood 2000, Brűggemann et al. 2000, and Kosovac et al. 2000). The problem is attributed to the fact that (Caldas et al. 2002): (1) a large amount of construction data is stored in semi-structured and unstructured files and formats; (2) the knowledge needed for construction decision making is very difficult to extract; (3) this knowledge is not integrated with other construction management systems; and (4) the association between construction data and their related project components is not clear. These facts make the management of construction knowledge a significant and challenging task.

In fact, the aforementioned challenges in managing construction knowledge extend to the legal domain, since cases are also stored in textual unstructured formats (Ashley and Rissland 1988). The highly sophisticated electronic information storage and retrieval systems available for researching the laws and case histories are extremely complex and time consuming. Sometimes this complexity creates problems for information seekers and can limit their access to relevant information. This adds to the complexity of the legal decision making process in construction, since the process is time consuming and may require knowledgeable professionals to extract the required knowledge from relevant case histories. As a result, an

automated legal decision support system that utilizes natural language processing techniques to identify, retrieve, reorganize legal information, and predict construction litigation outcomes is needed. This system is expected to reduce the time required and costs incurred by construction firms in the legal decision making process and improve overall project control.

The previous chapter of this dissertation illustrated the development of machine learning models that efficiently and effectively determine the outcomes of DSC disputes construction based on corpus of precedent cases. Those models are expected to help in relieving the negative consequences associated with lengthy DSC claim and dispute resolution in the construction industry. However, the manual extraction of significant legal factors that govern these cases that form the corpus is a significant time constraint that could reduce the efficiency of these models. The main goal of this chapter is to develop an automated methodology for the extraction of legal knowledge, in the form of significant legal factors, from precedent cases. Consequently, the focus of this chapter is to develop and evaluate the performance of different ML tools, namely Support Vector Machines (SVM), Naïve Bayes, and Inductive Rules, in an attempt to automate legal factors identification.

The research tasks described in this chapter will, therefore, include: (1) preparing the data for model implementation; (2) identifying the ML model parameters; (3) developing SVM, Naïve Bayes, and Rule Induction automated extraction models; and (4) validating and comparing the developed models.

5.2 Data Preparation

The first research task in this chapter aims at preparing the cases in the DSC precedent corpus for processing using the different ML methods that are being developed. The data preparation task is, therefore, composed of the following three steps: (1) defining the nature of the problem; (2) processing the collected data; and (3) preparing the processed data for model development (weighting scheme).

5.2.1 Defining the Nature of the Problem

The goal of this task is to automate the process of legal significant factors extraction in textual precedent cases. This implies that the knowledge that needs to be extracted is implicitly available within the textual cases. Consequently, this problem can be defined as one of extracting this tacit knowledge from a large text. The first step in solving such a problem is to analyze the text to evaluate how this tacit knowledge can be extracted. Each case includes a representation of the different legal factors in terms of words that are put together in a coherent manner to derive meaning. However, looking at the bigger picture, legal terms always refer to constant meanings. For example, the word contract legally refers to “A binding agreement between two or more parties for performing, or refraining from performing, some specified act(s) in exchange for lawful consideration” (Legal Dictionary 2008). This decreases the ambiguity of these terms, but also decreases their ability to distinguish between documents. In the same manner, each case will include terms that are pertinent to a specific legal factor defined in chapter 3.

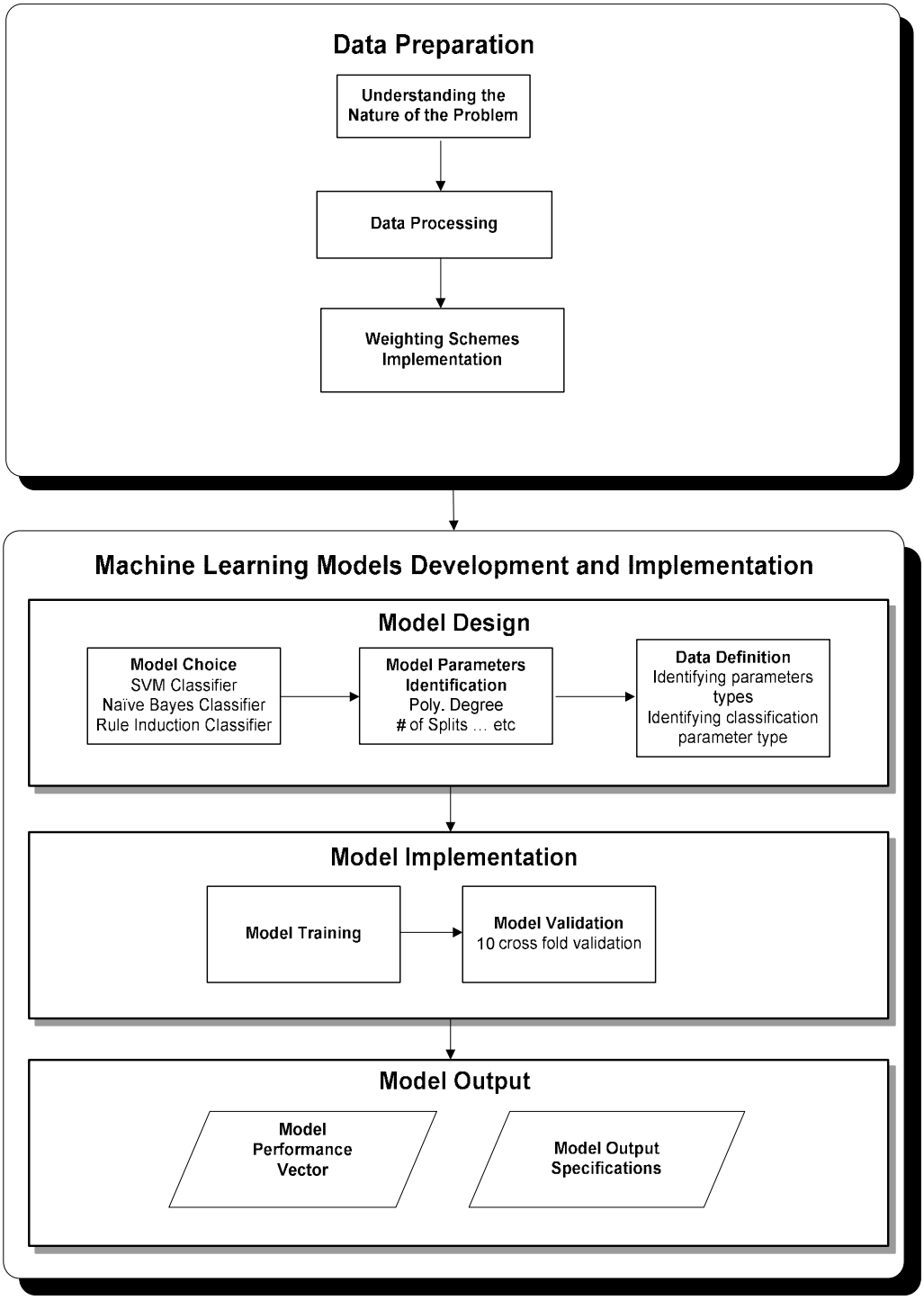


Figure 5.1 Research Tasks for Automated Significant Legal Factors Extraction

For example, in the case of All County Paving Corp., Doing Business as Collins Construction Co., Appellant, v Suffolk County Water Authority, Respondent, Judges Anita R. Florio, J.P., Robert W. Schmidt, Thomas A. Adams, and William F. Mastro stated in their opinion “since the defendant made no misrepresentations and withheld no information, the plaintiff was not entitled to extra compensation”. This sentence includes terms like misrepresentations, withheld, and information in a manner that relates to the legal factor “MMistake”. Consequently, these terms are the ones that the ML algorithms could use to determine the presence of this factor.

5.2.2 Processing the Case Corpus

As stated above, the 120 cases, earlier utilized for the analysis in chapter 4, were utilized for the analysis under this sub-task. The decision for using this set of cases was based on the fact that they were previously manually analyzed to define the significant legal factors pertinent to each case. As mentioned earlier, these cases are related to DSC disputes from The Federal Court of New York. They were filled in the period between 1912 and 2007. Although each case implicitly includes the required knowledge for such analysis, it also includes textual representations that are not related to this sub-task. This step involves preparing the collected dataset in an appropriate manner to enhance the analysis. Consequently, the processing step will include data cleaning, data integration, and data reduction (Ng. et al. 2006). A similar methodology has been utilized in the application of text mining techniques in construction as mentioned earlier in chapter 2 (Caldas et al. 2002). Data cleaning is performed by removing undesirable text (words). For more

illustrations, textual representation of cases might include frequent words that carry no meaning, misspelled words, outliers, noise, and inconsistent data. While data processing is performed on each textual case representation separately, data integration is performed over the entire dataset. In this step, the entire processed dataset is stored in a coherent manner that facilitates their use for further analysis. While the integrated data might be very large, data reduction can decrease the data size by aggregating and eliminating redundant features.

To perform the aforementioned sub-steps, an algorithm was developed and implemented in C++. A copy of the developed program is provided in Appendix E of this dissertation. The basic principle of the developed program is to represent each document as a vector of certain weighted word frequencies. The following steps outline the parsing and extraction procedure that are performed on each textual representation of a case (please refer to figure 5.2).

1. Extract all words in a document. The algorithm prompts the user to provide a directory that includes the document. The algorithm iterates through the documents one by one, associates each document with a unique numerical code and extracts all words in each document. Words are extracted based on white spaces and are stored in a document vector that is coded with the unique document code.
2. Eliminate non-content-bearing words, also known as stopwords (Rijsbergen 1979). The algorithm utilizes predeveloped files including a comprehensive list of non-content-bearing words. For example, words like and, if, or, then, that, the, he, me, they, nevertheless … etc. are

included in the file. Each word in the document vector is compared against these lists. If a word was found to be non-content-bearing words, it was excluded from the document vector.

3. Reduce each word to its "root" or "stem" eliminating plurals, tenses, prefixes, and suffixes. This technique is called stemming, suffix stripping, or term truncation. Stemming reduces different word forms to common roots. The purpose of stemming is to group words that are morphological variants on the same word stem (Porter 1980, Ng et al. 2006). The algorithm iterates through the document word vector stemming each word by inputting it into a loop that performs the following:
    a. All letters in the word are changed to lower case. For example, the word "**C**ontracts;" is stemmed to "contracts;".
    b. All punctuations and non alphabetical symbols or marks that are used to organize writing are removed. Consequently, the word "contracts**;**" is stemmed to "contracts".
    c. All words are converted to its singular form. The algorithm utilizes standard grammatical rules of pluralization to perform this step. Each word in the document word vector is transformed to its singular form by eliminating "s" or "es" or "ies" at its end. As a result, the word "contract**s**" is stemmed to "contract". Words in the document vector are replaced with their stemmed versions.
4. For each document, count the number of occurrences of each word. The algorithm iterates through each document vector and counts the number

of occurrences of each stemmed word. These frequencies are stored in a document term frequency vector that is coded with the unique document code generated by the algorithm.

5. Eliminate low frequency words (Salton 1989, Ng et al. 2006). Low frequency words are those that were repeated less than three times in a document. The algorithm iterates through the document frequency vector excluding each word with a frequency less than three from the document frequency and word vectors.

After the previous procedure, *w* unique words remain in *d* unique documents; a unique identifier is assigned between 1 and *w* to each remaining word, and a unique identifier between 1 and *d* to each document resulting in a term-frequency (tf) matrix.

5.2.3 Weighting scheme development

A mere representation of significant words in the form of term frequency is not sufficient to accurately extract the required knowledge from our case corpus. For example, a word like contract might exist in all processed documents in high frequency (tf). However, a decision must be made about whether this word would help define a significant legal concept or not. Consequently, an appropriate weighting mechanism must be implemented to create a representative matrix of these documents within the entire dataset. Literature in the field of ML and text mining illustrated the effectiveness of alternate term weighting schemes like logarithmic term frequency (ltf) (equation 5.1), augmented weighted term frequency

(atf) (equation 5.2), and term frequency inverse document frequency (tf.idf) (equation 5.3) (Manning and Scheutze 1999).

$$ltf_{i,d}=1+\log(tf_{i,d})\,;\ tf_{i,d}>0 \qquad 5.1$$

$$atf_{i,d}=0.5+\frac{0.5\times tf_{i,d}}{\max_t(tf_{i,d})} \qquad 5.2$$

$$tf.idf_{i,d}=(1+\log(tf_{i,d}))\times\log\left(\frac{N}{df_i}\right)\ \text{if}\ tf_{i,d}>0 \qquad 5.3$$

The four above mentioned weighting schemes namely tf, ltf, atf, and tf.idf were utilized for the analysis under this task. The developed algorithm mentioned in 5.2.2 implements the required calculations to develop 4 matrixes (please refer to figure 5.2).

5.2.4 ML Model Development

The adopted research approach developed kernel SVM, Naïve Bayes, and Rule Induction models that detect the presence of a significant legal factor in a case to its text. The proposed research approach developed and compared the outputs of 24 models illustrated in table 5.1. Validation of the best developed model was based on prediction accuracy and Kappa measures. Outputs of the developed models are compared to a base line prediction of 50%. Since the analysis is pertinent to automating the extraction of significant legal factors related to each case, each model is developed as a multiple classifier. In other words, each case is tagged with the set of existing significant legal factors defined manually in chapter 3.

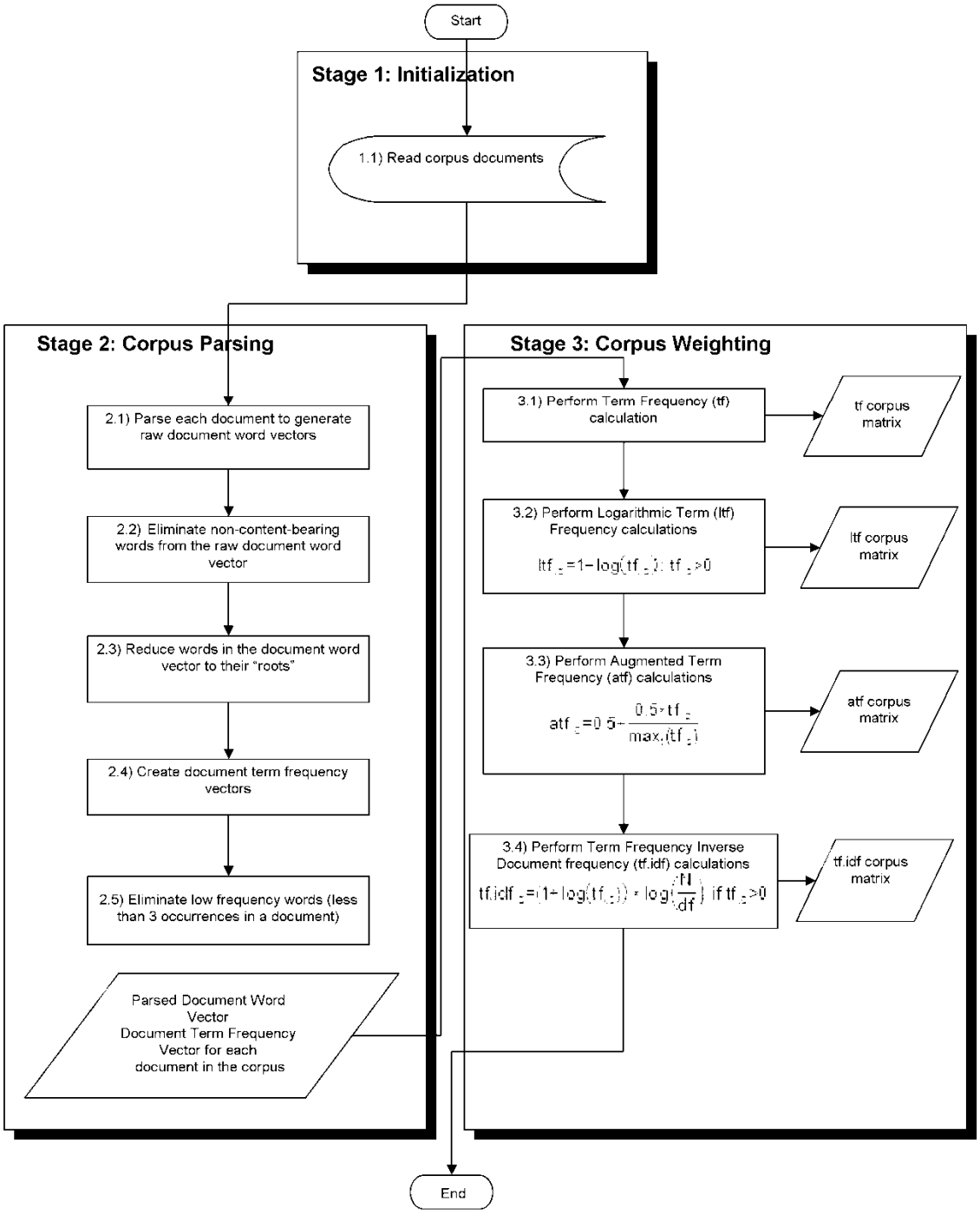


Figure 5. 2 Algorithm Implementation

The classifier is trained to predict the factors in the form of a set based on the significant words available in a text. The training and validation of the model is performed on a 10 fold scheme as detailed in chapter 4.

Table 5.1 ML Developed Models

Model #	ML Model Type	Weighting Scheme
1	1st Degree Polynomial SVM	Term Frequency (tf)
2	2nd Degree Polynomial SVM	Term Frequency (tf)
3	3rd Degree Polynomial SVM	Term Frequency (tf)
4	Naïve Bayes	Term Frequency (tf)
5	Decision Tree	Term Frequency (tf)
6	PART	Term Frequency (tf)
7	1st Degree Polynomial SVM	Logarithmic term frequency (ltf)
8	2nd Degree Polynomial SVM	Logarithmic term frequency (ltf)
9	3rd Degree Polynomial SVM	Logarithmic term frequency (ltf)
10	Naïve Bayes	Logarithmic term frequency (ltf)
11	Decision Tree	Logarithmic term frequency (ltf)
12	PART	Logarithmic term frequency (ltf)
13	1st Degree Polynomial SVM	Augmented term Frequency (atf)
14	2nd Degree Polynomial SVM	Augmented term Frequency (atf)
15	3rd Degree Polynomial SVM	Augmented term Frequency (atf)
16	Naïve Bayes	Augmented term Frequency (atf)
17	Decision Tree	Augmented term Frequency (atf)
18	PART	Augmented term Frequency (atf)
19	1st Degree Polynomial SVM	Term frequency inverse document frequency (tf.idf)
20	2nd Degree Polynomial SVM	Term frequency inverse document frequency (tf.idf)
21	3rd Degree Polynomial SVM	Term frequency inverse document frequency (tf.idf)
22	Naïve Bayes	Term frequency inverse document frequency (tf.idf)
23	Decision Tree	Term frequency inverse document frequency (tf.idf)
24	PART	Term frequency inverse document frequency (tf.idf)

5.2.5 Results and Discussion

The results of the application of the aforementioned research tasks are presented in tables 5.2, figures 5.3 and 5.4. The following is closer examination of these results. As can be noted from table 5.2, all models have an improved performance over base line (50%). Comparing all developed models yields the followings:

- Among the developed 1st degree polynomial SVM models, the highest prediction accuracy of 76% was achieved by using atf and tf.idf weighting schemes. An increase of 12% and 5% over tf and ltf schemes was attained, respectively.
- Among the developed 2nd degree polynomial SVM models, the highest prediction accuracy of 72% was achieved by using atf and tf.idf weighting schemes. An increase of 8% and 3% over tf and ltf schemes was obtained, respectively.
- Among the developed 3rd degree polynomial SVM models, the highest prediction accuracy of 74% was achieved by using tf.idf weighting schemes. An increase of 14%, 3%, and 2% over tf, ltf, and atf schemes was obtained, respectively.
- Among the developed Naïve Bayes models, the highest prediction accuracy of 84% was achieved by using tf.idf weighting schemes. An increase of 11%, 2%, and 61% over tf, ltf, and atf schemes was obtained, respectively.

Table 5.2 Accuracy and Kappa Measures of Developed Models

ACCURACY						
Weighting Scheme	Classifier					
	1st Poly. Deg. SVM	2nd Poly. Deg. SVM	3rd Poly. Deg. SVM	Naïve Bayes	Decision Tree	PART
tf	64%	64%	60%	73%	54%	52%
ltf	71%	69%	71%	82%	54%	52%
atf	76%	72%	72%	23%	54%	52%
tf.idf	76%	72%	74%	84%	54%	52%
KAPPA						
Weighting Scheme	Classifier					
	1st Poly. Deg. SVM	2nd Poly. Deg. SVM	3rd Poly. Deg. SVM	Naïve Bayes	Decision Tree	PART
tf	0.608	0.608	0.583	0.799	0.519	0.5
ltf	0.753	0.774	0.784	0.806	0.519	0.5
atf	0.806	0.795	0.795	0.186	0.519	0.5
tf.idf	0.806	0.795	0.8	0.827	0.519	0.5

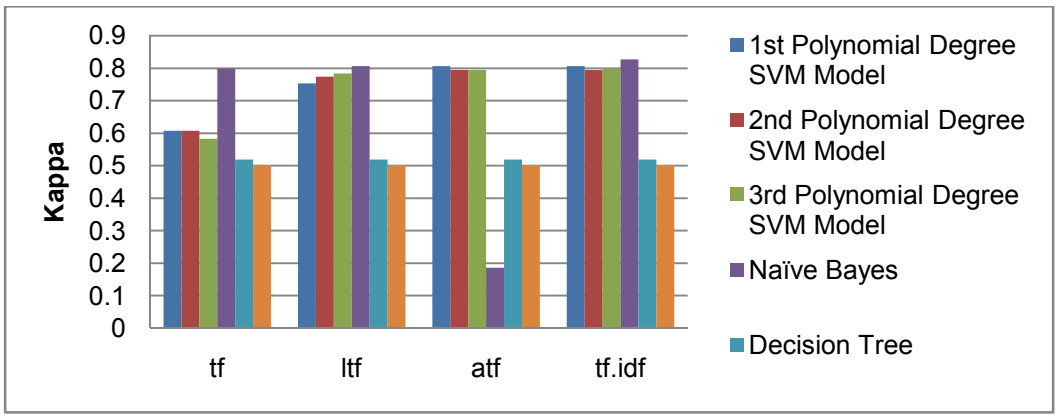


Figure 5.3 Kappa Measure of Developed Models

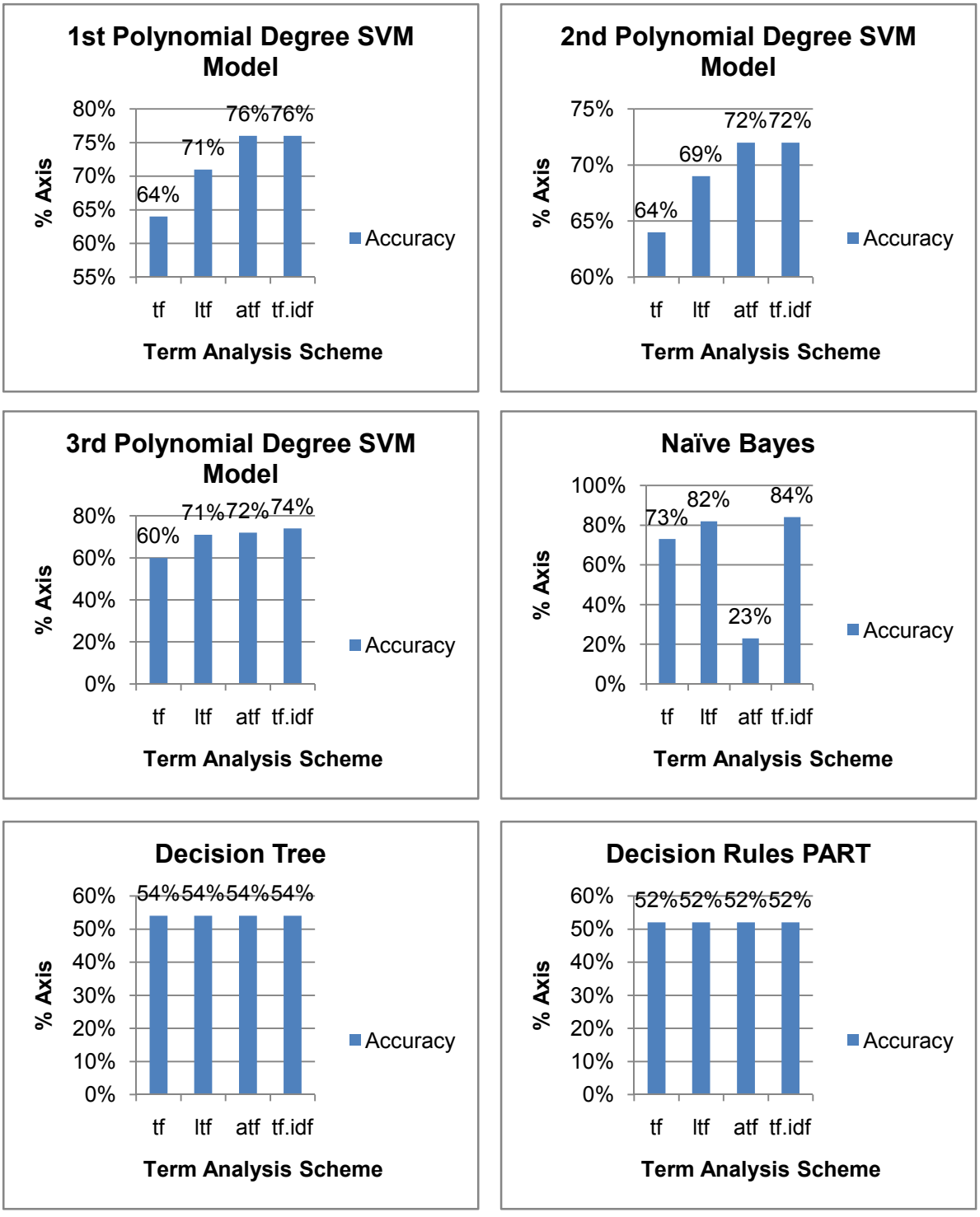


Figure 5.4 Accuracy Measure of Developed Models

- All developed models using Decision Tree and PART classifiers attained a prediction precision of 54% and 52%, respectively. This outcome is predictable since classification is based on induction rules. Consequently, varying the weighting scheme does not affect the derived rules.

It is clear from the above discussion, tf.idf weighting mechanism achieved better performance than tf, ltf, and atf. This could be attributed to that fact that raw term frequency (tf) representation suffers from a critical problem. That is all parsed terms in the corpus are considered equally important when it comes to assessing their relevance to a query. However, some terms, like highly occurring ones over all cases in the corpus, have no discriminating power in legal knowledge extraction. For example, a legal term like “contract” exists in almost all cases in our corpus. Consequently, this term holds no power to differentiate the existence of significant legal factors. This problem is slightly resolved by modifying the raw term frequency with the logarithmic and augmented weighting mechanisms. However, the term frequency inverse document frequency (tf.idf) mitigates this problem by scaling down the term weight of terms with a high frequency of occurrence. This is done by weighting a term frequency with respect to its occurrence in all cases within the corpus. In this case, discrimination between cases is done through the case-level statistic (such as the number of documents containing a term), which is considered to be of higher power than to use a cases-wide statistic. Consequently, the importance of terms increase proportionally to the number of times a term appears in the case but is offset by the frequency of that term in the corpus which leads to a suitable weighting mechanism for the purpose of the current research study.

5.2.6 Model Validation

As can be noted from the above discussion, table 5.3, and figures 5.5 and 5.5, the highest prediction accuracy rate of 84% was attained using Naïve Bayes model while implementing the tf.idf weighting scheme. Contrary to the findings of chapter 4, the performance of the SVM models was not found to be the highest. This can be attributed to the fact that SVM models implement active learning features as detailed in chapter 4. The performance of this feature deteriorates as the ratio of the number of cases to the number of features utilized to train the models decreases. In our case, the models are trained using 120 cases with respect to 2354 features. This increased number of features, however led to the enhanced performance of the NB model compared to the other models. As mentioned in chapter 4, the limited number of features restricted the performance of the NB automated litigation outcome prediction models.

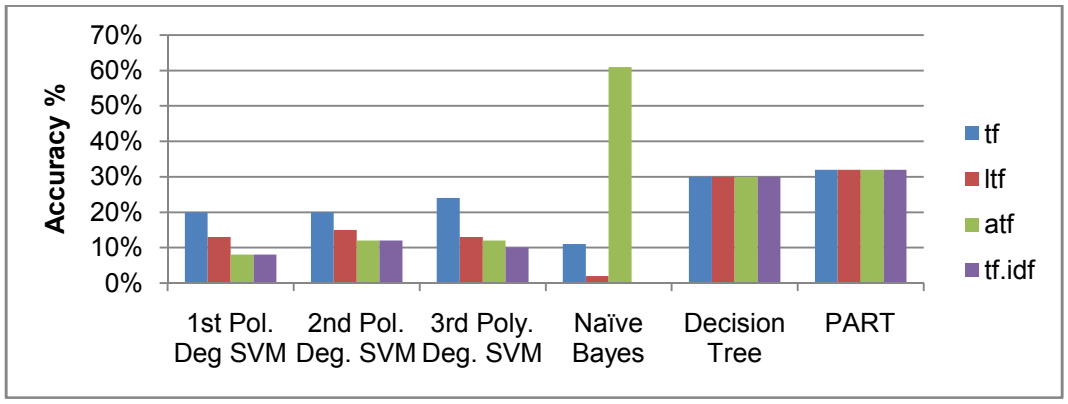


Figure 5.5 Accuracy Increase of Naive Bayes over Developed Models

Table 5.3 Accuracy and Kappa Increase of Naive Bayes over Developed Models

ACCURACY						
Weighting Scheme	Classifier					
	1st Poly. Deg. SVM	2nd Poly. Deg. SVM	3rd Poly. Deg. SVM	Naïve Bayes	Decision Tree	PART
tf	20%	20%	24%	11%	30%	32%
ltf	13%	15%	13%	2%	30%	32%
atf	8%	12%	12%	61%	30%	32%
tf.idf	8%	12%	10%	-	30%	32%
KAPPA						
Weighting Scheme	Classifier					
	1st Poly. Deg. SVM	2nd Poly. Deg. SVM	3rd Poly. Deg. SVM	Naïve Bayes	Decision Tree	PART
tf	0.219	0.219	0.244	0.028	0.308	0.33
ltf	0.074	0.053	0.043	0.021	0.308	0.33
atf	0.021	0.032	0.032	0.641	0.308	0.33
tf.idf	0.021	0.032	0.027	-	0.308	0.33

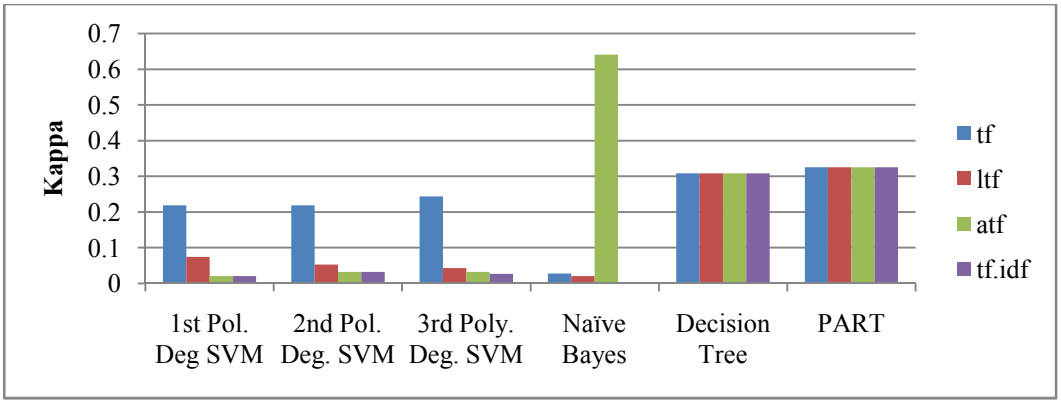


Figure 5.6 Kappa Increase of Naive Bayes Over Developed Models

Further validation of the best developed model (Naïve Bayes) was performed by testing a newly un-encountered dataset. Twenty two (22) arbitrary cases from the corpus created in chapter 3 were chosen for validation purposes. The following steps define the methodology adopted for validating the Naïve Bayes model.

1. Twenty two (22) cases were randomly chosen from the developed corpus in chapter 3.
2. Manual extraction of significant legal factors was performed.
3. A program in C++ was developed to perform data cleaning, data integration, and data reduction on the textual representation of the 22 cases. A copy of the developed program is provided in Appendix F.
4. Prediction of significant legal factors for each case was done by running the tf.idf term frequencies for the 22 cases through the Naïve Bayes model.
5. Validation of the model was based on prediction accuracy. Only cases that were predicted correctly through all related significant legal factors were considered as a true prediction. Accuracy was measured as the ratio of true predicted cases to total tested cases.

Table 5.4 and figure 5.6 illustrate the outcomes of the aforementioned methodology. As can be noted from table 5.4, 18 cases out of the 22 were predicted accurately leading to a prediction precision of 81.8%. In addition, among the 4 falsely predicted cases (1) 2 cases had an error in predicting 1 significant legal factor (case numbers 11 and 15); (2) 1 case had an additional predicted significant legal factor (case number 19); (3) 1 case had a missing significant legal factor (case number 20). These results further assure the suitability of the model to extract legal factors from the case corpus.

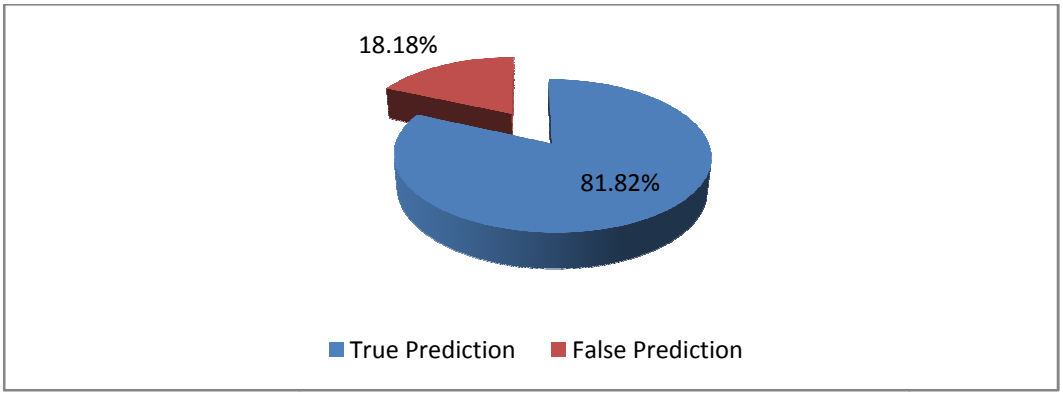


Figure 5.7 True and False Prediction Analysis of Best Model

Table 5.4 Prediction Analysis of 22 Newly Introduced Cases

Document #	Predicted Legal Factors	Manually Extracted Legal Factors	True/ False Prediction
1	2,3,7,9,10,11	2,3,7,9,10,11	TRUE
2	2,3,4,5,6,7,10,11	2,3,4,5,6,7,10,11	TRUE
3	2,3,4,5,6,7,10,11	2,3,4,5,6,7,10,11	TRUE
4	1,2,4,7,8,10,12	1,2,4,7,8,10,12	TRUE
5	3,4,9,10,13	3,4,9,10,13	TRUE
6	3,4,9,10,13	3,4,9,10,13	TRUE
7	1,4,9,10,11,12	1,4,9,10,11,12	TRUE
8	1,4,9,10,11,12	1,2,4,9,10,11,12	TRUE
9	2,4,5,8,10,11,12	2,4,5,8,10,11,12	TRUE
10	2,4,5,8,10,11,12	1,2,4,5,7,8,10,11,12	TRUE
11	2,4,7,8,10,12	2,4,7,8,10,11	FALSE
12	2,4,7,9,10,11	2,4,7,9,10,11	TRUE
13	1,3,4,10	1,3,4,10	TRUE
14	1,3,4,10	1,3,4,10	TRUE
15	1,2,7,9	1,2,7,10	FALSE
16	1,3,4,7,10	1,3,4,7,10	TRUE
17	1,3,4,7,9,13	1,3,4,7,9,13	TRUE
18	1,3,4	1,3,4	TRUE
19	1,3,4,10	1,3,4	FALSE
20	1,2,3,4,5	1,2,3,4,5,10	FALSE
21	1,4,10	1,4,10	TRUE
22	1,10	1,10	TRUE

5.3 Chapter Summary

The objective of this chapter was to automate significant legal factors identification within textual representations of DSC cases. To that end, SVM, Naïve Bayes, and Induction Rule classifiers were adopted for the study, and 24 models were developed. Four weighting schemes namely tf, ltf, atf, and tf.idf were implemented for each type of ML algorithm. The highest prediction rate of 84% was attained by Naïve Bayes classifier while implementing tf.idf weighting. The model was further validated by testing 22 newly un-encountered cases. A prediction precision of 81.8% was attained. From the above, it could be concluded that NB model with a tf.idf weighting mechanism is the most suitable ML algorithm to automate the extraction of legal factors from a large corpus of DSC cases.

CHAPTER 6

AUTOMATED EXTRACTION OF PRECEDENT DSC CASES

6.1 Introduction

Among the goals of the current research is to minimize time and cost associated with the need of legal experts in the construction industry for analysis and determination of the appropriate resolution mechanisms to be adopted in case of dispute. As illustrated earlier in chapter 2, the legal system in the United States of America is Anglo Saxon, and a corner stone of which is precedence. Precedence can be defined as utilizing verdicts from previous similar cases to decide on newly encountered ones. Although the work covered in chapters 3, 4, and 5 under the current research helps to alleviate the drawbacks of litigation and the need for experts in the construction industry. A final step is still needed to complete the contribution of the current research towards solving these problems. Through the earlier work performed under the current research, a party to a dispute can accurately determine the odds of winning or losing a case in court using ML. Consequently, they can decide on the appropriate dispute resolution strategy they should follow. If they decide to resolve the current dispute through litigation, having supporting documents and precedent cases of similar nature is a necessity. Consequently, the primary objective of this chapter is to develop an automated precedent case extraction model for DSC disputes in the construction industry. This model will extract precedents from large corpi based on similarity to newly un-encountered DSC cases using machine learning and NLP techniques. The research

tasks explained in this chapter include: (1) investigating Latent Semantic Analysis (LSA) algorithms; (2) developing reduced feature spaces; (3) developing LSA automated extraction models; and (5) testing and validating the developed models. Therefore, this chapter will start with an account of the features space selection process for the implementation of LSA algorithm.

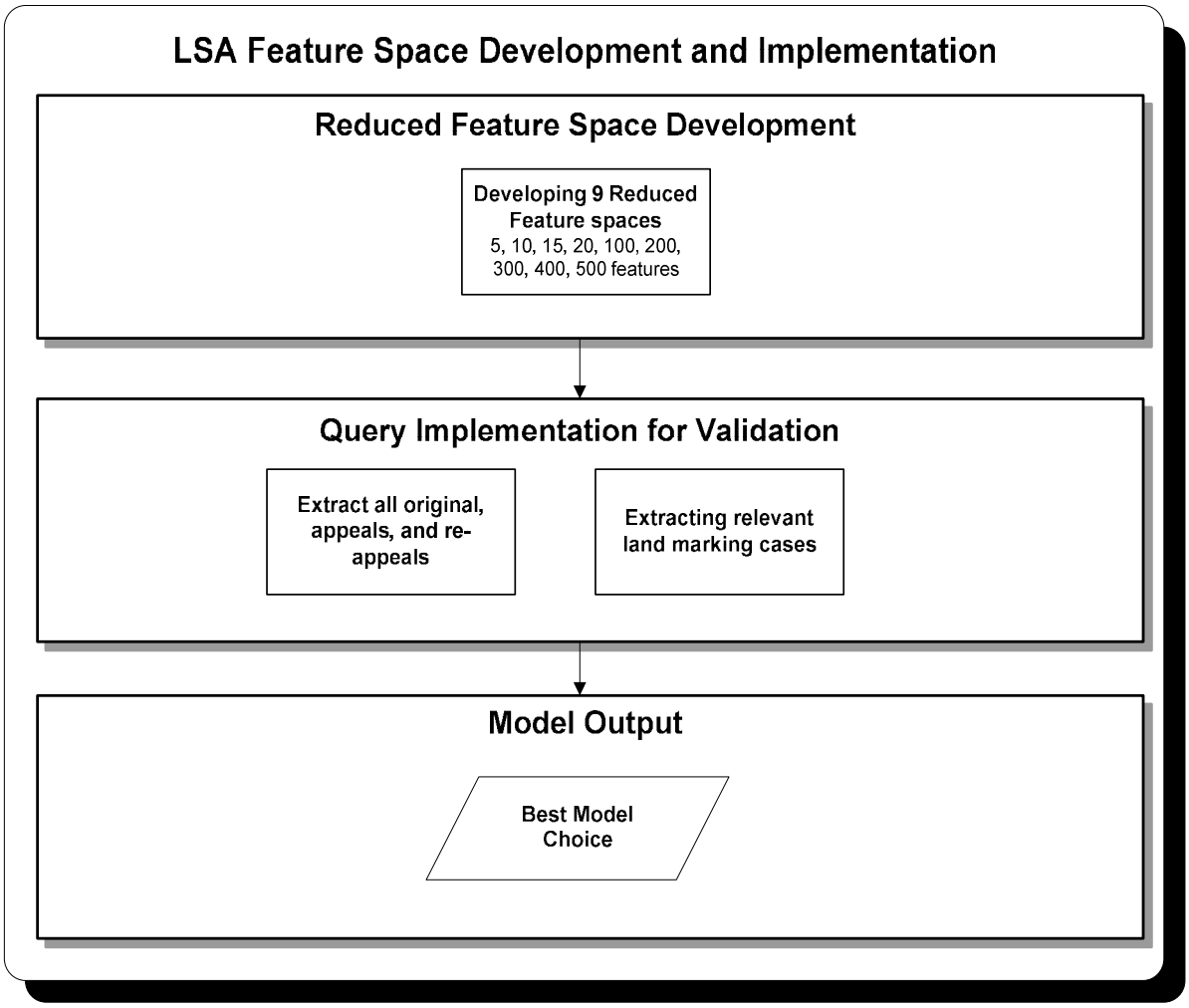


Figure 6.1 Research Tasks for Automated Precedent DSC Cases Extraction from Large Corpus

6.2 LSA Feature Space Development

In this research task, an important parameter for the LSA model implementation is determined. Feature space in LSA is defined by the number of feature (in this research features are words) that are used to represent a case as a vector. Research concerned with LSA feature space development covers a wide range of reduced feature space sizes that enhance the effectiveness of the algorithm. It was highlighted that for dispersed dataset a large feature space sizes ranging between 100 and 500 are appropriate (Choi et al. 2001). However, for closely related dataset, a feature space as small as 7 is appropriate (Koll 1979). Consequently, the present research task developed 9 different reduced feature spaces utilizing 5, 10, 15, 20, 100, 200, 300, 400, 500 features. These different feature spaces will be used in the testing and validation of the developed models in two ways. The first form of testing will evaluate the LSA algorithm's ability to extract all original, appeals, and re-appeals of a case using the developed feature spaces. This extraction will take place in a new corpus of 450 cases from the Federal Court of New York that were filled between 1919 and 2007. The feature space performance will be judges against an extraction similarity measure of 1. The second form of evaluation is based on correctly extracting land marking cases that were mentioned by a judge to be relevant within the body of cases from the expanded corpus of 450 cases while correctly rejecting others that were mentioned to be irrelevant.

6.3 Model Design and Implementation

The following is a description of the steps of the LSA algorithm implemented for precedent case extraction. The algorithm starts with an argument, filename, which is the name of the file or directory to be parsed. If filename is actually a directory, the algorithm traverses this directory and all subdirectories in a recursive fashion and parses each regular file it encounters (i.e. no symbolic links are parsed) (GTP 2008). If filename is a single file, the algorithm simply parses it only. The algorithm moves sequentially through each file, extracting keys comprised of relevant characters, and ignoring keys contained in the common word list specified in the input common word file. By default, only keys that begin with characters A-z will be parsed. Keys beginning with a digit (0-9), with the exception of numbers that could be interpreted as dates in the 1700's 1800's and 1900's, will be ignored to the next whitespace character. The algorithm converts all characters to lower case, and requires that a key contain at least 2 characters and no more than 20 characters. Single character keys are ignored and keys with more than 20 characters are truncated and all characters to the next unrecognized character or whitespace are ignored.

After tokenizing the keys and associating each one with the document it was extracted from, the algorithm begins calculating term weights. The (global) weights of the terms are computed over the collection of documents. By default, only a local weight is assigned and this is simply the frequency with which the term appears in a document. Two thresholds exist for term frequencies: Global and local (GTP 2008). By default, the global and local thresholds are both 1. A term must appear more than

1 time in the entire collection and in more than one document in the collection before it will be weighted. These thresholds can however be changed. Each term is assigned a term ID number (starting with 1, in alphanumeric order). Next, the local weights of the keys are computed. Each term weight is the product of a local weight times a global weight (if specified). Entries are grouped by document number. Next, the algorithm creates a term-by-document matrix using the Harwell-Boeing sparse matrix format. The algorithm finally performs SVD decomposition before cleaning all temporary files and writing a summary of the run.

General Text Parser (GTP) windows version, developed by Stefen Howard, Haibin Tang, Dian Martin, Justin Giles, Kevin Heinrich, Barry Britt, and Michael W. Berry, was utilized for the implementation of LSA feature spaces development and validation described above. GTP is a general purpose text parser with matrix decomposition option which can be used for generating vector space IR models. As stated by Landauer et al. (2007) “GTP could be considered the reference program for LSA analysis because it is a rewrite of the older Telcordia suite in more modern way. It is a very large program. Contrary to what its name indicates, GTP is not only a parser: It actually can run an SVD at the end of the process. GTP is a 100% C++ code”.

As mentioned earlier in section (6.2) 9 reduced feature spaces were generated. Each reduced feature space was generated with a local threshold of Log function and a global threshold of entropy function. The Log function (equation 6.1) decreases the effect of large differences in term frequencies (Landauer et al 2007). The entropy function (equation 6.2), on the other hand, assigns lower weights to

words repeated frequently over the entire document collection, as well as taking into consideration the distribution of each word frequency over the documents (Landauer et al. 2007). These thresholds were adopted for the current analysis due to their success over other types of threshold combinations. Dumais (1991) illustrated that the log-entropy threshold combination attained 40% higher retrieval precision over other threshold combinations.

$$ltf_{i,d}=1+\log(tf_{i,d})\,;\ tf_{i,d}>0 \qquad 6.1$$

$$\sum_i \frac{P_{ij}\log_2(P_{ij})}{\log_2 n} \text{ where } P_{ij}=\frac{tf_{ij}}{gf_i} \qquad 6.2$$

Where tf_{ij} is the word frequency of word *i* in document *j*, and gf_i is the total number of times that the word *i* appears in the entire collection of n documents.

6.4 Results and Discussion

As mentioned earlier in section 6.3, the testing and validation of the developed models is twofold. The first fold is through the successful extraction of a query case, its appeals, and re-appeals from the corpus with the highest similarity measure. To that end, each of the generated reduced feature spaces was tested with three query cases. Table 6.1 illustrates the similarities by which each space retrieved the related documents. As can be seen from table 6.1, reduced feature spaces with sizes between 5 and 20 retrieved the required documents with a similarity measure of 1. As the feature space increases in size, data becomes dispersed and the similarity decreases. Despite that fact, all feature spaces were able to retrieve all related documents of the three query cases. The highest

similarities of 1 were attained using the lower range of feature; whereas, the lowest of 0.999936 was attained using the 500 reduced feature space size.

Table 6.1 Similarity Measure of Similar Case Retrieval

Reduced Feature Space Size	Similarity Value			Average Similarity
	Case 1	Case 2	Case 3	
5	1	1	1	1
10	1	1	1	1
15	1	1	1	1
20	1	1	1	1
100	0.999996	0.999997	0.999996	0.999996333
200	0.999978	0.999983	0.999979	0.99998
300	0.999949	0.999948	0.999948	0.999948333
400	0.999936	0.999936	0.999936	0.999936
500	0.999936	0.999936	0.999936	0.999936

The second fold was based on the ability of a reduced feature space to extract related supporting cases from the corpus and rejecting unrelated ones. Confidence on whether a case is related or unrelated to a query one is based on the opinion of judges illustrated within the textual body of a case. For example, judges Greenblott, J. Koreman, P. J., Sweeney, Main and Larkin, JJ., included in their opinion in the case of Public Constructors, Inc., Respondent, v. State of New York, Appellant (1977) the following: “In a construction contract between the State and an individual, which contains representations as to existing conditions affecting work there under as well as an exculpatory clause relieving the State of liability and requiring personal inspection of the contract site, liability, nevertheless, may attach to the State if said conditions are not as represented and (1) inspection would have been unavailing to reveal the incorrectness of the representations (Foundation Co. v State of New York, 233 N. Y. 177, 184-185; Faber v. City of New York, 222 N. Y.

255, 260), or (2) the representations were made in bad faith (Young Fehlhaber Pile Co. v. State of New York, 265 App. Div. 61; Jackson v. State of New York, 210 App. Div. 115, affd. 241 N. Y. 563). In our view, the Court of Claims was clearly correct in finding that the contract documents furnished to the bidders contained misrepresentations, and in rejecting the State's contention that claimant must bear the responsibility as the result of an inadequate prebid investigation". It could be concluded from this opinion that cases like Foundation Co. v State of New York, 233 N. Y. 177, 184-185; Faber v. City of New York, 222 N. Y. 255, 260); Young Fehlhaber Pile Co. v. State of New York, 265 App. Div. 61; and Jackson v. State of New York, 210 App. Div. 115, affd. 241 N. Y. 563 are related to the case of Public Constructors, Inc., Respondent, v. State of New York, Appellant and can be used as supporting precedent cases.

To that end, each of the developed reduced feature space sizes was tested against two query cases. The first included three relevant cases and three irrelevant ones. The second included three relevant cases and two irrelevant ones. A value of 0.75 was maintained as a threshold for retrieval. Consequently, if a case was retrieved as a lower similarity, it was disregarded. Table 6.2 illustrates the similarity measures by which each reduced feature space retrieved the relevant and irrelevant documents. Average similarities are reported for each case and overall similarities are reported as the average of retrieval over the two query cases for each reduced feature space size. It can be noted from table 6.2 that feature spaces with sizes beyond 100 were not able to retrieve any of the related documents. This is due to the increased disparity that is generated in the feature space due to the increased

dimensions. Such outcomes support the findings of the literature review under section 2.6.4.

Table 6.2 Similarity Measures by Which Each Reduced Feature Space Retrieved the Relevant and Irrelevant Documents

Reduced Space Size	Case	Similarity Values		Average Similarity per Case	Overall Average Similarity
		Relevant Cases	Irrelevant Cases		
5	1	0.999995 0.885543 0.82938 0.999995	0.778905 <0.75 <0.75 <0.75	0.904972667	0.9203155
	2	0.999923 0.807057	<0.75	0.935658333	
10	1	0.996806 0.993939 0.801871 0.999997	<0.75 <0.75 <0.75 <0.75	0.930872	0.955640833
	2	0.981382 0.95985	<0.75	0.980409667	
15	1	0.991001 0.802091 0.778041 0.95951	0.761948 0.757547 <0.75 0.795354	0.857044333	0.855562833
	2	0.836486 0.766248	<0.75	0.854081333	
20	1	0.976275 0.87631 0.790509 0.885414	0.794503 0.791911 0.776489 0.80643	0.881031333	0.868614167
	2	0.85861 0.824567	<0.75	0.856197	
100	1	0.809697 0 0 0	0 0 0 0	0.269899	0.1349495
	2	0 0	0	0	

Table 6.2 (Continued)

Reduced Space Size	Case	Similarity Values		Average Similarity per Case	Overall Average Similarity
		Relevant Cases	Irrelevant Cases		
200	1	0	0	0	0
		0	0		
		0	0		
	2	0	0	0	
		0	0		
		0	0		
	1	0	0	0	
		0	0		
		0	0		
	2	0	0	0	
		0	0		
400	1	0	0	0	0
		0	0		
		0	0		
	2	0	0	0	
		0	0		
		0	0		
	1	0	0	0	
		0	0		
		0	0		
	2	0	0	0	
		0	0		
500	1	0	0	0	0
		0	0		
		0	0		
	2	0	0	0	
		0	0		
		0	0		

Furthermore, the highest overall average similarity of 0.955640833 was attained using a reduced feature space size of 10 features; whereas, the lowest of 0.1349495 was attained using a reduced feature space of 100 features. The superiority of the 10 feature reduced space was further demonstrated by not retrieving any of the irrelevant cases with a similarity above the threshold of 0.75. Figure 6.2 shows the advancement of the 10 feature reduced space size over other

developed spaces. It attained an increase of 3.84%, 11.7%, 10.02%, and 608.15% over the 5, 15, 20, and 100 reduced feature spaces respectively.

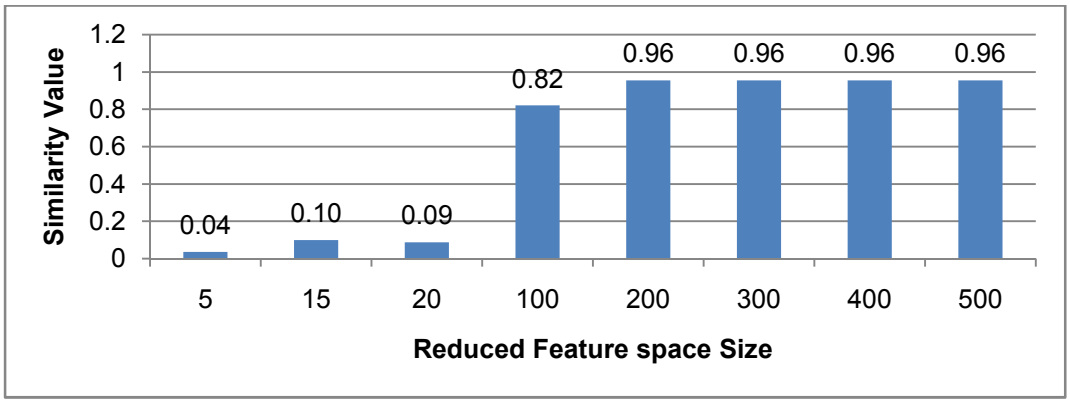


Figure 6.2 Advancement of 10 Feature Space Over Other Reduced Feature Spaces

6.5 Chapter Summary and Conclusion

The objective of this chapter was to automate the extraction of related DSC cases from large corpus to newly introduced ones. The chapter, therefore, implemented two main stages that: (1) implemented Latent semantic Analysis for the development of 9 reduced feature spaces representation of the gathered corpus; and (2) testing and validated the developed reduced feature spaces through twofold validation methodology. The main findings from the implementation of this two stage research methodology include:

1. Low dimensioned reduced feature spaces are more representative to domain problems analysis closely related document collection. A finding that supports outcomes achieved by earlier researchers as illustrated in the literature of the LSA domain.

2. Higher dimensioned reduced feature spaces are more representative to domain problems analysis in dispersed and unrelated document collections.
3. The highest similarity measure (equal to 1) with respect to retrieving initial case, appeals, and re-appeals was achieved using the 5, 10, 15, and 20 reduced feature spaces.
4. The highest overall similarity measure of 0.955640833 with respect to retrieving relevant cases as supporting documents was achieved using the 10 reduced feature space.
5. The lowest similarity measure of 0.999936 with respect to retrieving initial case, appeals, and re-appeals was achieved using the 400 and 500 reduced feature spaces.
6. The lowest overall similarity measure of 0 with respect to retrieving relevant cases as supporting documents was achieved using the 200, 300, 400, and 500 reduced feature spaces.

From the above it could be concluded that LSA reduced feature space of 10 features is the best to be adopted automating the extraction of similar DSC cases from large corpus to newly introduced one. The finding in this chapter are anticipated to decrease time consumed and overwhelming experts' fees related to analysis and extracting of relevant DSC cases. It is also anticipated that the benefits of these findings will not only help the construction industry, but will also extend to the legal domain.

CHAPTER 7

OVERALL SYSTEM PERFORMANCE EVALUATION

7.1 Introduction

In spite of the enhanced performance of the ML models developed at each task of the current research, an overall evaluation of the system performance as one package is much needed to understand its impact on construction legal decision support. Consequently, the objective of this chapter is to evaluate the aggregated errors of the developed models as a full system in an endeavor to assess its robustness and effectiveness in legal decision support. To this end, 5 arbitrary cases from the gathered corpus in task 2, not used earlier for training and testing of the developed ML models, were utilized to evaluate the overall performance of the system. Therefore, this chapter will provide: (1) a breif description of the utilized cases and an identification of the significant legal factors present in each case; (2) an evaluating the overall performance of the system through aggregated errors; and (3) a description of the areas of weaknesses and proposing enhancement methodologies. Therefore, this chapter will start with a brief account of the cases selection process for the implementation of the developed ML models.

7.2 Test Case Selection

This section of the chapter provides a brief description of case history and the Legal Factors pertinent to each case among the chosen ones for evaluation. Table 7.1 illustrates the legal factors existing in each case.

7.2.1 Case 1: Horgan, v. The City of New York

On the 14th day of January, 1893, William G. Horgan entered into a contract in writing with the state of New York to furnish and provide all the necessary materials and labor, and excavate, remove and dispose of all silt, sediment and other materials deposited in the bottom of "The Pond" near Fifty-ninth street and Fifth and Sixth avenues in the city of New York, and construct a concrete bottom over the pond. This contract was to be completed by June 1st, 1893. The contract document included an estimate prepared by the city engineers for the value of the works to be performed. It also included a statement clearly stating that by signing this contract, the contractor had familiarized and satisfied himself by personal examination of the accuracy of the engineers' estimates, and indemnified them from later complaining related to it. The contract provided that the contractor was to bear any damage from unforeseen obstructions in the work. Though there was an outlet pipe at the bottom of the lake, it stopped draining when the water level was 14 inches due to an obstruction in the sewer that it drained into. Thus, the contractor had to pump out the remaining water and sought extra money for this. Agreeing with the contractor that his pumping was beyond the terms of the contract, the court ruled that because the city's negligence in failing to properly maintain the outlet pipe increased his cost in doing the work, it was liable to reimburse him for the extra costs. Consequently, the Court reversed both the appellate and trial court judgment and ordered that the contractor receive reimbursement for the extra costs.

7.2.2 Case 2: Iacobelli Construction, Inc., v. County of Monroe, Rochester Pure Waters District, and Calocerinos & Spina Consulting Engineers, P.C.

Defendants, a consulting engineer and a county, that published bid documents for construction of a tunnel for storing and transporting sewage. The bid documents provided technical information about the site conditions. Plaintiff contractor was awarded the project. The construction contract contained a differing site conditions clause. However, during construction, the contractor encountered site conditions that differed materially from those anticipated from the bid documents, and submitted a claim for reimbursement. The claim was denied by the Defendants. The district court granted summary judgment in favor of defendants because plaintiff's expert's affidavits provided only opinions regarding the bid specifications. Plaintiff claimed on appeal that the affidavits established a factual question as to whether the bid specifications provided accurate information of the site conditions. The court agreed with plaintiff's claims, and reversed the district court's judgment.

7.2.3 Case 3: Piper, Inc., v. New York State Thruway Authority

On February 9, 1953 Piper Inc. entered into a contract with the New York State Thruway Authority for the construction of a portion of the Thruway, from Ontario Section, District 5, Subdivision 15, Ransom Road to Genesee County line in the County of Erie. Piper Inc appointed A. L. Dougherty Company as a subcontractor to perform certain items of its scope of work under this contract. The contract documents included the following section: “Sub-surface explorations have

been made within the limits of the proposed work. Interested parties may review the records of these explorations at the office of the District Engineer in Buffalo, New York… The information contained in the foregoing paragraph is offered in good faith by the Department and reflects the opinions of the Department engineers relative to the sub-surface conditions. The Contractor's attention, however, is called to the fact that the information obtained there from is not to be substituted for personal investigation and research by the Contractor as required by Article three of the Contract Agreement." The claim under this case was raised due to a sand stratum that was encountered while excavation works and the contractor claimed it did not discover the sand until after bidding, and that caused an increase in his costs to attain and procure sufficient gravel from another location. The contractor also claimed it was misled as to the amount of material available to meet gravel requirements. However, documents examined by the court illustrated that the subcontractor's agents visited the site before bidding, examined all sub-surface exploration reports, and familiarized themselves with the site conditions. In addition, witnesses testified that a sand stratum was readily observable at the site and the sand was not suitable for the contract's gravel requirements. The court denied the contractors' case because the he should have been aware of the situation before bidding, and no representation was made that the material found at the site would meet all gravel requirements.

7.2.4 Case 4: Fruin-colnon Corporation, Traylor Bros., Inc. and Onyx Construction & Equipment, Inc., A Joint Venture, v. Niagara Frontier Transportation Authority

The contract called for plaintiff to excavate and construct twin subway tunnels, each approximately two miles long, as part of the Buffalo light rail rapid transit system. During construction, the plaintiff raised a claim seeking reimbursement of $ 3,255,150 under DSC associated with extra work and delays incurred as a result of the unforeseen need to use steel ribbing for temporary support of the tunnel during excavation, must have been solely attributable to such materially different subsurface conditions. After factual extraction, the court found that the contract documents included the followings:

(1) “The Engineering Design Rationale (EDR) indicated that rock quality generally would be "average to good" for tunneling, but warned of the existence of localized areas of poor rock quality, opened, weathered, and in some cases "solutioned" fractures, and intersecting vertical and horizontal joints.” (Fruin 1992).

(2) “The Tunnel Interpretive Report (TIR) indicated the existence of intersecting joints forming blocks. The TIR explicitly cautioned that such blocks might not be self-supporting as excavation progressed and, depending on local conditions, might require varying levels of primary support. Like the EDR, the TIR specifically warned of areas of relatively permeable and solutionized rock and open, intersecting, and water bearing joints.” (Fruin 1992).

The court found that actual site conditions did not differ materially from conditions indicated in the contract, which put plaintiff on notice of the possibility of those

conditions. Consequently; the court affirmed the rulings denying plaintiff tunnel builder's claim for additional compensation for work related to DSC from defendant transit authority.

7.2.5 Case 5: The Foundation Company, v. The State of New York

The contractor entered into a unit price contract with the state to build a dam and lock across the Mohawk River at Scotia with a canal lock at its south end. The dam was designed by the State Engineers. The State stated that the bed of the stream constitutes of a gravel layer upon which the dam cannot be constructed. It also illustrated that cofferdams can be used. Consequently, it was determined to sink caissons under compressed air to bedrock for the whole distance. The final result would be a solid concrete cut-off wall, on top of which would be placed the other structures necessary to complete the dam. The contract documents included illustration that the bedrock "rock or boulders" upon which the caissons would rest is to be found not lower than level 148. During construction, the contractor raised a claim seeking reimbursement under DSC associated with extra work incurred as a result of the unforeseen need to excavate for deeper than level 148 to reach bedrock with appropriate properties for the current project. The court denied the contractors claim, finding that there was no bad faith and an honest mistake on the part of the state officers. The court held that where the representations of a contract for specifications were made in good faith, the contractor assumed the risk of their accuracy.

7.2.6 Case 6: Charles Sundstrom et al., v. The State of New York

The contractors had an agreement with the state to build a section of the barge canal. During construction, the contractor filed a claim under DSC to be reimbursed for extra cost incurred due to the negligence of the State. The court found that these costs could have been eliminated if the State maintained the canal in appropriate manner, which is part of the State's liability. In the contractors' action for damages, the board of claims held that the state was liable for the loss to the contracts from the damages from the canal on the basis that the contractors could not have determined that the canal was defective. The board found that the damage was due to the lack of repair of the canal. However, the appellate division reversed, finding that the defects could have been discovered by the experienced contractor. Consequently, the contractor appealed the case and the court reversed the verdict. It found that the state was liable for damages from their failure to adequately maintain the canal.

7.2.7 Case 7: James F. Leary and Thomas J. Morrison, v. The State of New York, City of Watervliet

On June 20 1913, James F. Leary and Thomas J. Morrison entered into a contract with the State of New York to perform a storm sewer system in the city of Watervliet. While construction, the contractor filed a claim under DSC to be reimbursed against extra costs the he incurred due to encountering underground rock formation that was not mentioned in the contract documents. The contractors illustrated that the extra work completed because of an underground rock formation

merited the payment of additional compensation for the reasonable value of their excavation work, and that they were also entitled to additional payments because of the city's failure to remove obstructions after having been property notified by the contractors of the existing conditions. “The court found in favor of the contractors in all regards based on its determination that (1) the contract provisions did not constitute a condition precedent to the contractors' recovery and the city failed to plead or prove the contractors' alleged non-compliance with the disputed contract provisions; and (2) the city benefitted from the value of the work completed by the contractors, who continued to do the work necessary that exceeded the anticipated costs in the absence of bad faith after notifying the city of existing conditions.” (Leary 1916).

7.2.8 Case 8: Tony Carfagno and Others, Copartners Doing Business under the Firm Name and Style of Carfagno & Dragonetti, v. The City of New York

The contractor entered into a contract with the city of New York to procure and install fire service system. While performing the work, the contractor discovered that the city made a mistake in calculating the length of pipes to be installed. The contractor raised a claim under DSC seeking compensation for the cost of the extra pipes procured and anticipating installation costs as well as the profit margin allocated to them. The city compensated the contractor for the cost of the extra pipes procured only. The court judged in favor of the city on the bases that there was nothing else the city could do to correct its innocent error. The court also relied on

the contract which had warned anyone bidding not to rely on the city's measurements and asked them to familiarize themselves with the site conditions and anticipated quantities. The court also found the contractor discovered the error and knew about it when they submitted their bid.

7.2.9 Case 9: S. Pearson & Son, Inc., v. The State of New York

The contractor entered into a contract with the State of New York to perform the barge canal contract 20-B. “The contract was for dredging a channel in the Mohawk river between Mindenville and Canajoharie, a length of ten and one-tenth miles, and the width of the channel or excavated prism was to be 200 feet, and the channel was to be excavated to a grade line fixed by the plans which would afford a flotation depth of water of at least 12 feet when water was maintained at designated pool elevations.” (Pearson 1920). While performing the work under the contract, the contractor raised a claim under DSC to be reimbursed for extra costs incurred due to the need for extra slope excavation, extra excavation below the grade line of a large number of boulders and a considerable quantity of rock, and the use of cobblestone protection works that were not disclosed in the contract documents. The court judged in favor of the state on the grounds that: (1) there was no misrepresentation in the contract documents or bad faith on the side of the state to conceal information; (2) the excavation works were performed outside of the excavation lines as shown on the plans, and it does not appear that the work was the subject of any alteration in the contract; and (3) the protection work was a necessity and should have been anticipated by the contractor.

7.2.10 Case 10: Christie v. United States

The contractor entered into a contract with the United States to perform a set of locks and dams on a river. After performing the works, the contractor filed a claim under DSC to be reimbursed for the extra expense incurred due to the increased difficulty in pile driving and excavation on account of state's misrepresentation of the materials to be penetrated and excavated. The court judged in favor of the contractor on the grounds that the specifications provided to the contractor were actually misleading, forcing the contractor to spend a substantial extra sum of money over and above their proposal and contract to perform the works.

Table 7.1 Legal Factors Pertinent to the Evaluation Set of Cases

Case #	TypeP	DSCC	N&C	Con raise	Com Impos	O change	M mistake	Year	O cause	Spec Warn	Spec Rep	CN Extra	O falsely	O Adjust	Outcome
1	4	0	1	1	0	0	1	11	1	0	1	1	0	0	Contractor
2	2	1	0	1	0	0	1	3	1	0	1	0	1	0	Contractor
3	3	0	1	1	0	0	0	11	0	1	1	0	0	1	Owner
4	3	1	0	1	0	0	0	4	0	1	1	1	1	0	Owner
5	4	0	0	1	0	0	1	11	0	1	1	1	0	0	Owner
6	4	0	0	1	0	0	1	11	1	0	1	1	0	0	Contractor
7	2	0	1	0	0	0	1	11	0	0	1	0	0	0	Contractor
8	2	0	1	0	0	0	1	11	0	0	1	1	0	1	Contractor
9	3	0	0	0	0	0	1	11	0	0	1	1	0	0	Owner
10	4	0	0	1	0	0	1	11	0	0	1	0	1	0	Contractor

7.3 System Performance Evaluation

This section of the chapter is describing the overall system performance evaluation. The evaluation is performed by analyzing the aggregated errors of the system as one package. Consequently, results will be reported at each step of the

system performance (i.e. automated legal factors extraction, automated prediction, and automated precedent cases extraction).

7.3.1 Significant Legal Factors Automated Extraction

The results of the application of the legal factor automated extraction model are presented in tables 7.2. The prediction accuracy was based on accurately predicting all factors pertinent to each case. Consequently, if one of the factors was not predicted accurately, the case is considered to be a false prediction. Examining the output of the model shows that the model attained an overall accuracy of 80%.

Table 7.2 Results of Automated Legal Factor Extraction Model

Case #	Prediction Accuracy
1	True
2	True
3	True
4	False
5	True
6	False
7	True
8	True
9	True
10	True

7.3.2 Litigation Outcome Automated Prediction

The results of the application of the automated litigation output prediction model are presented in tables 7.3. The prediction accuracy was based on accurately predicting the outcome of each case in comparison to actual verdict pertinent to each one. Examining the output of the model shows that the model attained an overall accuracy of 90%. The increase in the accuracy from the step of automated legal factor extraction was due to the fact that one of the cases that were falsely

predicted had only an error in one of the factors. It predicted that the contractor did not waive his right of extra compensation due to DSC. However, it accurately predicted that there was a warning in the specifications against the presence of DSC in the project. As illustrated in chapter 3, this factor had the highest increase on the prediction in favor of the owner.

Table 7.3 Results of Automated Litigation Prediction Model

Case #	Prediction Accuracy
1	True
2	True
3	True
4	True
5	True
6	False
7	True
8	True
9	True
10	True

7.3.3 Automated Precedent Case Extraction

The results of the application of the automated precedent case extraction model are presented in tables 7.4. The prediction accuracy was based on the average similarity measure by which relevant cases are extracted from the full corpus utilizing a feature space size of 10 features. To that end, case # 7 was excluded from the analysis, for the cases illustrated by the judge to be relevant were not part of the original corpus. Examining the output of the model shows that the model attained average retrieval similarities ranging between a lower end of 0.882 to a higher end of 0.976 with an overall average of 0.9217.

Table 7.4 Results of Automated Precedent Case Extraction Model

Case #	Prediction Accuracy
1	0.913
2	0.893
3	0.904
4	0.882
5	0.962
6	0.943
8	0.889
9	0.976
10	0.933

7.4 Chapter Summary and Conclusion

From the above, it could be deduced that the system attained an aggregated error of 10% since it achieved an overall accuracy of 90% after implementing the automated legal factor extraction and automated litigation prediction models. Furthermore, it attained an overall average similarity measure of 92.17%. Looking at the literature in the construction domain (chapter 2), it could be noticed that the performance of the developed system exceeded previously developed models by Arditi and Chau.

However, it is noticed that there has been a drop in the accuracy of the automated legal factor extraction model. This could be attributed to the features of the tested cases. Each new case might (1) include new features that are not included in the training of the model; and (2) exclude features that are included in the training of the model. This fact might affect the performance of the model. To enhance the performance of the model the followings are proposed: (1) tagging each legal factor with set of commonly used phrases by judges; (2) analyzing appropriate weights to be applied to different phrases; and (3) incorporating these phrases and

weights in the model development. These enhancements and others will, therefore, be the subject of future researches.

CHAPTER 8

CONCLUSION, CONTRIBUTIONS, AND FUTURE RESEARCH

8.1 Conclusion

The present research focused on developing a coherent and integrated methodology for construction legal decision support for Differing site Conditions (DSC) disputes through statistical modeling and machine learning (ML). The study developed a number of research products, including: (1) a set of statistically significant legal factors that governs verdicts related to DSC disputes in the construction industry; (2) an automated litigation prediction model for DSC disputes in the construction industry; (3) an automated extraction model for statistically significant legal factors from textual representations of DSC disputes; and (4) an automated retrieval model for supporting DSC cases from large corpus based on similarity measures to newly introduced ones.

First, a set of litigation cases related to DSC disputes in the construction industry was gathered and analyzed to define a comprehensive list of legal factors upon which judges base their verdicts. The analysis was pertinent to cases from the Federal Court of New York to standardize the jurisdiction and due to availability of a large number of cases related to the current study objectives. The initial analysis of cases which was based on detailed opinions of judges within the body of each case identified a set of 23 legal factors. Statistical models were developed to relate the likelihood of a DSC cases being judged in favor of one party over the other to the identified set of legal factors. Binary Probit and Logit Choice models were developed

in an endeavor to: (1) identify the effect of each extracted factor on the prediction of the winning party; (2) identify the best combination of factors with the highest significance on the prediction model; and (3) perform a sensitivity analysis to prioritize the most significant legal factors. Among the main findings of the aforementioned analysis are: (1) the developed Binary Probit Choice Model identified a set of 11 statistically significant legal factors with a prediction accuracy of 88.9%; whereas, the Binary Logit Choice Model identified a set of 9 statistically significant factors with a prediction accuracy of 93.3%; (2) generally, cases in which the Federal Government is a concerned party of dispute, judgments are in favor of the government (owner) over contractor; (3) the presence of "evident facts that the encountered conditions caused a change in the nature and cost of the contract" had the highest impact among variables causing a decrease in the prediction of judgment in favor of owner and caused an increase of 17.77% in prediction on favor of contractor; (4) the presence of "evident facts that the specifications included a warning against the presence of DSC from those conveyed in the contract documents" had the highest increases in the prediction of judgment in favor of owner and caused an increase of 56.56% in prediction in favor of owners. In addition, the development of Binary Probit and Logit choice models identified a joint set of 13 statistically significant legal factors related to DSC disputes in the construction industry. This set provided the grounds for the other three products of the current research.

Second, an automated machine learning DSC litigation outcome prediction model was developed. To that end, 120 DSC cases from The Federal Court of New

York that were filled in the period between 1912 and 2007 were utilized for the analysis. 10 machine learning models were developed namely 4 Support Vector Machines, 2 Naïve Bayes, and 4 Induction rule models. The highest prediction rate of 98% within the first category was attained by Kernel Polynomial 3rd degree model. Models developed under the second category yielded a highest rate of prediction of 93% attained by the Naïve Bayes model without implementing kernel estimators. A prediction rate of 97.8% was the highest attained within the third category by ADTree model with 15 boosts. Comparing the outputs of all developed models shows that they have achieved great advancements over the base line of 50% and previously performed researches. It could be concluded that SVM Kernel Polynomial 3rd degree model has achieved the best performance among all developed models.

Third, an automated machine learning significant legal factors extraction model was developed. The 120 cases, earlier utilized for the analysis of the previous task were utilized for the analysis under this task. Support Vector Machines, Naïve Bayes, and Rule Induction classifiers were also adopted for the study. 24 models were developed in which 4 weighting schemes namely tf, ltf, atf, and tf.idf were implemented for each type of classifier. The highest prediction rate of 84% was attained by Naïve Bayes classifier while implementing tf.idf weighting. The model was further validated by testing 22 newly un-encountered cases. A prediction precision of 81.8% was attained.

Fourth, an automated machine learning precedent case extraction model from large corpi was developed. An expanded corpus of 450 cases from the Federal Court of New York related to DSC disputes in the construction industry was utilized

for the development of the LSA feature space. Nine reduced feature spaces were developed with: 5, 10, 15, 20, 100, 200, 300, 400, and 500 features, respectively. From the analysis of this model, it could be concluded that: (1) low dimensioned reduced feature spaces are more representative to domain problems analysis closely related document collection; (2) higher dimensioned reduced feature spaces are more representative to domain problems analysis in dispersed and unrelated document collections; and (3) LSA reduced feature space of 10 features is the best to be adopted automating the extraction of similar DSC cases from large corpi.

The aforementioned research products contribute to the advancement of current practices of legal decision support and Knowledge Management in the construction legal domain. These advancements hold promise to: (1) decrease the costs associated with the utilization of legal experts in the construction industry for document analysis and initial advice on legal situation of a disputing party; (2) decrease the time related to litigation processes by allowing parties to investigate disputes and select alternate dispute resolution methods; (3) facilitate access to legal knowledge needed by practitioners in the construction industry; (4) provide a better understanding of the legal consequences of decision making in the construction industry; and (5) provide solid support documents and probabilistic indicators about the strength of a legal situation of a disputing party for better decision making about resolution mechanisms.

8.2 Research Contributions

The main contributions of the current research can be summarized as follows:

1. Development of a coherent and fully integrated methodology for legal decision support in the construction industry based on legal factors governing litigation outcomes. This contribution is considered to be the first of its nature in the construction legal domain. As illustrated in chapter 2 of this dissertation, all previous researches target generic factors for their analysis and did not incorporate legal factors.

2. Identification of a set of significant legal factors that governs verdicts of DSC cases in the construction industry. These factors provide very useful insight on this important type of construction disputes. As illustrated earlier, in case of a DSC dispute, an owner and/or a contractor can assess the strength of their situation based on the identified factors if resolving through litigation is decided. This assessment would allow disputing parties to take a more assured decision about other resolution mechanism like amicable settlement, mitigation, and/or arbitration. Furthermore, some of the identified factors are related to the wording of contracts and technical specifications in the construction industry. Therefore, the current research provides knowledge to contractors about factors to which emphasis should be given while bidding for new projects and upon which control should be maintained while performing a project.

3. Development of two automated models through machine learning. The first automates the prediction of outcomes of DSC litigation, and the second automates the extraction of significant legal factors governing this

prediction. Both models (1) provide a better understanding to decision makers about the legal consequences of their decisions; (2) save time and cost incurred due to the need of specialized legal expertise (3) help to relieve the negative consequences associated with lengthy claims and disputes resolution in the construction industry. In addition, the second model is considered to be a major contribution to the construction industry since it is the first of its nature.

4. Development of an automated model through machine learning for the extraction of supporting documents in the form of precedent DSC cases based on their similarity to newly introduced ones. The contribution of this model is not only anticipated to help practitioners in the construction industry better understand the consequences of their legal decision making, but is also expected to be beneficial to legal experts by saving time and money associated with these labor intensive tasks.

8.3 Future Research

Although the current research was able to fully accomplish its research objectives, a number of additional research directions have been identified including: (1) extending the research methodology of the current research to cover other types of major disputes in the construction industry like Damages for Breach of Contracts, Schedule Delays, Payment Delays, and Change Orders; (2) extending the research methodology of the current research to cover other jurisdictions; (3) extending the research methodology of the current research to cover financial claims and provide

automated models to monetary values related to different disputing parties; and (4) investigating other ML and NLP algorithms for the enhancement of the aforementioned methodology.

REFERENCES

All County Paving Corp. v Suffolk County Water Authority, 20 A.D.3d 438; 798 N.Y.S.2d 523; (2005).

Aleksander, I., and Morton, H. (1995). "An introduction to neural computing, 2nd Ed." *International Thomson Computer Press, London*.

Allen, J. (1995). "Natural language understanding." 2nd edition, *The Benjamin/Cummings Publishing Company, Inc., Redwood, California, USA.*

Aiolli, F., and Sperduti, A. (2005). "Multiclass classification with multi-prototype support vector machines." *Journal of Machine Learning Research*, (6); 817–850.

Aamodt, A. & Plaza, E. (1994). "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches." *AI Communications*, 7(i): 39-59.

Arditi, D., and Pulket, T. (2005). "Predicting the Outcome of construction litigation using boosted decision trees." *J. Comput. Civ. Eng.*, 19(4), 387-393.

Arditi, D., Oksay, F., and Tokdemir, O. (1998). "Predicting the outcomes of construction litigation using neural networks." *Comput Aid Civ Infrastruct Eng*, 13, 75-81.

Arditi, D., and Tokdemir, O. (1999). "Using case-based reasoning to predict the outcome of construction litigation." *Comput Aid Civ Infrastruct Eng*, 14(6), 385-393.

Ashley, K. D., and Rissland, E. L. (1988a). "Case-based approach to modeling legal expertise." *IEEE Expert*, 3(3), 70-77.

Ashley, K. D., and Rissland, E. L. (1988b). "Dynamic assessment of relevancy in a case-based reasoner." *IEEE*, 3, 208-214.

Bachant, J., & McDermott, J. (1984). "R1 Revisited: Four years in the Trenches." *The AI Magazine*, 5(iii).

Berman, D. H., and Hafner, C. D. (1989). "Potential of artificial intelligence to help solve the crisis in our legal system." *Commun ACM*, 32(8), 928-938.

Berman, D. H., and Hafner, C. D. (1991). "Incorporating procedural context into a model of case-based legal reasoning." *Third International Conference on Artificial Intelligence and Law*, ACM, New York, NY, 12-20.

Berman, D. H., and Hafner, C. D. (1995). "Understanding precedents in a temporal context of evolving legal doctrine." *Proc., International Conference on Artificial Intelligence and Law,* ACM, New York, NY, 42-51.

Bramer, M. (2007). "Principles of data mining." *Springer Verlag London Limited*.

Branting, K. (1993). "A Reduction-graph model of ratio decidendi." *In Proceedings of Fourth International Conference on Artificial Intelligence and Law*, 40- 49. Amsterdam: Association of Computing Machinery.

Branting, K.; Hastings, J.; and Lockwood, J. (2001). "CARMA: A case-based range management advisor." *In Proc. IAAI-2001*, 3–10.

Britannica. <http://www.britannica.com> (Accessed 2007).

Brüninghaus, S. and Ashley, K. D. (2001). "The role of information extraction for textual CBR." *Proc. 4th International Conference on Case-Based Reasoning*, Springer-Verlag, Berlin, Germany, 74-89.

Brüninghaus, S. and Ashley, K. D. (2003). "Combining Case-Based and Model-Based Reasoning for Predicting the Outcome of Legal Cases." *In Proc. 5th International Conference on Case-Based Reasoning.*

Brüninghaus, S. and Ashley, K. (2005). "Reasoning with textual cases." *Case-Based Reasoning Research and Development*, Springer-Verlag, Berlin, Germany, 137-15.

Bubbers, G., and Christian, J. (1992). "Hypertext and claim analysis." *J. Constr. Engrg. and Mgmt.*, 118(4), 716-730.

Caldas, C. H., Soibelman, L., and Han, J. (2002). "Automated classification of construction project documents." *J. Comput. Civ. Eng.*, 16(4), 234-243.

Caldas, C. H., and Soibelman, L. (2003). "Automating hierarchical document classification for construction management information systems." *Autom. Constr.*, 12(4), 395-406.

Caldas, C. H., Soibelman, L., and Gasser, L. (2005). "Methodology for the integration of project documents in model-based information systems." *J. Comput. Civ. Eng.*, 19(1), 25-33.

Callahan, M. T., Bramble B. B., and Lurie, P. M. (1990). "Arbitration of construction disputes." *Wiley, New York*.

Cannon, E. O., Amini, A., Bender, A., Sternberg, M. J. E., Muggleton, S. H., Glen, R. C., and Mitchel, J. B. O. (2007). "Support vector inductive logic programming outperforms the Naïve Bayes classifier and inductive logic programming for the classification of bioactive chemical compounds." *J. of Comput Aided Mol.,* 21, 269-280.

Charles Sundstrom et al., v. The State of New York, 106 N.E. 924; (1914).

Cheeks, R. J. (2003). "Multistep dispute resolution in design and construction industry." *J. of Prof. Issues in Eng. Education and Practices.*, 129, 84-90.

Christie v. United States, 35 S. Ct. 565; 59 L. Ed. 933; (1915).

Chua, D. K. H., Li, D. Z., and Chan, W. T. (2001). "Case-based reasoning approach in bid decision making." *J. Constr. Eng. Manage.*, 127(1), 35–45.

Chua, D. K. H., and Loh, P. K., (2006). "CB-Contract: case-based reasoning approach to construction contract strategy formulation." *J. Comp. in Civ. Engrg*., 20(5), 339-350.

Chau, K.W. (2005). "Predicting construction litigation outcome using particle swarm optimization." *Lecture Notes in Artificial Intelligence*, 3533, 571-578

Chau, K.W. (2006). "Application of a PSO-based neural network in analysis of outcomes of construction claims." *Automation in Construction,* 16, 642–646.

Chau, K.W. (2006). "Prediction of construction litigation outcome – a case-based reasoning approach." M. Ali and R. Dapoigny (Eds.): *IEA/AIE 2006, LNAI 4031, © Springer-Verlag Berlin Heidelberg 2006*, 548 – 553.

Cheeks, R. J. (2003). "Multistep dispute resolution in design and construction industry" *Journal of Professional issues in Engineering Education and Practice*, 129, 84-90.

Chen, J. and Hsu, S.C. (2007). "Hybrid ANN-CBR model for disputed change orders in construction projects." *Automation in Construction*, 17, 56–64.

Choi, F. Y. Y., Wiemer-Hastings, P., and Moore, J. (2001). "Latent semantic analysis for text segmentation." *In Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing*, Seattle, WA, 109–117.

Church, K. W., and Rau, L. F. (1995). “Commercial applications of natural language processing.” *Commun ACM*, 38(11), 71-79.

Clancey, W.J., (1985). “Heurestic Classification.” *Artificial Intelligence*, 27: pp289-350.

Coenen, F. & Bench-Capon, T.J.M. (1992). “Maintenance and maintainability in regulation based systems.” *ICL Technical Journal*, May 1992, pp.76-84.

Cobb, J. E., and Diekmann, J. E. (1986). “Claims analysis expert system.” *Proj Manage J*, 17(2), 39-48.

Costantino, M., Morgan, R. G., Collingham, R. J., and Garigliano, R. (1997). “Natural language processing and information extraction: Qualitative analysis of financial news articles.” *Proc., IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*, IEEE, Piscataway, NJ, 116-122.

Crammer, K., and Singer, Y. (2003). “Ultraconservative online algorithms for multiclass problems.” *Journal of Machine Learning Research,* 3, 951-991.

Daniel, G., Dienstuhl, J., Engell, S., Felske, S., Goser, K., Klinkenberg, R., Morik, K., Ritthoff, O., and Schmidt-Traub, H. (2002). “Advances in computational intelligence – theory and practice, chapter novel learning tasks, optimization, and their application,” Springer, 245-318.

Daniels, J., and Rissland, E. (1997). “Finding legally relevant passages in case opinions.” *In Proceedings of the Sixth International Conference on AI and Law. (ICAIL-97)*, SIGART, IAAIL, UMIACS, University of Melbourne, 39 – 46.

Demian, P., and Fruchter, R. (2005). “Measuring relevance in support of design reuse from archives of building product models.” *J. Comput. Civ. Eng.*, 19(2), 119-136.

Deerwester, S., dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). “Indexing by latent semantic analysis.” Journal of the American Society for Information Sciences, 41, 391-407.

Diekmann, J. E., and Kruppenbacher, T. A. (1984). “Claims analysis and computer reasoning.” *J. Constr. Eng. Manage.*, 110(4), 391-408.

Doğan, S. Z., Arditi, D., and Günaydın, H. M., (2006). “Determining attribute weights in a cbr model for early cost prediction of structural systems.” *J. Constr. Engrg. and Mgmt.*, 132(10), 1092-1098.

Doğan, S. Z., Arditi, D., and Gunaydin, H. M. (2008). "Using decision trees for determining attribute weights in case-based models of early cost prediction." *Journal of Construction Engineering and Management*, 134(2); 146-152.

Drake & Piper v. New York State Thruway Authority, 22 A.D.2d 321; 255 N.Y.S.2d 368; 1965 N.Y. App. Div (1965).

DTI (1992). "Knowledge-based systems survey of uk applications." *Department of Trade & Industry*, UK.

DTREG (2008). < http://www.dtreg.com/svm.htm> (Accessed 2008).

Dumais, S. (1990). "Improving the information retrieval from external sources." *Behavior Research Methods and Computers*, 23, 229-236.

Dumais, S. (1991). "Improving the retrieval of information from external sources." *Behavior Research Methods, Instruments, and Computers*, 23(2), 229-236.

Egri, P.A. and Underwood, P.F. (1995). "HILDA: knowledge extraction from neural networks in legal rule based and case based reasoning**."** *Neural Networks, 1995. Proceedings., IEEE International Conference*, 4, 1800-1805.

Elhadi, M. T. (2001). "Using statutes-based IR to derive legal CBR." *Applied Artificial Intelligence*, 15, 587- 600.

El-adaway, I. H. (2008). "Construction dispute mitigation through multi-agent based simulation and risk management modeling." *Ph. D. thesis*, Department of Civil, Construction, and Environmental Engineering, *Iowa State Univ., Ames, IA.*

El-Saadi, M. M. H. (1998). "Administrative Procedures for the Management of Construction Claims." *Master Thesis*, *Department of Engineering and Architecture, American University of Beirut*, *Beirut, Lebanon*.

Elvevag, B., Foltz, P. W., Weinberger, D. R., and Goldberg, T. E. (2007). "Quantifying incoherence in speech: An automated methodology and novel application to schizophrenia." *Schizophrenia Research*, 93(1), 304-316.

Erickson-Shaver Contracting Corp. v. United States, 9 Cl. Ct. 302, 304 (1985).

Faber v. The City of New York, 222 N.Y. 255; (1918).

Fisk, E. R. (2000). "Construction project administration." 6th Ed., *Prentice Hall*, New Jersey.

Foltz, P. W., Kintsch, W., and Landauer, T. K. (1998). “Analysis of text coherence using latent semantic analysis.” *Discourse Processes*, 25, 285-307.

Flemming, U. and Woodbury, R., (1995). “Software environment to support phases in building (SEED): overview,” *Journal of Architectural Engineering, ASCE*, 1(4) (1995), 147-52.

Foundation Company, v. The State of New York, 184 N.Y.S. 720; (1920).

French, S. (1982). “Sequencing and scheduling: an introduction to the mathematics of the job-shop”. *New York, NY.: Ellis Horwood.*

Fruin-colnon Corporation, traylor bros., inc. and Onyx Construction & Equipment, Inc., A Joint Venture, v. Niagara frontier transportation authority, 585 N.Y.S.2d 248; (1992).

Golding, A., and Rosenbloom, P. (1996). “Improving accuracy by combining rule-based and case-based reasoning.” *Artificial Intelligence*, 87(1-2):215–254.

Graham, D., and Smith, S. D., (2004). “Estimating the productivity of cyclic construction operations using case-based reasoning.” *Advanced Engineering Informatics, 18* (1), 17-28.

Greene, H. W. (1998). “LIMDEP-User’s Manual, Version 7.0.” *Econometric Software Inc., New York, NY.*

GTP. < http://www.cs.utk.edu/~lsi/soft.html> (Accessed 2008).

Hajjar, D., and AbouRizk, S. M. (2000). “Integrating document management with project and company data.” *J. Comput. Civ. Eng.*, 14(1), 70–77.

Harmon, K. J. (2003). “Dispute review board and construction conflicts: attitude and opinion of construction industry members.” *J. of Manag. in Eng.*, 19, 121-125.

Hegab, M., Nassar, K. (2005). “Decision support system for commencement delay claims.” *Practice Periodical On Structural Design And Construction © Asce*, 10(3), 1084-0680.

Hinze, J. (1998). “Construction planning and scheduling” 2nd Ed., *Upper Saddle River, N.J.: Prentice Hall.*

Hofmann, T. (1999). “Probabilistic latent semantic indexing.” *Proceedings of the National Academy of Science*, 101, 5228-5235.

Horgan, v. The City of New York, 160 N.Y. 516; (1899).

Howard, K. A., Jing, B., and Kahana, M.K. (2007). “Handbook of latent semantic analysis.” *Lawrence Erlbaum Associates, Mahawah, New Jersey*, Chapter 7: Semantic Structure and Episodic Memory, 121-142.

Hutchins, J., (1997), “Milestones in machine translation: episodes from the history of computers and translation.” *Language Today*, 3, 22-23.

Iacobelli Construction, Inc. v. County of Monroe, 32 F.3d 19; 1994 U.S. App.; (1994).

Ioannou, P. G., and Liu, L. Y. (1993). “Advanced construction technology system—ACTS.” *J. Constr. Eng. Manage.*, 119(2), 288-306.

James F. Leary and Thomas J. Morrison, v. The State of New York, City of Watervliet, 160 N.Y.S. 1042; (1916).

Jervis, B. M., & Levin, P. (1988). “Construction law principles and practice.” *New York: McGraw-Hill*, 9-36.

Johns, C. H. W. “Babylonian law - the code of Hammurabi.” <http://www.fordham.edu/halsall/ancient/hamcode.html> (Accessed 2007).

Jurafsky, D., and Martin, J. H. (2000). “Speech and Language Processing.” *Prentice Hall, New Jersey*.

Kaneta, T., Furusaka, S., Nagaoka, H., Kimoto, K., Okamoto, H., (1999). “Process model of design and construction activities of a building.” *Computer-Aided Civil and Infrastructure Engineering*, 14 (1), 45–54.

Kim, M. P. (1989). “U.S. Army Corps Engineers construction contract claims guidance system.” *Proc., Constr Congr I Excellence Constr Proj*, 203-209.

Kinser Construction Company v. The State of New York, 204 N.Y. 381; 97 N.E. 871; (1912).

Klinkenberg, R. (2001). “Using labeled and unlabeled data to learn drifting concepts.” *In Workshop notes of the IJCAI-01 Workshop on Learning from Temporal and Spatial Data*, 16–24.

Klinkenberg, R. (2003). “Predicting phases in business cycles under concept drift.” *Proc. of LLWA-03*, 3–10.

Klinkenberg, R. (2004). “Learning drifting concepts: Example selection vs. Example weighting.” *Intelligent Data Analysis (IDA), Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift*, 8(3).

Klinkenberg, R. and Joachims, T. (2000). “Detecting concept drift with support vector machines.” In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, 487–494, San Francisco, CA, USA, 2000. Morgan Kaufmann.

Klinkenberg, R. and R¨uping, S. (2003). “Concept drift and the importance of examples.” In J¨urgen Franke, Gholamreza Nakhaeizadeh, and Ingrid Renz, editors, *Text Mining – Theoretical Aspects and Applications*, 55–77. Physica-Verlag, Heidelberg, Germany, 2003.

Klinkenberg, R., Ritthoff, O., and Morik, K. (2002). “Novel learning tasks from practical applications.” *In Proceedings of the workshop of the special interest groups Machine Learning (FGML)*, 46–59.

Koll, M. (1979). “An approach to concept-based information retrieval.” *ACM SIGIR Forum, XIII,* 32-50.

Kolodner, J. L. (1993). “Case-Based Reasoning.” *MorganKaufmann*.

Kosovac, B., Froese, T., and Vanier, D. (2000). “Integrating heterogeneous data representations in model-based AEC/FM systems.” *Proc., CIT 2000*, Reykjavik, Iceland, 1, 556-566.

Kowalski, G. J., and Maybury, M. T. (2000). “Information Storage and Retrieval Systems: Theory and Implementation.” 2nd edition, in the Information Retrieval Series, Vol 8, *Springer*, Croft, W. B., editor.

Krol, J., (1993). “Construction Contract Law.” *John Wiley and Sons.*

Kumaraswamy, M.M., Ugwu, O.O., Palaneeswaran, E., and Rahman, M.M. (2004). “Empowering collaborative decisions in complex construction project scenarios.” *Eng. Constr. Archit. Manage.*, 11(2), 133-142.

Labidi, S. (1997). “Managing multi-expertise design of effective cooperative knowledge-based system.” *Proc., 1997 IEEE Knowledge & Data Engineering Exchange Workshop*, IEEE, Piscataway, NJ, 10-18.

Landauer, T. K. (2002). “On the computational basis of cognitive: Arguments from LSA.” *The psychology of learning and motivation, New York: Academic Press*, Ross, B. H., editor, 43-84.

Landauer, T. K. & Dumais, S. T. (1997). “A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge.” *Psychological Review*, 104, 211-140.

Landauer, T. K., Laham, d., and Foltz, P. W. (2003a). "Automated essay assessment." *Assessment in Education: Principles, Policy and Practice*, 10(30, 295-308.

Landauer T. K., Laham, d., and Foltz, P. W. (2003b). "Automated scoring and annotation of essays with the intelligent essay assessor." *Automated Essay Scoring: A Cross-disciplinary Prospective*, Shermis, M. D., and Burstein, J., editors, Mahwah, NJ: Lawrence Erlbaum Associates.

Landauer, T. K., McNamara, D. S., Dennis, S., and Kintsch, W. (2007). "Handbook of latent semantic analysis." *Lawrence Erlbaum Associates, London*.

Lee, M.J., Hanna, A. S., and Loh, W. Y. (2004). "Decision tree approach to classify and quantify cumulative impact of change orders on productivity." *Journal of Computing in Civil Engineering*, 18(2); 132-144.

Legal Dictionary. <http://legal-dictionary.com> (Accessed 2008).

Levin, P. (1998). "Construction contracts, claims, and disputes." *American Society of Civil Engineer (ASCE) Press, Reston, VA*.

Letsche, T., and Berry, M. W. (1997). "Large-scale information retrieval with latent semantic indexing." *Information sciences*, 100, 105-137.

LexisNexis. (2008). <http://www.lexisnexis.com/us/lnacademic/search/casessubmitForm.do> (Accessed 2008).

Li, H., (1996). "Case-based reasoning for intelligent support of construction negotiation," *Information and Management*, 5(30), 231 – 238.

Liddy, E. D., (2003). "Natural Language Processing, 2nd Ed." *Encyclopedia of Library and Information Science,* Bates, J., Maack, M. N., and Drake, M., editors, Marcel Decker, Inc.

Lin, K. Y. and Soibelman, L. (2005). "Knowledge assisted retrieval of online product information in A/E/C (Architecture/Engineering/Construction)." *J. Comput. Civ. Eng.*, 179, 48-59.

Lin, K. Y. and Soibelman, L. (2007). "Knowledge assisted retrieval of online product information in A/E/C (Architecture/Engineering/Construction)." *J. Constr. Eng. & Manage.*, 133, 871-879.

Love, P. E. D., Skitmore, R. M., and Earl, G. (1998). "Selecting a suitable procurement system for a building project." *Constr. Manage. Econom.*, 16, 221–233.

Lund, K., Burgess, C., and Atchley. R. A. (1995). "Semantic and associative priming in high-dimensional semantic space." *In Proceedings of the 17th Annual Conference of the Cognitive Science Society, CogSci'95*, Erlbaum, 660–665.

Luu DT, Ng ST, Chen, SE., (2005). "Formulating procurement selection criteria through case-based reasoning approach." *Jour. Comput. Civil Eng*. 19 (3), 269-76.

Luu, D. T., Ng, S. T., Chen, S. E., and Jefferies, M., (2006). "A strategy for evaluating a fuzzy case-based construction procurement selection system." *Advances in Engineering Software*, 37 (3), 159 – 171.

Mangasarian, O. L., and Musicant, D. R., (1999). "Massive Support Vector Regression." *Technical Report Data Mining Institute TR-99-02, University of Wisconsin.*

Maher, M. L. and Zhang, D. M., (1991). "CADSYN: Using case and decomposition knowledge for design synthesis." *In Proceeding of the First International Conference on Artificial Intelligence in Design,* Edinburgh, UK, Butterworth-Heineman, Oxford, 1991, pp. 137-50.

Manning, C. and Scheutze, H. (1999). "Foundations of statistical natural language processing." Cambridge: MIT Press.

Marcotte, P. (1990). "Hastening justice—Biden committee studies task force plan to cut trial delay." *Am. Bar Assoc. J.*, 76(1), 40.

Merrill, P. G. (2006). "Construction dispute review board + settlement panels: Save time, money, + headaches." *Contract Management Magazine*, 38-43.

Mierswa, I., and Morik, K. (2005a). "Automatic feature extraction for classifying audio data." *Machine Learning Journal*, 58:127–149.

Mierswa, I., and Morik, K. (2005b). "Method trees: building blocks for self-organizable representations of value series." *In Proc. of the Genetic and Evolutionary Computation Conference GECCO 2005, Workshop on Self-Organization In Representations For Evolutionary Algorithms: Building complexity from simplicity, 2005*. Washington,DC,USA, 1989-1992.

Mierswa, I., and Wurst, M. (2005a) "Efficient case based feature construction for heterogeneous learning tasks." *Technical Report CI-194/05, Collaborative Research Center 531, University of Dortmund, 2005*.

Mierswa, I., and Wurst, M. (2005b). "Efficient feature construction by meta learning – guiding the search in meta hypothesis space." *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining,* ACM, SIGKDD, and SIGMOD, Philadelphia, PA, USA, 935 – 940.

Mierswa. I. (2004). "Automatic feature extraction from large time series." *In Proc. of LWA 2004* - Lernen - Wissensentdeckung – Adaptivitat, Yale.

Misky, M. L., and Papert, S. A. (1990). "Perceptron." *MIT Press*.

Miyashita, K. and Sycara, K. (1992). "Cabins: Casebased interactive scheduler." In *Working Notes of AAAI Spring Symposium on Practical Approaches to Scheduling and Planning*, (Stanford, CA), AAAI, 1992.

Naoum, S. G. (1994). "Critical analysis of time and cost of management and traditional contracts." *J. Constr. Eng. Manage.*, 120(4), 687–705.

Ng, H. S., Toukourou, A., and Soibelman, L. (2006). "Knowledge discovery in a facility condition assessment database using text clustering." *J. Comput. Civ. Eng.*, 12(1), 50-59.

Nguyen, H., Ohn, S., and Chae, S. (2006). "Optimizing weighted kernel function for support vector macinesby genetic algorithm." *Mexican International Conference on Artificial Intelligence (MICAI), Apizaco, Mexico*, 583-592.

Nilsson, N. J. (2008). "Introduction to Machine Learning" <http://robotics.stanford.edu/~nilsson/mlbook.html> (Accessed 2008).

North Slope Technical Ltd. v. United States, 14 Cl. Ct. 242, 257 (1988).

Ohtake, Y., Nitta, K., Maeda, S., Ono, M., Ohsaki, H., and Yoneda, L. (1993). "HELIC-II: As a legal argumentation support system." *Proc., Conference on Artificial Intelligence Applications*, IEEE, Piscataway, NJ, 464.

Oracle. < http://www.oracle.com/index.html> (Accessed 2009).

Pearce, M., Goel, A., Kolodner, j. L., Zimring, C., Sentosa, L., and Billington, R., (1992). "Case-based design support: A case study in architectural design," *IEEE Expert*, 7(5) (1992), 14-20.

Peña-Mora, Feniosky, Sosa, Carlos E., and D. McCone, Sean. (2003). “Introduction to construction dispute resolution,” 1st Ed., *Prentice Hall, New Jersey.*

Piper, Inc., v. New York State Thruway Authority, ; 221 N.Y.S.2d 648, (1961).

Platt, J. (1999). “Fast training of support vector machines using sequential minimal optimization.” *MIT Press*.

Praehofer, H., and Kerschbaummayr, J., (1999). “Case-based reasoning techniques to support reusability in a requirement engineering and system design tool.” *Engineering Applications of Artificial Intelligence*, 12(6), 717-731.

Public Constructors, Inc. v. State of New York, 55 A.D.2d 368; 390 N.Y.S.2d 481; 1977 N.Y. App. Div.

Quinlan, J. R. (1993). “C4.5: Programs for machine learning.” *Morgan Kaufmann, Los Altos*.

Ralph S. Keep and Others v. The City of New York, 138 Misc. 194; 245 N.Y.S. 321; (1930).

Rapit-I (2008). <http://rapid-i.com/> (Accessed 2008).

Rehder, B., Schreiner, M. E., Wolfe, M. B., Laham, D., Landauer, T. K., and Kintsch, W. (1998). “Using latent semantic analysis to assess knowledge: Some teaching considerations.” *Discourse Processes*, 25, 337-354.

Ren, Z., Anumba, C. J., and Ugwa, O. O. (2001). “Construction claim management: Towards an agent based approach.” *J. Eng. Constr. Archit. Manage.*, 8, 185-197.

Reuber, R. (1997). “Management experience and management expertise.” *Decision Support Sys.*, 21(4), 51–60.

Riloff, E. (1996). “Automatically generating extraction patterns from untagged text.” *In Proceedingsof the Thirteenth National Conference on Artificial Intelligence. AAAI Press*. Menlo Park, CA, 1044-1049.

Rissland, E., Skalak, D. & Friedman, M.T. 1993. “Bank XX: A program to generate argument through case-base search.” *In Proceedings of the Fourth International Conference on Ardjicial Intelligence and Luw*, New York: Association for Computing Machinery, 117-124.

Ritthoff, O. and Klinkenberg, R. (2003). “Evolutionary feature space transformation using type-restricted generators.” *In Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, Chicago, IL, USA, 1606–1607.

Ritthoff, O., Klinkenberg, R., Fischer, S., and Mierswa, I. (2002). “A hybrid approach to feature selection and generation using an evolutionary algorithm.” *In Proc. of the 2002 U.K. Workshop on Computational Intelligence (UKCI-02)*, Univ. of Birmingham, UK, 147–154.

Robinson, S., Martinovski, B., Garg, S., Stephan, J., and Traum, D. (2004). “Issues in corpus development for multi-party multi-modal task-oriented dialogue.” *Proceedings of Fourth International Conference on Language Resources and Evaluation (LREC 2004*), Centro Cultural de Belem, Lisbon, Portugal, 1707-1710.

Roddis, W. M. K., and Bocox, J. (1997). “Case-based approach for steel bridge fabrication errors.” *J. Comput. Civ. Eng.*, 11(2), 84–91.

Rwelamila, P. D., and Meyer, C. (1999). “Appropriate or default project procurement systems.” *Cost Eng.*, 41(9), 40–44.

S. Pearson & Son, Inc., v. The State of New York, 182 N.Y.S. 481; (1920).

Salton, G. (1989). “Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.” *Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA*.

Salton, G., and Buckley, C. (1991). “Automatic text structuring and retrieval – experiment in automatic encyclopedia searching.” *Proceeding of the 14th Annual International ACM SIGIR Conference on Research and Development in information Retrieval*, 21-30.

Salton, G., and Lesk, M. E. (1968). “The SMART automatic document retrieval systems – an illustration.” *Journal of the ACM*, 15(1), 8-36.

Schank, R. (1982). “Dynamic memory: a theory of reminding and learning in computers and people.” Cambridge University Press, Cambridge, UK.

Schmitt, G. (1993). ‘‘Case-based design and creativity.’’ *Automation in Constr.*, 2, 11–19.

Schmitt, G. (1988). “ARCHPLAN- an Architectural front end to engineering expert systems.” *Expert Systems for Engineering Design*, Academic Press, New York, Rychener, M., editor.

Scherer, R. J., and Reul, S. (2000). "Retrieval of project knowledge from heterogeneous AEC documents." *Proc., Eight International Conference on Computer in Civil and Building Engineering*, Palo Alto, Calif., 812-819.

Schumacher, L. (1997). ''Defusing delay claims.'' *Civ. Eng*., 67(3), 60-62.

Servidone Construction Corporation v. The United States, 931F.3d 860; 1991 U.S. app.

Shawe-Taylor, J. and Cristianini, N. (1999). "Margin distribution bounds on generalization". *In Proceedings of the European Conference on Computational Learning Theory, EuroCOLT99*; 263-273.

Shawe-Taylor, J. and Cristianini, N. (2000). "Support vector machines and other kernel-based learning methods." *Cambridge University Press*.

Sirca, G., and Adeli, H. (2004). "Counterpropagation neural network model for steel girder bridge structures." *Journal of Bridge Engineering*, 9(1), 55–65.

Sirca, G. F., and Adili, H. (2005). "Case-based reasoning for converting working stress design-based bridge ratings to loaf factor design-based ratings." *Journal of Bridge Engineering*, 10(4), 450-459.

Soibelman, L., and Kim, H. (2002). "Data preparation process for construction knowledge generation through knowledge discovery in databases" *J. Comput. Civ. Eng.*, 16(1), 39-48.

Sycara, K., and Miyashita, K. (1994). "Learning from failure in casebased schedule repair." *Proc., 27th Hawaii Int. Conf. on System Sciences*, Institute of Electrical and Electronics Engineers, New York, 3, 122–131.

Tabachnick, B. G. and Fidell, L. S. (1996). "Using Multivariate Statistics." 3rd Edition, *Harper Collins, New York, NY, USA*.

Tony Carfagno and Others v. The City of New York, 187 A.D. 489; 175 N.Y.S. 682; 1919 N.Y. App. Div.

Travelers Casuality and Surety Company of America V. The United States of America, 75 Fed. Cl. 696; 2007 U.S. Claims.

Treacy, T. B. (1995). "Use of alternative dispute resolution in the construction industry." *Journal of Management in Engineering,* 11, 58-63.

Vapnik, V. (1998). "Statistical learning theory." Wiley.

Waheed, A., and Adeli, H. (2005). "Case-based reasoning in steel bridge engineering." *Journal of Knowledge based Systems*, 18, 37-46.

Wang, J., (1991). "Integrated Case –based Reasoning for Structural Design," *Ph.D. thesis, Department of Civil Engineering, Stanford University, CA,*.

Watson, I. and Abdullah, S., (1991). "Developing case-based reasoning system: A case study in diagnosing defects," *IEEE Digest No: 1994/057, Case Based Reasoning: Prospects for Applications (199)*, 1/1-1/3

Watson, I.D., Basden, A., and Brandon, P.S. (1992). "The client centred approach: expert system maintenance." *Expert Systems*, 9(iv): pp189-96.

Watson, I., and Marir, F., (2000). "Case-Based Reasoning: A Review." <http://www.ai-cbr.org/classroom/cbr-review.html>, (Accessed 2007).

Watson, I. (1997). "Applying case-based reasoning." *Morgan Kaufmann, San Mateo, California.*

Washington, S. P., Karlaftis, M. G., and Mannering, F.L. (2003). "Statistical and Econometric Methods for Transportation Data Analysis." *CHAPMAN & HALL/CRC, New York.*

Weber, R., Aha, D., Sandhu, N., and Munoz-Avila, H. (2001). "A textual case-based reasoning framework for knowledge management applications." *In Proceedings of the Ninth German Workshop on Case-Based Reasoning.* Shaker Verlag, 244-253.

Weber, R. (1998). "Intelligent Jurisprudence Research." *Ph.D. Dissertation, Federal University of Santa Catarina, Brazil.*

Weeks Dredging & Contracting, Inc. v. United States, 13 Cl. Ct. 193, 219 (1987).

Weston, J., and Watkins, C. (1999). "Support vector machines for multiclass pattern recognition." *In Proceedings of the 6th European Symposium on Artificial Neiral Networks (ESANN)*. Bruges, Belgium, 219-224.

William G. Horgan v. The City of New York; 160 N.Y. 516; 55 N.E. 204; 1899 N.Y. (1899).

Witten, I.H., and Frank, E. (2000). "Data mining: practical machine learning tools and techniques" *MorganKaufmann*.

Wolfe, M. B. W., Schreiner, M. E., Rehder, B., Laham, D., Foltz, P. W., and Kintsch, W. (1998). "Learning from text: Matching readers and text by latent semantic analysis." *Discourse Processes*, 25, 309-336

Wood, W. H. (2000). "The development of modes in textual design data." *Proc., Eight International Conference on Computer in Civil and Building Engineering*, Palo Alto, Calif., 882-889.

Xie, H., Isaa, R. A., and O'Brien W. (2003). "User model and configurable visitor for construction project information retrieval." *4th Joint International Symposium on Information Technology in Civil Engineering*, ASCE, Nashville, Tennessee, 47.

Yang, M. C., Wood, W. H., and Cutkosky, M. R. (1998). "Data mining for thesaurus generation in informal design information retrieval." *Proc., Int. Computing Congress*, ASCE, Reston, Va., 189-200.

Yang, M. C, and Lee, C.H. (2003). "A text mining approach on automatic generation of web directories and hierarchies." *Proc., 2003 IEEE/WIC International Conference on Web Intelligence*, IEEE, Washington, DC, 625.

Yau, N., and Yang, J., (1998). "Case-Based reasoning in construction management." *Computer-Aided Civil and Infrastructure Engineering,* 13 (2), 143–150.

Ye, S., and Liu, Y. (2008). "Study on development patterns of infrastructure projects." *Journal of Construction Engineering and Management*, 134(2); 94-102.

Zeleznikow, J. (2003). "An Australian perspective on research and development required for the construction of applied legal decision support systems." *Artificial Intelligence and Law,* 10: 237–260.

Zhu, Y., Mao, W., and Ahmad, I. (2007). "Capturing implicit structures in unstructured content of construction documents." *J. Comput. Civ. Eng.*, 21(3), 220-227.

APPENDIX A (LIST OF LEGAL FACTORS)

List of extracted factors:

Contract type

Mutual consent

Involved parties

Type of owner

Type of Contractor

Type of Project

Design responsibility (Contractor)

Full bidding documents (discrepancies reported)

Contractor deemed to have fully reviewed and familiarized himself with the site, conditions, and drawings

Is there an unforeseen physical condition clause?

(Type II differing site conditions) Are the conditions if any, unforeseen for an experienced Contractor?

(Type II differing site conditions) Did the Contractor know about the condition?

(Type II differing site conditions) Did the condition vary from the norm in similar construction operations?

(Type I differing site conditions) did the contract documents affirmatively indicate subsurface conditions?

(Type I differing site conditions) did the Contractor act as a reasonably prudent contractor in interpreting the contract documents?

(Type I differing site conditions) did the Contractor reasonably rely on the indications of subsurface conditions in the contract?

(Type I differing site conditions) did the subsurface conditions actually encountered differ materially from those indicated in the contract?

(Type I differing site conditions) were the actual subsurface conditions not reasonably foreseeable?

(Type I differing site conditions) was the Contractor's damage attributable to the materially different subsurface conditions?

Was the work stopped due to the encountered matter?

Did the Owner\ Owner Rep stop the works to perform adjustments due to factors related to the encountered matter?

Did the Contractor stop the works for any reason?

Did the Matter encountered require redesign?

Did the Matter encountered require changes in the nature and costs of the Contract?

Where the imposed changes made because it was cheaper\ better or because it was necessary?

Did the Matter encountered have safety related issues?

Did the Contractor raise the matter in the right procedural form stated by the Contract?

Was a decision taken with regard the settlement of the matter?

Did any of the parties raise his disagreement and stated his intentions for a claim under the contract?

Did the matter made completion of the project impossible?

Was there a clause giving the Owner the right to make changes to the project after final completion and acceptance without invalidating the contract

Was there a clause stopping the Contractor from claiming his lost profit against deducted\ changed\ modified works?

Did the parties make a mutual mistake as to the condition related to this matter?

Is there a sovereign immunity waiver clause?

Year (date) range on 5 years intervals

Type of judgment

Was there various changes made along the progress of the works?

Nature of damage

Does the contractor bare the risk for any unforeseen conditions?

Was the matter caused as a reason of the Owners own act, even if he did that unintentionally?

Did any of the concerned parties considered a breach of contract action?

Was experts' opinions provided by the Contractor's side?

Was experts' opinions provided by the Owner's side?

Did the specifications warn and illustrate the possibility of differing site conditions to those mentioned by the Contract Documents?

Was the Specifications governing the work if applicable "Performance specifications"?

Did the other party sough for a counter claim related to the same matter?

Did a triable issue of fact exist?

Did the specification have representation, even if found after that to be different from the actual conditions, of the matter in question?

Did the Contractor Under the terms of the contract agreed not to ask for or recover extra compensation beyond the contract price?

Was there a no allowance for extras clause?

Was the additional work approved by the Engineer\ State Engineer?

Did The Owner\ Owner Rep. falsely state that the matter encountered in hand, so far as known, was shown in the Contract documents?

Was the Contract Lumpsum or unit price or other?

Is the Contractor a foreign Company that does not has the right to sue in USA?

If the Matter was caused due to the fault of the Owner, did he adjust the mistake?

Was the extra work done for the benefit of the Owner or Contractor?

Are there evident facts showing that the Owner had bad intentions representing the matter in the Contract Documents?

Are there enough evidence to show that there were no time for the Contractor to perform his own investigations?

Was the extra work performed as temporary work to protect part of the permanent works required under the contract?

If this is an appeal, who was the winning party in the initial trial?

APPENDIX B (SVM MODEL OUTPUT)

SVM Modeling Output

Trial 1			
Model Properties			
C	1		
M (Fit Logistic Model to Output)	TRUE		
Polynomia Degree	**1**		
Model Output			
Accuracy	94.00%	±	9.17%
Precision	93.83%	±	9.60%
Recall	93.50%	±	13.43%
Area Under Curve (AUC)	95.40%	±	5.90%
Positive Class	CONTRACTOR		
	True OWNER	True CONTRACTOR	Class Precision
Prediction OWNER	53	3	94.64%
Prediction Contractor	1	43	97.73%
Class Recall	98.15%	93.48%	

W-SMO

SMO

Kernel used:

Linear Kernel: K(x,y) =

<x,y>

Classifier for classes: OWNER,

CONTRACTOR

BinarySMO

Machine linear: showing attribute weights, not support vectors.

0.3743 * (normalized)

Ptype

+ -0.0273 *

(normalized) DSCC

+ 2.2885 *

(normalized) DSC

+ 0.2049 * (normalized) N&C + 0.2618 * (normalized)

Conraise

+ 0.8931 * (normalized) ComImpossible
+ -0.0971 * (normalized) Ochange
+ 0.0345 * (normalized) Mmistake
+ -0.18 * (normalized)
Year
+ 0.8762 * (normalized) Ocause
+ -1.1228 * (normalized) SpecWarn
+ 0.0719 * (normalized) SpecRep
+ -1.183 * (normalized) CNoExtra
+ -0.6137 * (normalized) Ofalsely
+ 0.9626 * (normalized) OAdjust
-1.5445

Number of kernel evaluations: 2059 (86.149% cached)
Logistic Regression with ridge parameter of
1.0E-8
Coefficients...
Variable Coeff.
1 -3.2962
Intercept 1.3227
Odds Ratios...
Variable O.R.
1 0.037

Trial 2			
Model Properties			
C	1		
M (Fit Logistic Model to Output)	TRUE		
Polynomia Degree	**2**		
Model Output			
Accuracy	98.00%	±	6.00%
Precision	98.00%	±	6.00%
Recall	98.00%	±	6.00%
Area Under Curve (AUC)	99.60%	±	1.20%
Positive Class	CONTRACTOR		
	True OWNER	True CONTRACTOR	Class Precision
Prediction OWNER	53	1	98.15%
Prediction Contractor	1	45	97.83%

Class Recall	98.15%	97.83%

W-SMO

SMO

Kernel used:

Poly Kernel: K(x,y) =

<x,y>^2.0

Classifier for classes: OWNER,

CONTRACTOR

BinarySMO

- 0.1391 * <0.666667 0 1 0 0 0 0 1 1 0 0 1 1 0 0 > * X]
- 0.0593 * <0.333333 1 0 1 1 0 0 1 0.7 0 0 1 0 0 0 > * X]
+ 0.1121 * <0.333333 0 1 1 0 0 0 1 1 0 0 1 1 0 1 > * X]
- 0.0405 * <0.666667 0 0 1 1 0 0 0 1 0 1 1 0 0 1 > * X]
+ 0.0856 * <0.666667 1 1 0 0 0 0 1 0.4 0 0 1 0 0 0 > * X]
- 0.2478 * <1 1 1 0 1 1 0 1 1 1 0 1 1 0.5 0 > * X]
+ 0.0553 * <0.666667 1 1 0 0 0 0 1 0.4 0 0 1 0 0 0 > * X]
+ 0.0877 * <1 0 1 0 1 1 0 0 1 1 0 1 1 0.5 0 > * X]
+ 0.0033 * <0.666667 1 1 0 1 0 0 0 0.2 0 0 1 0 0 0 > * X]
+ 0.1551 * <0.666667 1 1 0 0 0 0 1 0.4 0 0 1 0 0 0 > * X]
+ 0.0794 * <0.666667 1 1 0 1 0 0 0 0.2 0 0 1 0 0 0 > * X]
- 0.0348 * <0.666667 1 0 0 1 0 0 0 0.3 0 0 1 0 0 0 > * X]
- 0.0215 * <0.666667 0 0 1 1 0 1 1 1 0 0 1 0 0 0 > * X]
- 0.0135 * <0.666667 1 0 1 1 0 0 0 0.8 0 0 0 0 0 0 > * X]
- 0.0332 * <1 0 0 0 1 0 0 1 1 0 0 0 0 0 0
> * X]
+ 0.1439 * <1 1 0 1 1 1 0 1 0.8 1 0 1 0 0 0 > * X]
- 0.06 * <0.333333 1 0 1 1 0 0 1 0.7 0 0 1 0 0 0 > * X]
+ 0.0068 * <0.333333 1 1 1 1 1 0 1 0.8 0 0 0 0 0.5 0 > * X]
+ 0.0201 * <1 0 1 0 1 1 0 0 1 1 0 1 1 0.5 0 > * X]
- 0.0738 * <0.666667 0 1 0 0 0 0 1 1 0 0 1 1 0 0 > * X]
+ 0.0478 * <0.666667 0 1 1 1 1 1 1 1 0 0 1 1 0 0 > * X]
- 0.0804 * <0.666667 1 0 0 0 0 0 0 0.5 0 0 1 0 0 0 > * X]
+ 0.0088 * <0.666667 0 1 1 1 1 1 1 1 0 0 1 1 0 0 > * X]
- 0.0312 * <1 0 0 0 0 0 0 1 0.8 0 0 1 0 0 0 > * X]
- 0.0293 * <1 1 1 0 1 1 0 1 1 1 0 1 1 0.5 0 > * X]
+ 0.0486 * <1 0 1 0 1 0 0 1 1 0 0 1 0 0.5 0 > * X]
- 0.0255 * <0.666667 1 0 0 0 0 0 0 0.5 0 0 1 0 0 0 > * X]
- 0.0208 * <0.666667 0 1 0 0 0 0 1 1 0 0 1 1 0 0 > * X]

- 0.0266 * <0.666667 0 0 1 1 0 1 1 1 0 0 1 0 0 0 > * X]
- 0.038 * <0.666667 0 0 1 1 0 1 1 1 0 0 1 0 0 0 > * X]
+ 0.0008 * <0.666667 1 1 0 1 0 0 0 0.2 0 0 1 0 0 0 > * X]
- 0.0017 * <0.666667 1 0 1 1 0 0 1 0.1 0 1 0 0 0 1 > * X]
- 0.0068 * <0.666667 1 0 0 1 0 0 0 0.3 0 0 1 0 0 0 > * X]
+ 0.0572 * <1 0 1 0 1 0 0 1 1 1 0 1 1 0 0 > * X]
+ 0.0373 * <0.333333 1 1 1 1 1 0 1 0.8 0 0 0 0 0.5 0 > * X]
- 0.0709 * <0.666667 0 1 1 0 0 0 1 0 0 1 1 1 0 0 > * X]
+ 0.1049 * <1 0 1 0 1 0 0 1 1 1 0 1 1 0 0 > * X]

-0.7655

Number of support vectors:
37

Number of kernel evaluations: 7637 (78.233% cached)
Logistic Regression with ridge parameter of
1.0E-8

Coefficients...

Variable	Coeff.
1	-28.0718
Intercept	-5.2416

Odds Ratios...

Variable	O.R.
1	0

Trial 3			
Model Properties			
C	1		
M (Fit Logistic Model to Output)	TRUE		
Polynomia Degree	**3**		
Model Output			
Accuracy	100.00%	±	0
Precision	100.00%	±	0
Recall	100.00%	±	0
Area Under Curve (AUC)	100.00%	±	0
Positive Class	CONTRACTOR		
	True OWNER	True CONTRACTOR	Class Precision
Prediction OWNER	53	1	98.15%
Prediction Contractor	1	45	97.83%

Class Recall	98.15%	97.83%

W-SMO

SMO

Kernel used:

Poly Kernel: K(x,y) =

<x,y>^3.0

Classifier for classes: OWNER,

CONTRACTOR

BinarySMO

```
 -       0.019  * <0.666667 0 1 0 0 0 0 1 1 0 0 1 1 0 0 > * X]
 +       0.0031 * <0.333333 0 1 1 0 0 0 1 1 0 0 1 0 0 0 > * X]
 -       0.006  * <0.333333 1 0 1 1 0 0 1 0.7 0 0 1 0 0 0 > * X]
 +       0.0079 * <0.333333 0 1 1 0 0 0 1 1 0 0 1 1 0 1 > * X]
 -       0.0016 * <0.666667 0 0 1 1 0 0 0 1 0 1 1 0 0 1 > * X]
 +       0.003  * <0.666667 0 1 1 1 0 0 1 1 0 0 0 0 0 0 > * X]
 -       0.0032 * <0.666667 1 0 1 1 0 0 0 0.8 0 0 0 0 0 0 > * X]
 +       0.0141 * <0.666667 1 1 0 0 0 0 1 0.4 0 0 1 0 0 0 > * X]
 -       0.0102 * <1 1 1 0 1 1 0 1 1 1 0 1 1 0.5 0 > * X]
 +       0.0044 * <0.666667 1 1 0 0 0 0 1 0.4 0 0 1 0 0 0 > * X]
 +       0.0075 * <1 0 1 0 1 1 0 0 1 1 0 1 1 0.5 0 > * X]
 +       0.0022 * <0.666667 1 1 0 1 0 0 0 0.2 0 0 1 0 0 0 > * X]
 +       0.0099 * <0.666667 1 1 0 0 0 0 1 0.4 0 0 1 0 0 0 > * X]
 +       0.0248 * <0.666667 1 1 0 1 0 0 0 0.2 0 0 1 0 0 0 > * X]
 +       0.0024 * <0.333333 1 1 1 0 1 0 1 0 1 0 0 0 0 0 > * X]
 -       0.0139 * <0.666667 1 0 0 1 0 0 0 0.3 0 0 1 0 0 0 > * X]
 -       0.0014 * <0.666667 0 0 1 1 0 1 1 1 0 0 1 0 0 0 > * X]
 -       0.0024 * <0.333333 1 0 1 1 0 0 1 0.7 0 0 1 0 0 0 > * X]
 -       0.005  * <1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 >
* X]
 +       0.0103 * <1 1 0 1 1 1 0 1 0.8 1 0 1 0 0 0 > * X]
 -       0.0051 * <0.333333 1 0 1 1 0 0 1 0.7 0 0 1 0 0 0 > * X]
 +       0.0008 * <0.333333 1 1 1 1 1 0 1 0.8 0 0 0 0 0.5 0 > * X]
 -       0.0004 * <1 0 1 0 1 0 0 1 1 0 1 1 1 0 0
> * X]
 +       0       * <0.666667 0 1 1 1 0 0 1 1 0 0 0 0 0 0 > * X]
 +       0.0019 * <0.333333 0 1 1 0 0 0 1 1 0 0 1 0 0 0 > * X]
 +       0.0013 * <1 0 1 0 1 1 0 0 1 1 0 1 1 0.5 0 > * X]
 -       0.0012 * <0.666667 0 1 0 0 0 0 1 1 0 0 1 1 0 0 > * X]
```

```
 +       0.0009 * <0.666667 0 1 1 1 1 1 1 1 0 0 1 1 0 0 > * X]
 -       0.0138 * <0.666667 1 0 0 0 0 0 0 0.5 0 0 1 0 0 0 > * X]
 +       0.0009 * <0.666667 0 1 1 1 1 1 1 1 0 0 1 1 0 0 > * X]
 -       0.006  * <1 0 0 0 0 0 0 1 0.8 0 0 1 0 0 0 > * X]
 -       0.0056 * <1 1 1 0 1 1 0 1 1 1 0 1 1 0.5 0 > * X]
 +       0.0033 * <1 0 1 0 1 0 0 1 1 0 0 1 0 0.5 0 > * X]
 -       0.0055 * <0.666667 1 0 0 0 0 0 0 0.5 0 0 1 0 0 0 > * X]
 -       0.0003 * <0.666667 0 1 0 0 0 0 1 1 0 0 1 1 0 0 > * X]
 -       0.0034 * <0.666667 0 0 1 1 0 1 1 1 0 0 1 0 0 0 > * X]
 -       0.0003 * <0.666667 0 0 1 1 0 1 1 1 0 0 1 0 0 0 > * X]
 -       0.0028 * <0.666667 1 1 0 1 0 0 0 0.3 0 1 1 1 0.5 0 > * X]
 +       0.0017 * <0.666667 1 1 0 1 0 0 0 0.2 0 0 1 0 0 0 > * X]
 +       0.0028 * <0.333333 0 1 1 0 0 0 1 1 0 0 1 0 0 0 > * X]
 -       0.0009 * <0.666667 1 0 1 1 0 0 1 0.1 0 1 0 0 0 1 > * X]
 -       0.0001 * <0.666667 0 0 1 1 0 0 0 1 0 1 1 0 0 1 > * X]
 -       0.005  * <0.666667 1 0 0 1 0 0 0 0.3 0 0 1 0 0 0 > * X]
 +       0.0055 * <0.333333 1 1 1 1 0 0 1 0.3 0 0 0 0 0 0 > * X]
 +       0.0055 * <1 0 1 0 1 0 0 1 1 1 0 1 1 0 0 > * X]
 -       0.0058 * <0.666667 0 1 1 0 0 0 1 0 0 1 1 1 0 0 > * X]
 +       0.0047 * <1 0 1 0 1 0 0 1 1 1 0 1 1 0 0 > * X]
                        -0.8445
Number of support vectors:
47
Number of kernel evaluations: 8944 (79.291% cached)
Logistic Regression with ridge parameter of
1.0E-8
Coefficients...
Variable      Coeff.
      1     -32.0189
Intercept     -3.1093
Odds Ratios...
Variable         O.R.
      1       0
```

APPENDIX C (NAÏVE BAYES MODEL OUTPUT)

Naïve Bayes Modeling Output

Trial 1				
Model Properties				
N	No			
S	1			
UseKernel Estimator	**FALSE**			
Model Output				
Accuracy	93.00%	±	Kappa statistic	0.8598
Precision	92.94%	±	Mean absolute error	0.095
Recall	93.20%	±	Root mean squared error	0.2251
Area Under Curve (AUC)	94.30%	±	Relative absolute error	19.11%
			Root relative squared error	45.12%
Positive Class	Contractor			
	True Owner	True Contractor	Class Precision	Class F-Measure
Prediction OWNER	49	2	96.08%	93.33%
Prediction Contractor	5	44	89.80%	92.63%
Class Recall	90.74%	95.65%		

W-NaiveBayes

The word weights for each class are:

	OWNER	CONTRACTOR
Ptype	-1.6981386828507514	-1.6928195213731514
DSCC	-3.19140516352281	-3.4069280563711404
DSC	-2.818729878237636	-4.376328613559243

N&C -3.1596564652082297 -
3.5161273483361324
Conraise -3.258096538021482 -
3.036554268074246
ComImpossible -3.4473385376600105 -
5.675611597689505
Ochange -5.526780079339846 -
4.8283137373023015
Mmistake -3.041873429551846 -
3.3402366818724682
Year -0.8142513750312556 -
0.7011793522572096
Ocause -3.7350206101117913 -
5.387929525237724
SpecWarn -6.625392368007956 -
3.4783870203532854
SpecRep -3.0144744553637315 -
2.9240762846475556
CNoExtra -4.428167790671737 -
3.5553480614894135
Ofalsely -3.917342166905746 -
4.135166556742356
OAdjust -5.932245187448011 -
4.135166556742356
Outcome 0.0 0.0

=== Run information
===

Scheme:
weka.classifiers.bayes.NaiveBayes
Relation: test 1 svm 100 added con
Instances: 100
Attributes: 16
Ptype
DSCC
DSC
N&C
Conraise

ComImpossible
Ochange
Mmistake
Year
Ocause
SpecWarn

SpecRep
CNoExtra
Ofalsely
OAdjust
Outcome

Test mode: 10-fold cross-validation

=== Classifier model (full training set)
===

Naive Bayes Classifier

Class OWNER: Prior probability = 0.54

Ptype: Normal Distribution. Mean = 2.963 StandardDev = 0.8157
WeightSum = 54 Precision = 1.0
DSCC: Normal Distribution. Mean = 0.5185 StandardDev = 0.4997
WeightSum = 54 Precision = 1.0
DSC: Normal Distribution. Mean = 0.1852 StandardDev = 0.3884
WeightSum = 54 Precision = 1.0
N&C: Normal Distribution. Mean = 0.463 StandardDev = 0.4986
WeightSum = 54 Precision = 1.0
Conraise: Normal Distribution. Mean = 0.7593 StandardDev = 0.4275
WeightSum = 54 Precision = 1.0
ComImpossible: Normal Distribution. Mean = 0.037 StandardDev = 0.1889
WeightSum = 54 Precision = 1.0
Ochange: Normal Distribution. Mean = 0.1111 StandardDev = 0.3143
WeightSum = 54 Precision = 1.0
Mmistake: Normal Distribution. Mean = 0.5556 StandardDev = 0.4969
WeightSum = 54 Precision = 1.0
Year: Normal Distribution. Mean = 8.1019 StandardDev = 3.3828
WeightSum = 54 Precision = 1.25
Ocause: Normal Distribution. Mean = 0.0556 StandardDev = 0.2291 WeightSum
= 54 Precision = 1.0
SpecWarn: Normal Distribution. Mean = 0.4815 StandardDev = 0.4997
WeightSum = 54 Precision = 1.0
SpecRep: Normal Distribution. Mean = 0.8519 StandardDev = 0.3552
WeightSum = 54 Precision = 1.0
CNoExtra: Normal Distribution. Mean = 0.4444 StandardDev = 0.4969
WeightSum = 54 Precision = 1.0
Ofalsely: Discrete Estimator. Counts = 45 8 4
(Total = 57)
OAdjust: Normal Distribution. Mean = 0.2407 StandardDev = 0.4275 WeightSum
= 54 Precision = 1.0

Class CONTRACTOR: Prior probability
= 0.46

Ptype: Normal Distribution. Mean = 2.9783 StandardDev = 0.8467
WeightSum = 46 Precision = 1.0
DSCC: Normal Distribution. Mean = 0.6522 StandardDev = 0.4763
WeightSum = 46 Precision = 1.0
DSC: Normal Distribution. Mean = 0.9565 StandardDev = 0.2039
WeightSum = 46 Precision = 1.0
N&C: Normal Distribution. Mean = 0.6739 StandardDev = 0.4688
WeightSum = 46 Precision = 1.0
Conraise: Normal Distribution. Mean = 0.6087 StandardDev = 0.488 WeightSum = 46 Precision = 1.0
ComImpossible: Normal Distribution. Mean = 0.5 StandardDev = 0.5 WeightSum = 46 Precision = 1.0
Ochange: Normal Distribution. Mean = 0.0435 StandardDev = 0.2039
WeightSum = 46 Precision = 1.0
Mmistake: Normal Distribution. Mean = 0.7609 StandardDev = 0.4266
WeightSum = 46 Precision = 1.0
Year: Normal Distribution. Mean = 7.2283 StandardDev = 3.7673
WeightSum = 46 Precision = 1.25
Ocause: Normal Distribution. Mean = 0.3696 StandardDev = 0.4827 WeightSum = 46 Precision = 1.0
SpecWarn: Normal Distribution. Mean = 0 StandardDev = 0.1667
WeightSum = 46 Precision = 1.0
SpecRep: Normal Distribution. Mean = 0.7826 StandardDev = 0.4125
WeightSum = 46 Precision = 1.0
CNoExtra: Normal Distribution. Mean = 0.1739 StandardDev = 0.379
WeightSum = 46 Precision = 1.0
Ofalsely: Discrete Estimator. Counts = 33 15 1
(Total = 49)
OAdjust: Normal Distribution. Mean = 0.0217 StandardDev = 0.1667 WeightSum = 46 Precision = 1.0

Time taken to build model: 0.11
seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	93	93 %
Incorrectly Classified Instances	7	7 %
Kappa statistic	0.8598	
Mean absolute error	0.095	
Root mean squared error	0.2251	

Relative absolute error
19.1053 %
Root relative squared error
45.1206 %
Total Number of Instances 100

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.907	0.043	0.961	0.907	0.933	OWNER
0.957	0.093	0.898	0.957	0.926	CONTRACTOR

=== Confusion Matrix ===

a	b	<-- classified as
49	5	a = OWNER
2	44	b = CONTRACTOR

Trial 2

Model Properties

N	No
S	1
UseKernel Estimator	**TRUE**

Model Output

Accuracy	94.00%	±	Kappa statistic	0.8788
Precision	91.00%	±	Mean absolute error	0.1093
Recall	94.00%	±	Root mean squared error	0.2366
Area Under Curve (AUC)	89.30%	±	Relative absolute error	21.98%
			Root relative squared error	47.42%
Positive Class	Contractor			

	True Owner	True Contractor	Class Precision	Class F-Measure
Prediction OWNER	52	4	92.86%	94.55%
Prediction Contractor	2	42	95.45%	93.33%
Class Recall	96.30%	91.30%		

W-NaiveBayes

The word weights for each class are:

OWNER CONTRACTOR
Ptype -0.031096365007504383 -
0.031238496243336272
DSCC -0.05844110428315457 -
0.06286984994356457
DSC -0.05161666360729127 -
0.08075871244879167
N&C -0.057859721195145555 -
0.06488496238098278
Conraise -0.05966235867492273 -
0.05603514603788319
ComImpossible -0.06312776368273439 -
0.1047350702981274
Ochange -0.10120655774405961 -
0.08909943360176982
Mmistake -0.05570287481654815 -
0.061639158646922515
Year -0.01491059488929836 -
0.012939234386678313
Ocause -0.06839580617032322 -
0.09942632047211497
SpecWarn -0.12132437796402092 -
0.06418852009695351
SpecRep -0.05520114400340014 -
0.0539595301684002
CNoExtra -0.0810887375240983 -
0.065608722997534 45
Ofalsely -0.07173448382725259 -
0.07630842113847014
OAdjust -0.10863144661024585 -
0.07630842113847014
Outcome 0.0 0.0

=== Run information
===

Scheme: weka.classifiers.bayes.NaiveBayes -K
Relation: test 1 svm 100 added con
Instances: 100
Attributes: 16
Ptype

DSCC
DSC
N&C
Conraise

ComImpossible
Ochange
Mmistake
Year
Ocause
SpecWarn
SpecRep
CNoExtra
Ofalsely
OAdjust
Outcome

Test mode: 10-fold cross-validation

=== Classifier model (full training set)
===

Naive Bayes Classifier

Class OWNER: Prior probability = 0.54

Ptype: 4 Normal
Kernels.
StandardDev = 0.4082 Precision = 1.0
Means = 1.0 2.0 3.0 4.0
Weights = 5.0 4.0 33.0
12.0
DSCC: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 26.0 28.0
DSC: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 44.0 10.0
N&C: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 29.0 25.0
Conraise: 2 Normal

Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 13.0 41.0
ComImpossible: 2 Normal Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 52.0 2.0
Ochange: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 48.0 6.0
Mmistake: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 24.0 30.0
Year: 8 Normal
Kernels.
StandardDev = 1.3608 Precision =
1.25
Means = 1.25 2.5 3.75 5.0 6.25 7.5
8.75 11.25
Weights = 1.0 4.0 8.0 3.0 4.0 3.0 6.0
25.0
Ocause: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 51.0 3.0
SpecWarn: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 28.0 26.0
SpecRep: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 8.0 46.0
CNoExtra: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0

Weights = 30.0 24.0
Ofalsely: Discrete Estimator. Counts = 45 8 4
(Total = 57)
OAdjust: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 41.0 13.0

Class CONTRACTOR: Prior probability
= 0.46

Ptype: 3 Normal
Kernels.
StandardDev = 0.2949 Precision = 1.0
Means = 2.0 3.0 4.0
Weights = 17.0 13.0
16.0
DSCC: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 16.0 30.0
DSC: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 2.0 44.0
N&C: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 15.0 31.0
Conraise: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 18.0 28.0
ComImpossible: 2 Normal Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 23.0 23.0
Ochange: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0

Weights = 44.0 2.0
Mmistake: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 11.0 35.0
Year: 8 Normal
Kernels.
StandardDev = 1.4744 Precision =
1.25
Means = 1.25 2.5 3.75 5.0 6.25 7.5
8.75 11.25
Weights = 5.0 7.0 1.0 5.0 1.0 2.0 9.0
16.0
Ocause: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 29.0 17.0
SpecWarn: 1 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0
Weights = 46.0
SpecRep: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 10.0 36.0
CNoExtra: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 38.0 8.0
Ofalsely: Discrete Estimator. Counts = 33 15 1
(Total = 49)
OAdjust: 2 Normal
Kernels.
StandardDev = 0.1667 Precision = 1.0
Means = 0.0 1.0
Weights = 45.0 1.0

Time taken to build model: 0.03
seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 94 94 %
Incorrectly Classified Instances 6 6 %
Kappa statistic 0.8788
Mean absolute error 0.1093
Root mean squared error 0.2366
Relative absolute error 21.9837 %
Root relative squared error 47.4177 %
Total Number of Instances 100

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.963	0.087	0.929	0.963	0.945	OWNER
0.913	0.037	0.955	0.913	0.933	CONTRACTOR

=== Confusion Matrix ===

a	b	<-- classified as
52	2	a = OWNER
4	42	b = CONTRACTOR

APPENDIX D (RULE INDUCTION MODELS OUTPUT)

Rule Induction Modeling Output

Decision Tree

Trial 1				
Model Properties				
Minimum # of Objects (M)	2			
Confidence Factor (C)	0.25			
Binary Split	**FALSE**			
# of Folds	3			
Model Output				
Accuracy	94.00%	±	Kappa statistic	0.8792
Precision	93.96%	±	Mean absolute error	0.0662
Recall	94.00%	±	Root mean squared error	0.2352
Area Under Curve (AUC)	91.20%	±	Relative absolute error	13.31%
			Root relative squared error	47.15%
Positive Class	Contractor			

	True Owner	True Contractor	Class Precision	Class F-Measure
Prediction OWNER	51	3	94.44%	94.44%
Prediction Contractor	3	43	93.48%	93.48%
Class Recall	94.44%	93.48%		

W-J48

J48 pruned tree

DSC <= 0

| Ocause <= 0: OWNER (43.0)

| Ocause > 0: CONTRACTOR (3.0/1.0)

DSC > 0

```
|   SpecWarn <= 0
|   |   CNoExtra <= 0:
CONTRACTOR (36.0)
|   |   CNoExtra > 0
|   |   |   DSCC <= 0
|   |   |   |   Conraise <= 0:
OWNER (4.0/1.0)
|   |   |   |   Conraise > 0:
CONTRACTOR (7.0)
|   |   |   DSCC > 0:
OWNER (2.0)
|   SpecWarn > 0:
OWNER (5.0)

Number of Leaves  :     7

Size of the tree :      13
=== Run information ===
Scheme:
weka.classifiers.trees.J48
-C 0.25 -M 2
Relation:     test 1 svm
100 added con
Instances:    100
Attributes:   16
              Ptype
              DSCC
              DSC
              N&C
              Conraise
              ComImpossible
              Ochange
              Mmistake
              Year
              Ocause
              SpecWarn
              SpecRep
              CNoExtra
              Ofalsely
              OAdjust
              Outcome
Test mode:    10-fold
cross-validation
```

```
=== Classifier model (full
training set) ===

J48 pruned tree
------------------
DSC <= 0
|   Ocause <= 0: OWNER
(43.0)
|   Ocause > 0:
CONTRACTOR (3.0/1.0)
DSC > 0
|   SpecWarn <= 0
|   |   CNoExtra <= 0:
CONTRACTOR (36.0)
|   |   CNoExtra > 0
|   |   |   DSCC <= 0
|   |   |   |   Conraise <= 0:
OWNER (4.0/1.0)
|   |   |   |   Conraise > 0:
CONTRACTOR (7.0)
|   |   |   DSCC > 0:
OWNER (2.0)
|   SpecWarn > 0:
OWNER (5.0)
Number of Leaves  :                 7
Size of the tree :                  13
Time taken to build
model: 0.11 seconds
=== Stratified cross-
validation ===
=== Summary ===
Correctly Classified
Instances          94
94      %
Incorrectly Classified
Instances          6
6      %
Kappa statistic
0.8792
Mean absolute error
0.0662
Root mean squared error
0.2352
```

Relative absolute error 13.3147 %
Root relative squared error 47.149 %
Total Number of Instances 100
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure Class
0.944 0.065 0.944 0.944 0.944 OWNER
0.935 0.056 0.935 0.935 0.935 CONTRACTOR
=== Confusion Matrix ===
a b <-- classified as
51 3 | a = OWNER
3 43 | b = CONTRACTOR

AD Tree

	Trial 1			
Model Properties				
Number of Boosting Iterations	10			
Model Output				
Accuracy	95.00%	±	Kappa statistic	0.9397
Precision	95.33%	±	Mean absolute error	0.0915
Recall	94.88%	±	Root mean squared error	0.1563
Area Under Curve (AUC)	93.20%	±	Relative absolute error	18.39%
			Root relative squared error	31.32%
Positive Class	Contractor			

	True Owner	True Contractor	Class Precision	Class F-Measure
Prediction OWNER	53	4	92.98%	95.50%
Prediction Contractor	1	42	97.67%	94.38%
Class Recall	98.15%	91.30%		

== Run information

===

Scheme: weka.classifiers.trees.ADTree -B 10 -E -3
Relation: test 1 svm 100 added con
Instances: 100
Attributes: 16
Ptype
DSCC
DSC
N&C
Conraise
ComImpossible

Ochange
Mmistake
Year
Ocause
SpecWarn
SpecRep
CNoExtra
Ofalsely
OAdjust
Outcome

Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
Alternating decision tree:
: -0.079
| (1)DSC < 0.5: -1.289
| | (3)ComImpossible < 0.5: -1.793
| | (3)ComImpossible >= 0.5: 1.469
| (1)DSC >= 0.5: 0.778
| | (2)CNoExtra < 0.5: 2.141
| | (2)CNoExtra >= 0.5: -0.902
| | | (7)DSCC < 0.5: 0.177
| | | | (8)Conraise < 0.5: -0.492
| | | | (8)Conraise >= 0.5: 0.55
| | | (7)DSCC >= 0.5: -0.754
| | (5)N&C < 0.5: -0.355
| | (5)N&C >= 0.5: 0.725
| (4)SpecWarn < 0.5: 0.373
| (4)SpecWarn >= 0.5: -1.002
| (6)Ocause < 0.5: -0.44
| (6)Ocause >= 0.5: 0.474
Legend: -ve = OWNER, +ve = CONTRACTOR
Tree size (total number of nodes): 25
Leaves (number of predictor nodes): 17

Time taken to build model: 0.06 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances
97 97 %
Incorrectly Classified
Instances 3 3
%
Kappa statistic
0.9397
Mean absolute error
0.0915
Root mean squared error
0.1563
Relative absolute error
18.3872 %
Root relative squared error
31.3185 %
Total Number of Instances
100
=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision
Recall F-Measure Class 7
0.963 0.022 0.981
0.963 0.972 OWNER
0.978 0.037 0.957
0.978 0.968
CONTRACTOR 13

=== Confusion Matrix ===

a b <-- classified as
52 2 | a = OWNER
1 45 | b = CONTRACTOR

	Trial 2			
Model Properties				
Number of Boosting Iterations	15 and 20			
Model Output				
Accuracy	97.80%	±	Kappa statistic	0.9798
Precision	97.99%	±	Mean absolute error	0.0727
Recall	94.00%	±	Root mean squared error	0.1356
Area Under Curve (AUC)	98.00%	±	Relative absolute error	14.62%
			Root relative squared error	27.19%
Positive Class	Contractor			
	True Owner	True Contractor	Class Precision	Class F-Measure
Prediction OWNER	53	1	98.15%	98.15%
Prediction Contractor	1	45	97.83%	97.83%
Class Recall	98.15%	97.83%		

=== Run information ===

Scheme: weka.classifiers.trees.ADTree -B 15 -E -3
Relation: test 1 svm 100 added con
Instances: 100
Attributes: 16
Ptype
DSCC
DSC
N&C
Conraise
ComImpossible
Ochange
Mmistake
Year
Ocause
SpecWarn
SpecRep

CNoExtra
Ofalsely
OAdjust
Outcome
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
Alternating decision tree:

: -0.079
| (1)DSC < 0.5: -1.289
| | (3)ComImpossible < 0.5: -2.173
| | (3)ComImpossible >= 0.5: 1.756
| (1)DSC >= 0.5: 0.778
| | (2)CNoExtra < 0.5: 2.141
| | (2)CNoExtra >= 0.5: -0.902
| | | (7)DSCC < 0.5: 0.177
| | | | (8)Conraise < 0.5: -0.492
| | | | (8)Conraise >= 0.5: 0.55
| | | | | (9)SpecWarn < 0.5: 0.579
| | | | | (9)SpecWarn >= 0.5: -0.223
| | | | (12)Ocause < 0.5: -0.217
| | | | (12)Ocause >= 0.5: 0.403
| | | (7)DSCC >= 0.5: -0.754
| | | (10)N&C < 0.5: -0.397
| | | (10)N&C >= 0.5: 0.285
| | (5)N&C < 0.5: -0.355
| | (5)N&C >= 0.5: 0.725
| | | (11)SpecWarn < 0.5: 0.476
| | | (11)SpecWarn >= 0.5: -0.164
| (4)SpecWarn < 0.5: 0.373
| (4)SpecWarn >= 0.5: -1.002
| (6)Ocause < 0.5: -0.44

```
|  (6)Ocause >= 0.5: 0.474
Legend: -ve = OWNER, +ve
= CONTRACTOR
Tree size (total number of
nodes): 37
Leaves (number of predictor
nodes): 25

Time taken to build model:
0.05 seconds

=== Stratified cross-validation
===
=== Summary ===

Correctly Classified Instances
99               99      %
Incorrectly Classified
Instances        1               1
%
Kappa statistic
0.9798
Mean absolute error
0.0727                                    7
Root mean squared error
0.1356
Relative absolute error
14.6204 %                                 13
Root relative squared error
27.1865 %
Total Number of Instances
100
=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision
Recall  F-Measure   Class
  1         0.022      0.982     1
0.991    OWNER
  0.978     0          1
0.978     0.989
CONTRACTOR
=== Confusion Matrix ===
  a  b   <-- classified as
 54  0 |  a = OWNER
  1 45 |  b = CONTRACTOR
```

PART

		Trial 1		
Model Properties				
Minimum # of Objects (M)	2			
Confidence Factor (C)	0.25			
Binary Split	**FALSE**			
# of Folds		3		
Model Output				
Accuracy	95.00%	±	Kappa statistic	0.8792
Precision	94.33%	±	Mean absolute error	0.0662
Recall	96.00%	±	Root mean squared error	0.2204
Area Under Curve (AUC)	95.60%	±	Relative absolute error	12.51%
			Root relative squared error	44.17%
Positive Class	Contractor			
	True Owner	True Contractor	Class Precision	Class F-Measure
Prediction OWNER	51	2	96.23%	95.33%
Prediction Contractor	3	44	93.62%	94.62%
Class Recall	94.44%	95.65%		

W-PART

PART decision list

DSC <= 0 AND
Ocause <= 0: OWNER (43.0)
SpecWarn <= 0
AND
CNoExtra <= 0: CONTRACTOR (38.0)
SpecWarn <= 0

AND
DSCC <= 0 AND
Conraise > 0: CONTRACTOR (7.0)
: OWNER (12.0/1.0)
Number of Rules : 4
=== Run information ===
Scheme: weka.classifiers.rules.PART -M 2 -C 0.25
-Q 1
Relation: test 1 svm 100 added
con
Instances: 100
Attributes: 16
Ptype
DSCC
DSC
N&C
Conraise

ComImpossible
Ochange
Mmistake
Year
Ocause
SpecWarn
SpecRep
CNoExtra
Ofalsely
OAdjust
Outcome
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
PART decision list

DSC <= 0 AND
Ocause <= 0: OWNER (43.0)
SpecWarn <= 0 AND
CNoExtra <= 0: CONTRACTOR (38.0)
SpecWarn <= 0 AND
DSCC <= 0 AND
Conraise > 0: CONTRACTOR (7.0)
: OWNER (12.0/1.0)

Number of Rules : 4

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 94 94 %
Incorrectly Classified Instances 6 6 %
Kappa statistic
0.8792
Mean absolute error
0.0622
Root mean squared error
0.2204
Relative absolute error
12.5138 %
Root relative squared error
44.1736 %
Total Number of Instances
100

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure Class
0.944 0.065 0.944 0.944 0.944 OWNER
0.935 0.056 0.935 0.935 0.935 CONTRACTOR
=== Confusion Matrix ===
a b <-- classified as
51 3 | a = OWNER
3 43 | b = CONTRACTOR

APPENDIX E (PARSING ALGORITHM)

```
// CollectionAnalyzer class definition - DocumentAnalyzer class public interface
#include <iostream>
#include <vector>
#include <string>
using namespace std;

// preventing multiple inclusion of the header file
#ifndef COLLECTIONANALYZER_H
#define COLLECTIONANALYZER_H

// defining the DocumentAnalyzer class and prototypes
class CollectionAnalyzer
{
public:

        void setInitialCollection(vector<string>);
        void getInitialCollection()const;
        vector<string> & getInitialCollectionSize();
        void setInitialFrequency(vector<int>);
        void getInitialFrequency() const;
        vector<vector<int>> & getInitialFrequencySize();
        vector<vector<double>> & getpFrequencySize();
        vector<vector<double>> & getpiFrequencySize();
        vector<vector<double>> & getaFrequencySize();
        void print() const;
        void searchLoop(vector<string>, vector<int>, int);
        void approvedMatrix();
        void TermFrequencyWeight();
        void AugmentedTermFrequencyWeight();
        void dfidfCalculation();

private:

        vector <string> iCollection; // a vector of a vector of strings representing the
collection words for each document
        vector <vector<int>> iFrequency; // a vector of a vector of integers storing the
frequency of occurrence of each collection word of each document
        vector <vector<double>> piFrequency; // a vector holding the weighted term
frequencies.
        vector <vector<double>> aFrequency; // a vector holding the augmented
frequencies of terms.
```

```
        vector <string> pCollection; // a vector of a vector of strings representing the
processed matrix of collection words for each document
        vector <vector<double>> pFrequency; // a vector of a vector of integers
storing the processed dfidf frequency of occurrence of each collection word of each
document
        vector <int> dfVector; // a vector including document frequency of terms.
        vector <int> NVector; // a vector including collection numbers.
        void addDummyVector();
        void matricAdjustment();
};

        #endif

        ---------------------------------------------------------------------------------------

// DocumentAnalyzer class definition - CollectionAnalyzer class public interface
#include <iostream>
#include <vector>
#include <string>
using namespace std;
#include "CollectionAnalyzer.h"
// preventing multiple inclusion of the header file
#ifndef PROJECT_H
#define PROJECT_H
// defining the DocumentAnalyzer class and prototypes
class DocumentAnalyzer
{
public:
        DocumentAnalyzer(string="00");
        ~DocumentAnalyzer();

        void setOriginalString(string);
        void setWordsVector();
        void getWordsVector()const;
        vector<string> & getWordsVectorSize();
        void setDocumentWordCount();
        int getDocumentWordCount ()const;
        void setDocumentSentencesCount();
        int getDocumentSentencesCount() const;
        void setWordSignificance();
        void getWordSignificance()const;
        void setStartEndCharacters();
        void setUnecessaryWords();
        void setEndOfSentence();
        void setPrefix();
```

```
        void setCapitalLetters();
        void documentProcessing();
        vector<string> & getInitialWordList();
        void setApprovedValidTermsandWordCount ();
        vector<string> & getApprovedValidTerms();
        vector<int> & getApprovedWordCount();
        void getSentences() const;
        void getValidTerms() const;
        void getWordCount() const;
        vector<string> & getValidTermsSize();
        vector<int> & getWordCountSize();
private:
        int DocumentWordCount; // integer that holds the number of words in a
provided text
        int documentSentencesCount; // integer that holds the number of sentences
in a provided text
        string originalString; // string that intakes the passed string to be processed
        vector <string> words; // a vector that holds all words in the passed text
        vector <vector<string>> sentences; // a vector that holds all sentences of the
passed text
        vector <string> validTerms; // a vector that holds word objects. It includes
words to be further processed
        vector <int> wordCount; // a vector that hold the number of occurancec of
each word int he valid terms
        vector<string> approvedValidTerms; // accepts valid terms that were repeated
more than a certain number of times
        vector<int> approvedWordCount; // accepts valid terms counts that were
repeated more than a certain number of times
        vector<string> startEndCharacters;// a vector that holds characters to be
removed from the start and end of word
        vector<string> unecessaryWords;// a vector that includes words to be
removed from the text before processing
        vector<string> endOfSentence;// a vector that includes strings considered to
be end of sentence characters
        vector<string> prefix;// a vector including most known prefixes
        vector<string> capitalLetters;// a vector inlcuding a set of all 26 in the upper
case form.
        vector<double> wordSignificance; // a vector of doubles representing the
signifcance of each repeated term
        int startingcharacter (string &); // a utility function that removes starting
characters
        int endingCharacter (string &); // a utility function that removes ending
characters
        int possisveCheck (string &); // a utility function that removes possisive
characters
```

```
        int pluralCheck (string &, int); // a utility function that changes a plural forms of
a word
        int checkPlural (string &);
        int endOfSentenceCheck (string &, int); // a utility function that defines he end
of sentence within a text
        int unecessaryWordsCheck (string &); // a utility function that removes
unwanted wards from the text
        void upperToLower (string &); // a utility function that converts all upper case
letters to lower ones
        void sorting (vector<string> &, vector<int> &); // a utility function that performs
a sorting algorithm
};
        #endif

        ---------------------------------------------------------------------------------

// the code utilizes the input/output standard stream, vector, and standard string
classes
#include <fstream> // file stream
using std::ifstream; // input file stream
using std::ofstream; // output file stream
#include <iomanip>
#include <cstdlib>
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
using namespace std;
#include <iostream>
#include <vector>
#include <string>
using namespace std;
//Including header files of DocumentAnalyzer and Word classes
#include "DocumentAnalyzer.h"
#include "CollectionAnalyzer.h"
// declairing member functions
void CollectionAnalyzer::setInitialCollection(vector<string> iC)
{
        for(int i=0; i<iC.size(); i++)
        {
                iCollection.push_back(iC[i]);
        }
}
void CollectionAnalyzer::getInitialCollection()const
{
```

```
	for(int i=0; i<iCollection.size(); i++)
	{
		cout<<iCollection[i]<<endl;
	}
}
vector<string> & CollectionAnalyzer::getInitialCollectionSize()
{
	return iCollection;
}
void CollectionAnalyzer::setInitialFrequency(vector<int> iF)
{
	iFrequency.push_back(iF);
}
void CollectionAnalyzer::getInitialFrequency()const
{
	for(int i=0; i<iFrequency.size(); i++)
	{
		for(int j=0; j<iFrequency[i].size(); j++)
		{
			cout<<iFrequency[i][j]<<" ";
		}
		cout<<endl;
	}
}
vector<vector<int>> & CollectionAnalyzer::getInitialFrequencySize()
{
	return iFrequency;
}
vector<vector<double>> & CollectionAnalyzer::getpFrequencySize()
{
	return pFrequency;
}
vector<vector<double>> & CollectionAnalyzer::getpiFrequencySize()
{
	return piFrequency;
}
vector<vector<double>> & CollectionAnalyzer::getaFrequencySize()
{
	return aFrequency;
}
void CollectionAnalyzer::print() const
{
	for(int n=0; n<iCollection.size(); n++)
	{
		cout<<left<<setw(25)<<iCollection[n]<<" ";
```

```
                for(int m=0; m<iFrequency.size(); m++)
                {
                        cout<<left<<setw(10)<<iFrequency[m][n]<<" ";
                }
                cout<<endl;
        }
}
void CollectionAnalyzer::addDummyVector()
{
        vector<int> tempVector;
        for(int i=0; i<iCollection.size(); i++)
        {
                tempVector.push_back(0);
        }
        iFrequency.push_back(tempVector);
        tempVector.clear();
}
void CollectionAnalyzer::matricAdjustment()
{
        for(int i=0; i<iFrequency.size(); i++)
        {
                int missingData=iCollection.size()- iFrequency[i].size();
                for(int j=0; j<missingData; j++)
                {
                        iFrequency[i].push_back(0);
                }
                missingData=0;
        }
}
void CollectionAnalyzer::searchLoop(vector<string> iC, vector<int> iF, int counter)
{
        int tempIndex=0;
        bool tempBool=false;
        addDummyVector();
        for(int i=0; i<iC.size(); i++)
        {
                tempIndex=0;
                for(int j=0; j<iCollection.size(); j++)
                {
                        if(iC[i]==iCollection[j])
                        {
                                tempIndex=j;
                                tempBool=true;
                        }
                }
```

```
                if(tempBool==true)
                {
                        iFrequency[counter-1][tempIndex]=iF[i];
                }
                if(tempBool==false)
                {
                        iCollection.push_back(iC[i]);
                        iFrequency[counter-1].push_back(iF[i]);
                }
                tempBool=false;
        }
        matricAdjustment();
}
void CollectionAnalyzer::approvedMatrix()
{
        vector<int> sumOverDocuments;
        vector<string> tempICollection;
        int sum=0;

        for(int i=0; i<iCollection.size(); i++)
        {
                for(int j=0; j<iFrequency.size(); j++)
                {
                        sum=sum+iFrequency[j][i];
                }

                sumOverDocuments.push_back(sum);
                sum=0;
        }

        for(int v=0; v<iCollection.size(); v++)
        {
                tempICollection.push_back(iCollection[v]);
        }

        int turn=0;
        bool first=false;
        for(int k=0; k<sumOverDocuments.size(); k++)
        {
                if(sumOverDocuments[k]<3)
                {
                        if(k==0)
                        {
                                first=true;
                                iCollection.erase(iCollection.begin());
```

```
                        for(int t=0; t<iFrequency.size(); t++)
                                {
                                        iFrequency[t].erase(iFrequency[t].begin());
                                }
                        }
                        if(first==true)
                        {
                                if(iCollection[0]==tempICollection[k])
                                {
                                        iCollection.erase(iCollection.begin());
                                        for(int l=0; l<iFrequency.size(); l++)
                                        {
                                                iFrequency[l].erase(iFrequency[l].begin());
                                        }
                                }
                                else
                                {
                                        iCollection.erase(iCollection.begin()+k-1);
                                        for(int l=0; l<iFrequency.size(); l++)
                                        {
                                                iFrequency[l].erase(iFrequency[l].begin()+k-
1);
                                        }
                                }
                        }
                        if(first==false)
                        {
                                iCollection.erase(iCollection.begin()+(k-turn));
                                for(int l=0; l<iFrequency.size(); l++)
                                        {

        iFrequency[l].erase(iFrequency[l].begin()+(k-turn));
                                        }
                                turn++;
                        }
                }
        }
}
// dfidf calculations function
void CollectionAnalyzer::dfidfCalculation()
{
        vector <double> tempdVector;
        vector <double> tempNVector;
        int dfCounter=0;
```

```
for(int i=0; i<iCollection.size(); i++)
{
	for(int j=0; j<iFrequency.size(); j++)
	{
		if(iFrequency[j][i]>0)
		{
			dfCounter++;
		}
	}
	dfVector.push_back(dfCounter);
	NVector.push_back(iFrequency.size());
	dfCounter=0;
}

for(int t=0; t<iCollection.size(); t++)
{
	double dN=0.0;
	double dF=0.0;
	double tempf=0.0;
	dN=static_cast< double >(NVector[t]);
	dF=static_cast< double >(dfVector[t]);
	tempf=log10(dN)-log10(dF);
	tempdVector.push_back(tempf);
}

vector <double> tempPFrequency;
for(int r=0; r<iCollection.size(); r++)
{
	tempPFrequency.push_back(0.0);
}
for(int u=0; u<piFrequency.size(); u++)
{
	pFrequency.push_back(tempPFrequency);
}
tempPFrequency.clear();

double pf=0.0;
for(int x=0; x<iCollection.size(); x++)
{
	for(int z=0; z<piFrequency.size(); z++)
	{
		if(iFrequency[z][x]>0)
		{
			pf=piFrequency[z][x]*tempdVector[x];
			pFrequency[z][x]=pf;
```

```
                        }
                }
        }
}
static double Log10(double d);
void CollectionAnalyzer::TermFrequencyWeight()
{
        double tempf=0.0;
        double l=0.0;
        vector <double> tempPiFrequency;
        for(int r=0; r<iCollection.size(); r++)
        {
                tempPiFrequency.push_back(0.0);
        }
        for(int u=0; u<iFrequency.size(); u++)
        {
                piFrequency.push_back(tempPiFrequency);
        }
        tempPiFrequency.clear();

        for (int i=0; i<iCollection.size(); i++)
        {
                for(int j=0; j<iFrequency.size(); j++)
                {
                        if(iFrequency[j][i]>0)
                        {
                                tempf = static_cast< double >(iFrequency[j][i]);
                                l=log10(tempf);
                                piFrequency[j][i]=1+l;
                        }
                }
        }
}
void CollectionAnalyzer::AugmentedTermFrequencyWeight()
{
        double tempaf=0.0;
        double l=0.0;
        vector <double> tempaiFrequency;
        for(int r=0; r<iCollection.size(); r++)
        {
                tempaiFrequency.push_back(0.0);
        }
        for(int u=0; u<iFrequency.size(); u++)
        {
                aFrequency.push_back(tempaiFrequency);
```

```
}
tempaiFrequency.clear();

int maxFrequency=0;
vector <int> tempMaxFrequency;
for (int i=0; i<iCollection.size(); i++)
{
        for(int j=0; j<iFrequency.size(); j++)
        {
                if(iFrequency[j][i]>maxFrequency)
                {
                        maxFrequency=iFrequency[j][i];
                }
                tempMaxFrequency.push_back(maxFrequency);
        }
}

double tempA=0.0;
double tempAugmentedFrequency=0.0;
for (int c=0; c<iCollection.size(); c++)
{
        for (int h=0; h<iFrequency.size(); h++)
        {
                tempA=0.5+((0.5*iFrequency[h][c])/tempMaxFrequency[c]);
                aFrequency[h][c]=tempA;
        }
}
}

-------------------------------------------------------------------------------------------
```

```
// DocumentAnalyzer member-function definitions - DocumentAnalyzer class
member-function implementation
// the code utilizes the input/output standard stream, vector, and standard string
classes
#include <fstream> // file stream
using std::ifstream; // input file stream
using std::ofstream; // output file stream
#include <iomanip>
#include <cstdlib>
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
using namespace std;
```

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
//Including header files of DocumentAnalyzer and Word classes
#include "DocumentAnalyzer.h"
//Constructor that takes in as an argument the initial string to set its initial private
data members
DocumentAnalyzer::DocumentAnalyzer(string s)
{
	setOriginalString(s);
	setWordsVector();
	setDocumentWordCount();
	setDocumentSentencesCount();
	setStartEndCharacters();
	setUnecessaryWords();
	setEndOfSentence();
	setPrefix();
	setCapitalLetters();
}
DocumentAnalyzer::~DocumentAnalyzer()
{
}
// a set function for the initial string
void DocumentAnalyzer::setOriginalString(string s)
{
	originalString = s;
}
void DocumentAnalyzer::setWordsVector()
{
	int indexOfSpace;
	string word;

	for(int i=0; i<originalString.length(); i++)
	{
		indexOfSpace=originalString.find(" ");
		word=originalString.substr(0,indexOfSpace);
		if(indexOfSpace>0)
		{
			words.push_back(word);

	originalString=originalString.substr(indexOfSpace+1,originalString.length()-1);
			i=0;
		}
		else
```

```
                {

        originalString=originalString.substr(indexOfSpace+1,originalString.length()-1);
                        i=0;
                }
        }
}
//a get function that prints out the words of a document
void DocumentAnalyzer::getWordsVector() const
{
        for(int k=0; k<words.size(); k++)
                cout<<words[k]<<endl;
}
vector<string> & DocumentAnalyzer::getWordsVectorSize()
{
        return words;
}
// a set function to set the private data member DocumentWordCount
void DocumentAnalyzer::setDocumentWordCount()
{
        DocumentWordCount = words.size();
}
// a get function that returns the number of words in a text
int DocumentAnalyzer::getDocumentWordCount() const
{
        return DocumentWordCount;
}
// a get function to return an aliace of the valid terms vector
vector<string> & DocumentAnalyzer::getValidTermsSize()
{
        return validTerms;
}
// a get function to return an aliace of the wordCount vector
vector<int> & DocumentAnalyzer::getWordCountSize()
{
        return wordCount;
}
// a set function to set the private data memebr startEndCharacters vector
void DocumentAnalyzer::setStartEndCharacters()
{
        ifstream inCharFile( "startendchar.txt", ios::in ); // declairing the output file
        // exit program if ifstream could not open file
        if ( !inCharFile )
        {
                cerr << "File could not be opened" << endl;
```

```
                exit( 1 );
        } // end if

        string end;
        while(inCharFile>>end)
        {
                startEndCharacters.push_back(end);
        }
}
// a set function to set the private data memebr unecessaryWords vector
void DocumentAnalyzer::setUnecessaryWords()
{
        ifstream inUnWordFile( "unecessaryWords.txt", ios::in ); // declairing the
output file
        // exit program if ifstream could not open file
        if ( !inUnWordFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if

        string unword;

        while(inUnWordFile>>unword)
        {
                unecessaryWords.push_back(unword);
        }
}
// a set function to set the private data memebr endOfSentence vector
void DocumentAnalyzer::setEndOfSentence()
{
        ifstream inEndSentFile( "endsentence.txt", ios::in ); // declairing the output file
        // exit program if ifstream could not open file
        if ( !inEndSentFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        string endsent;
        while(inEndSentFile>>endsent)
        {
                endOfSentence.push_back(endsent);
        }
}
// a set function to set the private data memebr prefix vector
```

```
void DocumentAnalyzer::setPrefix()
{
	ifstream inPrefixFile( "prefix.txt", ios::in ); // declairing the output file
	// exit program if ifstream could not open file
	if ( !inPrefixFile )
	{
		cerr << "File could not be opened" << endl;
		exit( 1 );
	} // end if
	string pref;
	while(inPrefixFile>>pref)
	{
		prefix.push_back(pref);
	}
}
// a set function to set the private data memebr capitalLetters vector
void DocumentAnalyzer::setCapitalLetters()
{
	ifstream inCapFile( "capittalletters.txt", ios::in ); // declairing the output file
	// exit program if ifstream could not open file
	if ( !inCapFile )
	{
		cerr << "File could not be opened" << endl;
		exit( 1 );
	} // end if
	string cap;
	while(inCapFile>>cap)
	{
		capitalLetters.push_back(cap);
	}
}
// a set function to set the private data memebr documentSentencesCount, which
represents the number of sentences within a text
void DocumentAnalyzer::setDocumentSentencesCount()
{
	documentSentencesCount=0;
}
// a get functionthat returns the private data memebr documentSentencesCount,
which represents the number of sentences within a text
int DocumentAnalyzer::getDocumentSentencesCount() const
{
	return documentSentencesCount;
}
// setting the Word Significance vector
void DocumentAnalyzer::setWordSignificance()
```

```
{
	double sum=0.0;
	for(int i=0; i<validTerms.size(); i++)
	{
		sum=sum+wordCount[i];
	}
	for(int s=0; s<validTerms.size(); s++)
	{
		double temp=0.0;
		temp=(((static_cast<double>(wordCount[s])/sum))*100);
		wordSignificance.push_back(temp);
	}
}
void DocumentAnalyzer::getWordSignificance() const
{
	for(int i=0; i<wordSignificance.size(); i++)
	{
		cout<<wordSignificance[i];
	}
}
// a utility function that removes starting characters. It takes string by reference and
returns an integer
int DocumentAnalyzer::startingcharacter(string & str)
{
	int counter=0;
	string sub="00";
	sub=str.substr(0,1);

	for(int i=0; i< startEndCharacters.size(); i++) // a loop to check if the first lette
in the word is an unwatnted starting character
	{
		if (sub==startEndCharacters[i])
		{
			counter=1; // if the first letter in the word is an unwatnted starting
character a counter is set to 1
			str=str.substr(1,str.length()-1);
		}
	}
	if (counter == 1)
		return 1; // the function returns 1 if the first lette in the word is an
unwatnted starting character
	else
		return 0; // the function returns 0 if the first lette in the word is not an
unwatnted starting character
}
```

```
// a utility function that removes ending characters. It takes string by reference and
returns an integer
int DocumentAnalyzer::endingCharacter (string & str)
{
	int counter=0;
	string sub="00";
	sub=str.substr(str.length()-1,1);

	for(int i=0; i< startEndCharacters.size(); i++) // a loop to check if the last lette
in the word is an unwatnted ending character
	{
		if (sub==startEndCharacters[i])
		{
			counter=1; // if the last lette in the word is an unwatnted ending
character a counter is set to 1
			str=str.substr(0,str.length()-1);
		}
	}
	if (counter == 1)
		return 1; // the function returns 1 if the last lette in the word is an
unwatnted ending character
	else
		return 0; // the function returns 0 if the last lette in the word is not an
unwatnted ending character
}
// a utility function that removes possisive characters. It takes string by reference and
returns an integer
int DocumentAnalyzer::possisveCheck(string & str)
{
	int counter=0;
	if (str.length()>1)
	{
		string last="00";
		string beforelast="00";
		last=str.substr(str.length()-1,1);
		beforelast=str.substr(str.length()-2,1);
		if (last=="s") // nested if conditions to test if the last two letters of a
word are 's
		{
			if (beforelast=="'"||beforelast=="’"||beforelast=="‘")
			{
				counter=1;
				str=str.substr(0,str.length()-2);
			}
		}
```

```
        }
        if (counter == 1)
                return 1; // if the last two letters of a word are 's, the function returns 1
        else
                return 0; // if the last two letters of a word are not 's, the function
returns 0
}
// a utility function that changes a plural forms of a word. It takes a sting by reference
and returns an integer
int DocumentAnalyzer::pluralCheck(string & str, int pos)
{
        static vector<string> temp;
        string tempString="00", tempStringIes="00";
        int counter=0;
        if(str.length()>1)
        {
                string last="00", last2="00";
                last=str.substr(str.length()-1,1);
                last2=str.substr(str.length()-2,2);

                if (last=="s"&&last2!="ss")
                {
                        bool partOfUnecessaryWors=true;

                        for(int i=0; i<unecessaryWords.size(); i++)
                        {
                                if(str==unecessaryWords[i])
                                {
                                        partOfUnecessaryWors=false;
                                        break;
                                }
                        }
                        if(partOfUnecessaryWors==true) // if the last letter of a word is
s, the user is prompter to define if the word is in the plural form or not. If yes, he is
prompted to enter the singular form.
                        {
                                bool pluralCheckBool=true;
                                if(temp.size()>=1)
                                {
                                        for(int v=0; v<temp.size(); v++)
                                        {
                                                if(str==temp[v])
                                                {
                                                        pluralCheckBool=false;
                                                        str=temp[v+1];
```

```
                                        break;
                                }
                        }
                }
                if(pluralCheckBool==true)
                {
                        char choice='0', confirm='0';

                        cout<< "Is the following word in the plural form?
<"<< str <<">"<<endl;

                        if(words.size()>=3)
                        {
                                if(pos==0)
                                        cout<< "The Word was mentioned in
the following context <"<<words[pos]<<" "<<words[pos+1]<<"
"<<words[pos+2]<<">."<<endl;
                                if(pos>0&&pos<words.size()-1)
                                        cout<< "The Word was mentioned in
the following context <"<<words[pos-1]<<" "<<words[pos]<<"
"<<words[pos+1]<<">."<<endl;
                                if(pos==words.size()-1)
                                        cout<< "The Word was mentioned in
the following context <"<<words[pos-2]<<" "<<words[pos-1]<<"
"<<words[pos]<<">."<<endl;
                        }

                        cout<< "Please enter the appropriate number
corresponding to your choice\n"
                                << "<y> for Yes\n" << "<n> for No\n" <<
endl;
                        cin>>choice;

        while(choice!='y'&&choice!='n'&&choice!='Y'&&choice!='N')
                        {
                                cout<<"You have entered an invalid
choice.\n"<<endl;
                                cout<<"Please limit your choice between
<y> or <n>"<<endl;
                                cin>>choice;
                        }
                        if(choice=='y'||choice=='Y')
                        {
                                temp.push_back(str);
                                tempString=str.substr(0,str.length()-1);
```

```
                                        tempStringIes=str.substr(str.length()-3,3);
                                        if(tempStringIes=="ies")
                                        {
                                                tempString=str.substr(0,str.length()-
3);
                                                tempString.append("y");
                                        }
                                        cout<<"Is this the singular form of the
word? <"<<tempString<<">\n"
                                                <<"Please enter the appropriate
number corresponding to your choice\n"
                                                << "<y> for Yes\n" << "<n> for No\n"
<< endl;
                                        cin>>confirm;

        while(confirm!='y'&&confirm!='n'&&confirm!='Y'&&confirm!='N')
                                        {
                                                cout<<"You have entered an invalid
choice.\n"<<endl;
                                                cout<<"Please limit your choice
between <y> and <n>"<<endl;
                                                cin>>confirm;
                                        }
                                        if(confirm=='y'||confirm=='Y')
                                        {
                                                temp.push_back(tempString);
                                                str=tempString;
                                        }
                                        else
                                        {
                                                string newWord="00";
                                                char check='0';
                                                cout<< "Please enter the singular
form from the previouse word with no spaces in between\n" << endl;
                                                cin>>newWord;
                                                cout<<"Is the word you have entered
is <"<<newWord<<">"<<endl;
                                                cout<<"Please enter your choice
below:\n" <<"<y> for Yes\n" << "<n> for No\n" <<endl;
                                                cin>>check;

        while(check!='y'&&check!='n'&&check!='Y'&&check!='N')
                                                {
                                                        cout<<"You have entered an
invalid choice.\n"<<endl;
```

```
                                                cout<<"Please limit your
choice between <y> and <n>"<<endl;
                                                cin>>check;
                                        }
                                        while(check=='n'||check=='N')
                                        {
                                                cout<< "Please enter the
singular form from the previouse word with no spaces in between\n" << endl;
                                                cin>>newWord;
                                                cout<<"Is the word you have
entered is <"<<newWord<<">"<<endl;
                                                cout<<"Please enter your
choice below:\n" <<"<y> for Yes\n" << "<n> for No\n"<<endl;
                                                cin>>check;

        while(check!='y'&&check!='n'&&check!='Y'&&check!='N')
                                                {
                                                        cout<<"You have
entered an invalid choice.\n"<<endl;
                                                        cout<<"Please limit
your choice between <y> and <n>"<<endl;
                                                        cin>>check;
                                                }
                                        }
                                        temp.push_back(newWord);
                                        str=newWord;
                                        counter=1;
                                }
                        }
                        if(choice=='n'||choice=='N')
                        {
                                tempString=str;
                                temp.push_back(str);
                                temp.push_back(tempString);
                        }
                }
        }
        }
        if (counter == 1)
        {
                return 1; // if the form of the word was changed, the memeber function
returns a 1
        }
        else
```

```
		return 0; // if the form of the word was not changed, the memeber function returns a 0
}
// a utility function that changes a word from its plural form into its singular form without user's feedback. Returns 1
// the tested word has changed, returns 0 if it is unchanged.
int DocumentAnalyzer::checkPlural(string & word)
{
	bool isUnecessary=false;
	bool hasChanged=false;
	for(int i=0; i<unecessaryWords.size();i++)
	{
		if(unecessaryWords[i]==word)
		{
			isUnecessary=true;
			break;
		}
	}
	if(!isUnecessary)
	{
		if(word.substr(word.length()-1,1)=="s")
		{
			if(word.substr(word.length()-2,2)!="as"&&word.substr(word.length()-2,2)!="is"&&word.substr(word.length()-2,2)!="os"
				&&word.substr(word.length()-2,2)!="us"&&word.substr(word.length()-2,2)!="ss")
			{
				if(word.substr(word.length()-3,3)=="ies")
					word=word.substr(0,word.length()-3)+"y";
				else if(word.substr(word.length()-2,2)=="es")
				{
					if(word.substr(word.length()-4,4)=="sses"||word.substr(word.length()-3,3)=="xes")
						word=word.substr(0,word.length()-2);
					else
						word=word.substr(0,word.length()-1);
				}
				else
					word=word.substr(0,word.length()-1);

				hasChanged=true;
			}
		}
	}
```

```
        if(hasChanged)
                return 1;
        else
                return 0;
}
// a utility function the tests is a word is at the ned of the sentence or not. It takes a
tring by reference as argument and returns an integer
int DocumentAnalyzer::endOfSentenceCheck(string & str, int pos)
{
        int counter=0;

        if(str.length()>1)
        {
                string last="00";
                bool endOfSentenceBool=true;
                last=str.substr(str.length()-1, 1);

                for (int i=0; i<endOfSentence.size(); i++) // aloop to test if the last letter
of the word is considered as an end of sentence character
                {
                        if(last==endOfSentence[i])
                        {
                                endOfSentenceBool=false;
                        }
                }
                if(endOfSentenceBool==false)
                {
                        string newWord="00";
                        counter=1;
                        newWord=str.substr(0, str.length()-1);
                        str=newWord; // modifing the passed argument by removing the
end of sentence chracter

                        if(last==".") // testing if the end of sentence was a period or not
                        {
                                for(int k=0; k<prefix.size(); k++) // making sure thatthe
period was not used for a prefix
                                {
                                        if(str==prefix[k])
                                                counter=0;
                                }
                                if(counter==1 && words.size()>(pos+1))
                                {
                                        for(int h=0; h<words[pos+1].length(); h++)
                                        {
```

```
                                        string q="00";
                                        q=words[pos+1].substr(0,1);
                                        for(int g=0; g<startEndCharacters.size();
g++)
                                        {
                                                if(q==startEndCharacters[g])
                                                {

        words[pos+1]=words[pos+1].substr(1, words[pos+1].length()-1);
                                                        break;
                                                }
                                        }
                                }
                        }
                        if(counter==1 && words.size()>(pos+1)) // making sure
thatthe period was not used for abbreviation
                        {
                                string first="00";
                                first=words[pos+1].substr(0, 1);
                                for(int z=0; z<capitalLetters.size(); z++)
                                {
                                        if(first==capitalLetters[z])
                                        {
                                                counter=1;
                                                break;
                                        }
                                        else
                                                counter=0;
                                }
                        }
                }
        }
        }
        if(counter==1)
                return 1; //if the  last letter was an end of sentence, the function returns
1
        else
                return 0; //if the  last letter was not an end of sentence, the function
returns 0
}
// a utility function to check if the word is an unwanted word or not. The function
takes a string as an argument and returns an integer
int DocumentAnalyzer::unecessaryWordsCheck(string & str)
{
        int counter=0;
```

```
	for(int i=0; i<unecessaryWords.size(); i++) // a loop to check if the word is considere as an unecessary word or not
	{
		if(str==unecessaryWords[i])
			counter=1;
	}

	if(counter==1)
		return 1; // if the word was found to be unecessary, the function returns 1
	else
		return 0; // if the word was not found to be unecessary, the function returns 0
}
// a member function that utilizes the diffeent utility functions of the DocuentAnalyzer class to process all words and fill the following private data member
// vector <vector<string>> sentences that holds all sentences of the passed text
// vector <Word> validTerms that holds word objects. It includes words to be further processed
void DocumentAnalyzer::documentProcessing()
{
	vector<string> temp;
	vector <string> tempValidTerms;
	int termCounter=0;

	for(int i=0; i<words.size(); i++) // a loop that iterates through the vector of all words
	{

		bool fullProcess=true;
		bool endOfSentenceBool=true;
		int wordValidation=0, size=0;

		upperToLower(words[i]); // converting all upper case letters to lower ones. This is to make sure that if a word is included more than once with lower and upper case letters, they will be treated the same.
		// code that defines string senseNum and boolean isTagged=false, and checks if word[i] contains '\'
		// if word[i] contains '\', split word[i] at '\' into word[i] and senseNum and make isTagged=true
		// a loop used to make sure that the edited word has undergone all required processing aspects and is ready to be included in the rest of the private data members.
```

```
                // the choosen order of pocessing is set in the follwoing manner to save
processing time.
                while(fullProcess==true)
                {
                        bool startCharacter=true, endCharacter=true, possisive=true,
plural=true, sentenceTest=true;
                        int sC=0, eC=0, pS=0, pL=0, eS=0;
                        sC=startingcharacter(words[i]); // processing the word for
starting characters
                        if(sC==1)
                                startCharacter=false;
                        eC=endingCharacter(words[i]); // processing the word for ending
characters
                        if(eC==1)
                                endCharacter=false;
                        pS=possisveCheck(words[i]); // processing the word for possive
check
                        if(pS==1)
                                possisive=false;
                        pL=checkPlural(words[i]); // processing the word for plural check
                        if(pL==1)
                                plural=false;
                        eS=endOfSentenceCheck(words[i], i); // testing if the word is at
the end of a sentence
                        if(eS==1)
                        {
                                sentenceTest=false;
                                endOfSentenceBool=false;
                        }

        if(startCharacter==true&&endCharacter==true&&possisive==true&&plural==tr
ue&&sentenceTest==true)
                        {
                                fullProcess=false;
                        }
                }
                // code that appends to word[i] its senseNum before inserting word[i]
into the sentences vector of vector and
                // the validTerms vector
                temp.push_back(words[i]); // pushing back the word into a local vector
                if(endOfSentenceBool==false) // testing if the word was at the end of
sentence or not.
                {
                        vector <int> sentnecCheckVector;
                        int sum=0;
```

```
                documentSentencesCount++;
                for(int q=0; q<sentences.size(); q++)
                {
                        int sentenceEquality=0;
                        if(temp.size()==sentences[q].size())
                        {
                                for(int w=0; w<sentences[q].size(); w++)
                                {
                                        if(temp[w]!=sentences[q][w])
                                        {
                                                sentenceEquality=1;
                                                break;
                                        }
                                }

        sentnecCheckVector.push_back(sentenceEquality);
                        }
                }
                if(sentnecCheckVector.size()>=1)
                {
                        for(int e=0; e<sentnecCheckVector.size(); e++)
                                sum=sum+sentnecCheckVector[e];
                        if(sum==sentnecCheckVector.size())
                        {
                                sentences.push_back(temp); // if the word was at
the end of a sentence, the local vector is pushed back into the private data member
                                temp.clear();
                        }
                        else
                                temp.clear();
                }
                else
                {
                        sentences.push_back(temp); // if the word was at the end
of a sentence, the local vector is pushed back into the private data member
                        temp.clear();
                }
        }
        wordValidation=unecessaryWordsCheck(words[i]); // performing the
unecessary word check
        if(wordValidation==0) // checking if the word is a valid one or not
        {
                bool validTermCheck=true;
                for(int y=0; y<tempValidTerms.size(); y++)
                {
```

```
                        if(words[i]==tempValidTerms[y])
                                validTermCheck=false;
                }
                if(validTermCheck==true)
                        tempValidTerms.push_back(words[i]); // if the word is a
valid term; it is inlcuded into a local vector.
                }
        }
        for(int a=0; a<tempValidTerms.size(); a++) // a loop to find the number of
repetitions of the valid word
        {
                for(int b=0; b<words.size(); b++)
                {
                        if(tempValidTerms[a]==words[b])
                                termCounter++;
                }
                validTerms.push_back(tempValidTerms[a]); // including the created
object into the private data member
                wordCount.push_back(termCounter);
                termCounter=0;
        }
        temp.clear();
        tempValidTerms.clear();
        sorting(validTerms, wordCount); // performing a sort algorithm for the private
data memeber that includes instances of objects of the Word class
        setApprovedValidTermsandWordCount();
}
// a utility member function that converts all upper caseletters to lower ones. This is
to make sure that if a word is included more than once with lower and upper case
letters, they will be treated the same.
void DocumentAnalyzer::upperToLower(string & str)
{
        string tempString;
        tempString.clear();
        int stringSize=0;
        stringSize=str.length();
        for(int t=0; t<stringSize; t++) //a loop to iterate through a stringconvering all
upper case letters to lower ones
        {
                string sub="00";
                sub=str.substr(t,1);
                if(sub=="A")
                        sub="a";
                if(sub=="B")
                        sub="b";
```

```
if(sub=="C")
        sub="c";
if(sub=="D")
        sub="d";
if(sub=="E")
        sub="e";
if(sub=="F")
        sub="f";
if(sub=="G")
        sub="g";
if(sub=="H")
        sub="h";
if(sub=="I")
        sub="i";
if(sub=="J")
        sub="j";
if(sub=="K")
        sub="k";
if(sub=="L")
        sub="l";
if(sub=="M")
        sub="m";
if(sub=="N")
        sub="n";
if(sub=="O")
        sub="o";
if(sub=="P")
        sub="p";
if(sub=="Q")
        sub="q";
if(sub=="R")
        sub="r";
if(sub=="S")
        sub="s";
if(sub=="T")
        sub="t";
if(sub=="U")
        sub="u";
if(sub=="V")
        sub="v";
if(sub=="W")
        sub="w";
if(sub=="X")
        sub="x";
if(sub=="Y")
```

```
                sub="y";
            if(sub=="Z")
                sub="z";
            tempString.append(sub);
        }
        str=tempString; // modifyingthe initial passed string
}
// a utility function to perform a sorting algorithm
void DocumentAnalyzer::sorting(vector<string> & vecS, vector<int> & vecInt)
{
        for(int i=0; i<vecS.size(); i++)
        {
            int maxVal=0, maxPos=0;
            string maxString="00";
            maxVal=vecInt[i];
            maxPos=i;
            maxString=vecS[i];
            for(int l=(i+1); l<vecInt.size(); l++)
            {
                if(vecInt[l]>maxVal)
                {
                    maxVal=vecInt[l];
                    maxPos=l;
                    maxString=vecS[l];
                }
            }
            vecInt[maxPos]=vecInt[i];
            vecInt[i]=maxVal;
            vecS[maxPos]=vecS[i];
            vecS[i]=maxString;
        }
}
// a member function to return the sentences stored in the private data member
sentences
void DocumentAnalyzer::getSentences() const
{
        cout<<"The sentences within the edited text after editting are: \n"<<endl;
        for(int i=0; i<sentences.size(); i++) // a loop to iterate within the main vector
        {
            for(int k=0; k<sentences[i].size(); k++) // a loop to iterate within each
vector of strings stored at each position in the main vector
            {
                cout<<sentences[i][k]<< " ";
            }
            cout<<"\n"<<endl;
```

```
        }
}
vector<string> & DocumentAnalyzer::getInitialWordList()
{
        return words;
}
void DocumentAnalyzer::setApprovedValidTermsandWordCount()
{
        for(int i=0; i<validTerms.size(); i++)
        {
                if(wordCount[i]>2)
                {
                        approvedValidTerms.push_back(validTerms[i]);
                        approvedWordCount.push_back(wordCount[i]);
                }
        }
}
vector<string> & DocumentAnalyzer::getApprovedValidTerms()
{
        return approvedValidTerms;
}
vector<int> & DocumentAnalyzer::getApprovedWordCount()
{
        return approvedWordCount;
        }


// Project Main for Finding Potential Collocations within the Inputted Text
#include <fstream> // file stream
using std::ifstream; // input file stream
using std::ofstream; // output file stream
#include <iomanip>
#include <cstdlib>
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
//#include <math>
using namespace std;
// including the altime header file
#include "atltime.h"
#include "DocumentAnalyzer.h"
#include "CollectionAnalyzer.h"
```

```
// Global Function to calculate the mean
double mean(vector<int> v)
{
	double sum=0;
	for(int i=0; i<v.size(); i++)
		sum+=v[i];
	double mean=sum/v.size();
	return mean;
}
// Global Function to calculate standard deviation
double stdDev(vector<int> v, double mean)
{
	double sum=0;
	for(int i=0; i<v.size(); i++)
		sum+=pow((mean-v[i]),2)/v.size();
	double stdDev=sqrt(sum);
	return stdDev;
}

//Main Function
int main()
{
	//Declairing local variables
	string iS, name="00";
	int wordscount=0, validWindow=0, threshold=0, boarder=0;
	double average=0.0;
	const char *namePtr = 0;
	vector<int> wordCt;
	double avg=0.0;
	double sDev=0.0;
	CTime startTime, endTime;
	//the user is prompted to input the file name
	cout<<"Please enter the file name that contains your data to be analysed."
<<endl;
	cout<<"Make sure that the file is placed within the folder of this project\n in
the Visual Studio Directory."<<endl;
	cout<<"Make sure that the file name is spelled correctly, case-sensitive \n and
includes the extension <*.dat> or <*.txt>." <<endl;
	cin>>name;
	namePtr= name.data ( ); // casting the string into a constant character pointer
to be used
	ifstream inClientFile( namePtr, ios::in ); // declairing the input file
	// exit program if ifstream could not open file
	if ( !inClientFile )
	{
```

```
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        ofstream outClientFile( "Term Frequency.txt", ios::out ); // declairing the
output file
        // exit program if ifstream could not open file
        if ( !outClientFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        ofstream outClientFile1( "Initial Strings.txt", ios::out ); // declairing the output
file
        // exit program if ifstream could not open file
        if ( !outClientFile1 )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        ofstream outClientFile2( "tfid Frequency.txt", ios::out ); // declairing the output
file
        // exit program if ifstream could not open file
        if ( !outClientFile2 )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        ofstream outClientFile3( "Weighted Term Frequency.txt", ios::out ); //
declairing the output file
        // exit program if ifstream could not open file
        if ( !outClientFile3 )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        ofstream outClientFile4( "Augmented Weighted Term Frequency.txt", ios::out
); // declairing the output file
        // exit program if ifstream could not open file
        if ( !outClientFile4 )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        const char *filePtr = 0;
        string fileNameStr;
```

```
        int loopCounter=0;
        CollectionAnalyzer C1;// creating an instance of the class collectionanalyzer
        while(inClientFile>>fileNameStr)
        {
                loopCounter++;
                filePtr=fileNameStr.data();
                ifstream inDataBaseFile( filePtr, ios::in ); // declairing the input file
                // exit program if ifstream could not open file
                if ( !inDataBaseFile )
                {
                        cerr << "File could not be opened" << endl;
                        exit( 1 );
                } // end if
                string tempStr;
                while(inDataBaseFile>>tempStr)
                {
                        string ex, space=" ";
                        getline(inDataBaseFile, ex);
                        iS.append(tempStr);
                        iS.append(ex);
                        iS.append(space);
                }
                outClientFile1<<iS<<endl;

        outClientFile1<<"========================================
=================================="<<endl;

                DocumentAnalyzer D1(iS); // creatig an instance of a
DocumentAnalyzer class object
                D1.documentProcessing(); // performing document processing
operations on inputed text
                if(loopCounter==1)
                {
                        C1.setInitialCollection(D1.getApprovedValidTerms());
                        C1.setInitialFrequency(D1.getApprovedWordCount());
                }
                if(loopCounter>1)
                {
                        C1.searchLoop(D1.getApprovedValidTerms(),
D1.getApprovedWordCount(), loopCounter);
                }

                D1.~DocumentAnalyzer();
                iS.clear(); // clearing out the initial string to be ready to recieve a new
one
```

```
        }

        C1.approvedMatrix();
// printing out the matrix
        for(int n=0; n<C1.getInitialCollectionSize().size(); n++)
        {
                outClientFile<<left<<setw(25)<<C1.getInitialCollectionSize()[n]<<" ";
                for(int m=0; m<C1.getInitialFrequencySize().size(); m++)
                {

        outClientFile<<left<<setw(10)<<C1.getInitialFrequencySize()[m][n]<<" ";
                }
                outClientFile<<endl;
        }

        C1.TermFrequencyWeight();

        for(int o=0; o<C1.getInitialCollectionSize().size(); o++)
        {
                outClientFile3<<left<<setw(25)<<C1.getInitialCollectionSize()[o]<<" ";
                for(int p=0; p<C1.getpiFrequencySize().size(); p++)
                {

        outClientFile3<<left<<setw(10)<<C1.getpiFrequencySize()[p][o]<<" ";
                }
                outClientFile3<<endl;
        }

        C1.dfidfCalculation();

        for(int o=0; o<C1.getInitialCollectionSize().size(); o++)
        {
                outClientFile2<<left<<setw(25)<<C1.getInitialCollectionSize()[o]<<" ";
                for(int p=0; p<C1.getpFrequencySize().size(); p++)
                {

        outClientFile2<<left<<setw(10)<<C1.getpFrequencySize()[p][o]<<" ";
                }
                outClientFile2<<endl;
        }

        C1.AugmentedTermFrequencyWeight();
        for(int v=0; v<C1.getInitialCollectionSize().size(); v++)
        {
                outClientFile4<<left<<setw(25)<<C1.getInitialCollectionSize()[v]<<" ";
```

```
        for(int s=0; s<C1.getpFrequencySize().size(); s++)
        {

outClientFile4<<left<<setw(10)<<C1.getaFrequencySize()[s][v]<<" ";
        }
        outClientFile4<<endl;
}

return 0;
}
```

APPENDIX F (WEIGHTING ALGORITHM)

```
// CollectionAnalyzer class definition - DocumentAnalyzer class public interface
#include <iostream>
#include <vector>
#include <string>
using namespace std;
// preventing multiple inclusion of the header file
#ifndef COLLECTIONANALYZER_H
#define COLLECTIONANALYZER_H
// defining the DocumentAnalyzer class and prototypes
class CollectionAnalyzer
{
public:
        void setInitialCollection(vector<string>);
        void getInitialCollection()const;
        vector<string> & getInitialCollectionSize();
        void setInitialFrequency(vector<int>);
        void getInitialFrequency() const;
        vector<vector<int>> & getInitialFrequencySize();
        vector<vector<double>> & getpFrequencySize();
        vector<vector<double>> & getpiFrequencySize();
        vector<vector<double>> & getaFrequencySize();
        void print() const;
        void searchLoop(vector<string>, vector<int>, int);
        void approvedMatrix();
        void TermFrequencyWeight();
        void AugmentedTermFrequencyWeight();
        void dfidfCalculation();
        void processOriginalSpace();
        void implementNewSpace();
private:
        vector <string> iCollection; // a vector of a vector of strings representing the
collection words for each document
        vector <vector<int>> iFrequency; // a vector of a vector of integers storing the
frequency of occurrence of each collection word of each document
        vector <vector<double>> piFrequency; // a vector holding the weighted term
frequencies.
        vector <vector<double>> aFrequency; // a vector holding the augmented
frequencies of terms.
        vector <string> pCollection; // a vector of a vector of strings representing the
processed matrix of collection words for each document
```

```
        vector <vector<double>> pFrequency; // a vector of a vector of integers
storing the processed dfidf frequency of occurrence of each collection word of each
document
        vector <int> dfVector; // a vector including document frequency of terms.
        vector <int> NVector; // a vector including collection numbers.
        void addDummyVector();
        void matricAdjustment();
        vector <string> originalSpace; // a vector of a vector of strings representing
the original space genereated
        vector <vector<int>> originalSpaceFrequency; // a vector of a vector of
integers storing the frequency of occurrence of original space
};
#endif
  ---------------------------------------------------------------------------------------


// DocumentAnalyzer class definition - CollectionAnalyzer class public interface
#include <iostream>
#include <vector>
#include <string>
using namespace std;
#include "CollectionAnalyzer.h"
// preventing multiple inclusion of the header file
#ifndef PROJECT_H
#define PROJECT_H
// defining the DocumentAnalyzer class and prototypes
class DocumentAnalyzer
{
public:
        DocumentAnalyzer(string="00");
        ~DocumentAnalyzer();
        void setOriginalString(string);
        void setWordsVector();
        void getWordsVector()const;
        vector<string> & getWordsVectorSize();
        void setDocumentWordCount();
        int getDocumentWordCount ()const;
        void setDocumentSentencesCount();
        int getDocumentSentencesCount() const;
        void setWordSignificance();
        void getWordSignificance()const;
        void setStartEndCharacters();
        void setUnecessaryWords();
        void setEndOfSentence();
        void setPrefix();
        void setCapitalLetters();
```

```
        void documentProcessing();
        vector<string> & getInitialWordList();
        void setApprovedValidTermsandWordCount ();
        vector<string> & getApprovedValidTerms();
        vector<int> & getApprovedWordCount();
        void getSentences() const;
        void getValidTerms() const;
        void getWordCount() const;
        vector<string> & getValidTermsSize();
        vector<int> & getWordCountSize();
private:
        // defining the private data memebrs of each object of DocumentAnalyzer
class
        int DocumentWordCount; // integer that holds the number of words in a
provided text
        int documentSentencesCount; // integer that holds the number of sentences
in a provided text
        string originalString; // string that intakes the passed string to be processed
        vector <string> words; // a vector that holds all words in the passed text
        vector <vector<string>> sentences; // a vector that holds all sentences of the
passed text
        vector <string> validTerms; // a vector that holds word objects. It includes
words to be further processed
        vector <int> wordCount; // a vector that hold the number of occurancec of
each word int he valid terms
        vector<string> approvedValidTerms; // accepts valid terms that were repeated
more than a certain number of times
        vector<int> approvedWordCount; // accepts valid terms counts that were
repeated more than a certain number of times
        vector<string> startEndCharacters;// a vector that holds characters to be
removed from the start and end of word
        vector<string> unecessaryWords;// a vector that includes words to be
removed from the text before processing
        vector<string> endOfSentence;// a vector that includes strings considered to
be end of sentence characters
        vector<string> prefix;// a vector including most known prefixes
        vector<string> capitalLetters;// a vector inlcuding a set of all 26 in the upper
case form.
        vector<double> wordSignificance; // a vector of doubles representing the
signifcance of each repeated term
        int startingcharacter (string &); // a utility function that removes starting
characters
        int endingCharacter (string &); // a utility function that removes ending
characters
```

```
        int possisveCheck (string &); // a utility function that removes possisive
characters
        int pluralCheck (string &, int); // a utility function that changes a plural forms of
a word
        int checkPlural (string &);
        int endOfSentenceCheck (string &, int); // a utility function that defines he end
of sentence within a text
        int unecessaryWordsCheck (string &); // a utility function that removes
unwanted wards from the text
        void upperToLower (string &); // a utility function that converts all upper case
letters to lower ones
        void sorting (vector<string> &, vector<int> &); // a utility function that performs
a sorting algorithm
};
        #endif

        ---------------------------------------------------------------------------------------

// the code utilizes the input/output standard stream, vector, and standard string
classes
#include <fstream> // file stream
using std::ifstream; // input file stream
using std::ofstream; // output file stream
#include <iomanip>
#include <cstdlib>
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
//#include <math>
using namespace std;
#include <iostream>
#include <vector>
#include <string>
using namespace std;
//Including header files of DocumentAnalyzer and Word classes
#include "DocumentAnalyzer.h"
#include "CollectionAnalyzer.h"
// declairing member functions
void CollectionAnalyzer::setInitialCollection(vector<string> iC)
{
        for(int i=0; i<iC.size(); i++)
        {
                iCollection.push_back(iC[i]);
        }
```

```
}
void CollectionAnalyzer::getInitialCollection()const
{
	for(int i=0; i<iCollection.size(); i++)
	{
		cout<<iCollection[i]<<endl;
	}
}
vector<string> & CollectionAnalyzer::getInitialCollectionSize()
{
	return iCollection;
}
void CollectionAnalyzer::setInitialFrequency(vector<int> iF)
{
	iFrequency.push_back(iF);
}
void CollectionAnalyzer::getInitialFrequency()const
{
	for(int i=0; i<iFrequency.size(); i++)
	{
		for(int j=0; j<iFrequency[i].size(); j++)
		{
			cout<<iFrequency[i][j]<<" ";
		}
		cout<<endl;
	}
}
vector<vector<int>> & CollectionAnalyzer::getInitialFrequencySize()
{
	return iFrequency;
}
vector<vector<double>> & CollectionAnalyzer::getpFrequencySize()
{
	return pFrequency;
}
vector<vector<double>> & CollectionAnalyzer::getpiFrequencySize()
{
	return piFrequency;
}
vector<vector<double>> & CollectionAnalyzer::getaFrequencySize()
{
	return aFrequency;
}
void CollectionAnalyzer::print() const
{
```

```
        // printing out the matrix
        for(int n=0; n<iCollection.size(); n++)
        {
                cout<<left<<setw(25)<<iCollection[n]<<" ";
                for(int m=0; m<iFrequency.size(); m++)
                {
                        cout<<left<<setw(10)<<iFrequency[m][n]<<" ";
                }
                cout<<endl;
        }
}
void CollectionAnalyzer::addDummyVector()
{
        vector<int> tempVector;
        for(int i=0; i<iCollection.size(); i++)
        {
                tempVector.push_back(0);
        }
        iFrequency.push_back(tempVector);
        tempVector.clear();
}
void CollectionAnalyzer::matricAdjustment()
{
        for(int i=0; i<iFrequency.size(); i++)
        {
                int missingData=iCollection.size()- iFrequency[i].size();
                for(int j=0; j<missingData; j++)
                {
                        iFrequency[i].push_back(0);
                }
                missingData=0;
        }
}
void CollectionAnalyzer::searchLoop(vector<string> iC, vector<int> iF, int counter)
{
        int tempIndex=0;
        bool tempBool=false;
        addDummyVector();
        for(int i=0; i<iC.size(); i++)
        {
                tempIndex=0;
                for(int j=0; j<iCollection.size(); j++)
                {
                        if(iC[i]==iCollection[j])
                        {
```

```
                                tempIndex=j;
                                tempBool=true;
                        }
                }
                if(tempBool==true)
                {
                        iFrequency[counter-1][tempIndex]=iF[i];
                }
                if(tempBool==false)
                {
                        iCollection.push_back(iC[i]);
                        iFrequency[counter-1].push_back(iF[i]);
                }
                tempBool=false;
        }
        matricAdjustment();
}
void CollectionAnalyzer::approvedMatrix()
{
        vector<int> sumOverDocuments;
        vector<string> tempICollection;
        int sum=0;
        for(int i=0; i<iCollection.size(); i++)
        {
                for(int j=0; j<iFrequency.size(); j++)
                {
                        sum=sum+iFrequency[j][i];
                }

                sumOverDocuments.push_back(sum);
                sum=0;
        }
        for(int v=0; v<iCollection.size(); v++)
        {
                tempICollection.push_back(iCollection[v]);
        }
        int turn=0;
        bool first=false;
        for(int k=0; k<sumOverDocuments.size(); k++)
        {
                if(sumOverDocuments[k]<3)
                {
                        if(k==0)
                        {
                                first=true;
```

```
                        iCollection.erase(iCollection.begin());
                        for(int t=0; t<iFrequency.size(); t++)
                                {
                                        iFrequency[t].erase(iFrequency[t].begin());
                                }
                }
                if(first==true)
                {
                        if(iCollection[0]==tempICollection[k])
                        {
                                iCollection.erase(iCollection.begin());
                                for(int l=0; l<iFrequency.size(); l++)
                                {
                                        iFrequency[l].erase(iFrequency[l].begin());
                                }
                        }
                        else
                        {
                                iCollection.erase(iCollection.begin()+k-1);
                                for(int l=0; l<iFrequency.size(); l++)
                                {
                                        iFrequency[l].erase(iFrequency[l].begin()+k-
1);
                                }
                        }
                }
                if(first==false)
                {
                        iCollection.erase(iCollection.begin()+(k-turn));
                        for(int l=0; l<iFrequency.size(); l++)
                                {

        iFrequency[l].erase(iFrequency[l].begin()+(k-turn));
                                }
                        turn++;
                }
                }
        }
}
// dfidf calculations function
void CollectionAnalyzer::dfidfCalculation()
{
        vector <double> tempdVector;
        vector <double> tempNVector;
        int dfCounter=0;
```

```
for(int i=0; i<iCollection.size(); i++)
{
	for(int j=0; j<iFrequency.size(); j++)
	{
		if(iFrequency[j][i]>0)
		{
			dfCounter++;
		}
	}
	dfVector.push_back(dfCounter);
	NVector.push_back(iFrequency.size());
	dfCounter=0;
}
for(int t=0; t<iCollection.size(); t++)
{
	double dN=0.0;
	double dF=0.0;
	double tempf=0.0;
	dN=static_cast< double >(NVector[t]);
	dF=static_cast< double >(dfVector[t]);
	tempf=log10(dN)-log10(dF);
	tempdVector.push_back(tempf);
}
vector <double> tempPFrequency;
for(int r=0; r<iCollection.size(); r++)
{
	tempPFrequency.push_back(0.0);
}
for(int u=0; u<piFrequency.size(); u++)
{
	pFrequency.push_back(tempPFrequency);
}
tempPFrequency.clear();
double pf=0.0;
for(int x=0; x<iCollection.size(); x++)
{
	for(int z=0; z<piFrequency.size(); z++)
	{
		if(iFrequency[z][x]>0)
		{
			pf=piFrequency[z][x]*tempdVector[x];
			pFrequency[z][x]=pf;
		}
	}
```

```
		}
}
void CollectionAnalyzer::implementNewSpace()
{
		int fcounter=0;
		vector<double> tempdf, tempN;
		for(int i=0; i<iCollection.size(); i++)
		{
			tempN.push_back(iFrequency.size()+originalSpaceFrequency.size());
			tempdf.push_back(0.0);
			for(int j=0; j<originalSpace.size(); j++)
			{
				if (iCollection[i]==originalSpace[j])
				{
					for (int n=0; n<originalSpaceFrequency.size();
n++)
					{
						if(originalSpaceFrequency[n][j]>0)
						{
							fcounter++;
						}
					}
					for(int k=0; k<iFrequency.size(); k++)
					{
						if(iFrequency[k][i]>0)
						{
							fcounter++;
						}
					}
					tempdf[i]=fcounter;
					fcounter=0;
			}
		}
		vector<double> tempdVector;
		for(int s=0; s<iCollection.size(); s++)
		{
			double tempf=0.0;
			if(tempdf[s]==0)
			{
				tempdVector.push_back(0.0);
			}
			else
			{
				tempf=log10(tempN[s])-log10(tempdf[s]);
				tempdVector.push_back(tempf);
```

```
                }
        }
        vector <double> tempPFrequency;
        for(int r=0; r<iCollection.size(); r++)
        {
                tempPFrequency.push_back(0.0);
        }
        for(int u=0; u<piFrequency.size(); u++)
        {
                pFrequency.push_back(tempPFrequency);
        }
        tempPFrequency.clear();

        double pf=0.0;
        for(int x=0; x<iCollection.size(); x++)
        {
                for(int z=0; z<piFrequency.size(); z++)
                {
                        if(iFrequency[z][x]>0)
                        {
                                pf=piFrequency[z][x]*tempdVector[x];
                                pFrequency[z][x]=pf;
                        }
                }
        }
}
static double Log10(double d);
void CollectionAnalyzer::TermFrequencyWeight()
{
        double tempf=0.0;
        double l=0.0;
        vector <double> tempPiFrequency;
        for(int r=0; r<iCollection.size(); r++)
        {
                tempPiFrequency.push_back(0.0);
        }
        for(int u=0; u<iFrequency.size(); u++)
        {
                piFrequency.push_back(tempPiFrequency);
        }
        tempPiFrequency.clear();

        for (int i=0; i<iCollection.size(); i++)
        {
                for(int j=0; j<iFrequency.size(); j++)
```

```
                {
                        if(iFrequency[j][i]>0)
                        {
                                tempf = static_cast< double >(iFrequency[j][i]);
                                l=log10(tempf);
                                piFrequency[j][i]=1+l;
                        }
                }
        }
}
void CollectionAnalyzer::AugmentedTermFrequencyWeight()
{
        double tempaf=0.0;
        double l=0.0;
        vector <double> tempaiFrequency;
        for(int r=0; r<iCollection.size(); r++)
        {
                tempaiFrequency.push_back(0.0);
        }
        for(int u=0; u<iFrequency.size(); u++)
        {
                aFrequency.push_back(tempaiFrequency);
        }
        tempaiFrequency.clear();
        int maxFrequency=0;
        vector <int> tempMaxFrequency;
        for (int i=0; i<iCollection.size(); i++)
        {
                for(int j=0; j<iFrequency.size(); j++)
                {
                        if(iFrequency[j][i]>maxFrequency)
                        {
                                maxFrequency=iFrequency[j][i];
                        }
                        tempMaxFrequency.push_back(maxFrequency);
                }
        }
        double tempA=0.0;
        double tempAugmentedFrequency=0.0;
        for (int c=0; c<iCollection.size(); c++)
        {
                for (int h=0; h<iFrequency.size(); h++)
                {
                        tempA=0.5+((0.5*iFrequency[h][c])/tempMaxFrequency[c]);
                        aFrequency[h][c]=tempA;
```

```
                }
        }
}
void CollectionAnalyzer::processOriginalSpace()
{
        ifstream inUnWordFile( "Term Frequency.txt", ios::in ); // declairing the output
file
        // exit program if ifstream could not open file
        if ( !inUnWordFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        string oWord;
        int
f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13,f14,f15,f16,f17,f18,f19,f20,f21,f22,f23,f24,f25,f
26,f27,f28,f29,f30,f31,f32,f33,f34,f35,f36,f37,f38,f39,f40;
        vector <int>
v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,v18,v19,v20,v21,v22,v
23,v24,v25,v26,v27,v28,v29,v30,v31,v32,v33,v34,v35,v36,v37,v38,v39,v40;
        vector <string> tempSpace;
        while(inUnWordFile>>oWord>>f1>>f2>>f3>>f4>>f5>>f6>>f7>>f8>>f9>>f10>
>f11>>f12>>f13>>f14>>f15>>f16>>f17>>f18>>f19>>f20>>f21>>f22>>f23>>f24>>f2
5>>f26>>f27>>f28>>f29>>f30>>f31>>f32>>f33>>f34>>f35>>f36>>f37>>f38>>f39>
>f40)
        {
                tempSpace.push_back(oWord);
                v1.push_back(f1);
                v2.push_back(f2);
                v3.push_back(f3);
                v4.push_back(f4);
                v5.push_back(f5);
                v6.push_back(f6);
                v7.push_back(f7);
                v8.push_back(f8);
                v9.push_back(f9);
                v10.push_back(f10);
                v11.push_back(f11);
                v12.push_back(f12);
                v13.push_back(f13);
                v14.push_back(f14);
                v15.push_back(f15);
                v16.push_back(f16);
                v17.push_back(f17);
                v18.push_back(f18);
```

```
		v19.push_back(f19);
		v20.push_back(f20);
		v21.push_back(f21);
		v22.push_back(f22);
		v23.push_back(f23);
		v24.push_back(f24);
		v25.push_back(f25);
		v26.push_back(f26);
		v27.push_back(f27);
		v28.push_back(f28);
		v29.push_back(f29);
		v30.push_back(f30);
		v31.push_back(f31);
		v32.push_back(f32);
		v33.push_back(f33);
		v34.push_back(f34);
		v35.push_back(f35);
		v36.push_back(f36);
		v37.push_back(f37);
		v38.push_back(f38);
		v39.push_back(f39);
		v40.push_back(f40);
}
for(int i=0; i<tempSpace.size(); i++)
{
		originalSpace.push_back(tempSpace[i]);
}
originalSpaceFrequency.push_back(v1);
originalSpaceFrequency.push_back(v2);
originalSpaceFrequency.push_back(v3);
originalSpaceFrequency.push_back(v4);
originalSpaceFrequency.push_back(v5);
originalSpaceFrequency.push_back(v6);
originalSpaceFrequency.push_back(v7);
originalSpaceFrequency.push_back(v8);
originalSpaceFrequency.push_back(v9);
originalSpaceFrequency.push_back(v10);
originalSpaceFrequency.push_back(v11);
originalSpaceFrequency.push_back(v12);
originalSpaceFrequency.push_back(v13);
originalSpaceFrequency.push_back(v14);
originalSpaceFrequency.push_back(v15);
originalSpaceFrequency.push_back(v16);
originalSpaceFrequency.push_back(v17);
originalSpaceFrequency.push_back(v18);
```

```
originalSpaceFrequency.push_back(v19);
originalSpaceFrequency.push_back(v20);
originalSpaceFrequency.push_back(v21);
originalSpaceFrequency.push_back(v22);
originalSpaceFrequency.push_back(v23);
originalSpaceFrequency.push_back(v24);
originalSpaceFrequency.push_back(v25);
originalSpaceFrequency.push_back(v26);
originalSpaceFrequency.push_back(v27);
originalSpaceFrequency.push_back(v28);
originalSpaceFrequency.push_back(v29);
originalSpaceFrequency.push_back(v30);
originalSpaceFrequency.push_back(v31);
originalSpaceFrequency.push_back(v32);
originalSpaceFrequency.push_back(v33);
originalSpaceFrequency.push_back(v34);
originalSpaceFrequency.push_back(v35);
originalSpaceFrequency.push_back(v36);
originalSpaceFrequency.push_back(v37);
originalSpaceFrequency.push_back(v38);
originalSpaceFrequency.push_back(v39);
originalSpaceFrequency.push_back(v40);
}

---------------------------------------------------------------------------------------

// DocumentAnalyzer member-function definitions - DocumentAnalyzer class
member-function implementation
// the code utilizes the input/output standard stream, vector, and standard string
classes
#include <fstream> // file stream
using std::ifstream; // input file stream
using std::ofstream; // output file stream
#include <iomanip>
#include <cstdlib>
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
using namespace std;
#include <iostream>
#include <vector>
#include <string>
using namespace std;
//Including header files of DocumentAnalyzer and Word classes
```

```
#include "DocumentAnalyzer.h"
//Constructor that takes in as an argument the initial string to set its initial private
data members
DocumentAnalyzer::DocumentAnalyzer(string s)
{
	setOriginalString(s);
	setWordsVector();
	setDocumentWordCount();
	setDocumentSentencesCount();
	setStartEndCharacters();
	setUnecessaryWords();
	setEndOfSentence();
	setPrefix();
	setCapitalLetters();
}
DocumentAnalyzer::~DocumentAnalyzer()
{
}
// a set function for the initial string
void DocumentAnalyzer::setOriginalString(string s)
{
	originalString = s;
}
void DocumentAnalyzer::setWordsVector()
{
	int indexOfSpace;
	string word;

	for(int i=0; i<originalString.length(); i++)
	{
		indexOfSpace=originalString.find(" ");
		word=originalString.substr(0,indexOfSpace);
		if(indexOfSpace>0)
		{
			words.push_back(word);

	originalString=originalString.substr(indexOfSpace+1,originalString.length()-1);
			i=0;
		}
		else
		{

	originalString=originalString.substr(indexOfSpace+1,originalString.length()-1);
			i=0;
		}
```

```
        }
}
//a get function that prints out the words of a document
void DocumentAnalyzer::getWordsVector() const
{
        for(int k=0; k<words.size(); k++)
                cout<<words[k]<<endl;
}
vector<string> & DocumentAnalyzer::getWordsVectorSize()
{
        return words;
}
// a set function to set the private data member DocumentWordCount
void DocumentAnalyzer::setDocumentWordCount()
{
        DocumentWordCount = words.size();
}
// a get function that returns the number of words in a text
int DocumentAnalyzer::getDocumentWordCount() const
{
        return DocumentWordCount;
}
// a get function to return an aliace of the valid terms vector
vector<string> & DocumentAnalyzer::getValidTermsSize()
{
        return validTerms;
}
// a get function to return an aliace of the wordCount vector
vector<int> & DocumentAnalyzer::getWordCountSize()
{
        return wordCount;
}
// a set function to set the private data memebr startEndCharacters vector
void DocumentAnalyzer::setStartEndCharacters()
{
        ifstream inCharFile( "startendchar.txt", ios::in ); // declairing the output file
        // exit program if ifstream could not open file
        if ( !inCharFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if

        string end;
```

```
        while(inCharFile>>end)
        {
                startEndCharacters.push_back(end);
        }
}
// a set function to set the private data memebr unecessaryWords vector
void DocumentAnalyzer::setUnecessaryWords()
{
        ifstream inUnWordFile( "unecessaryWords.txt", ios::in ); // declairing the
output file
        // exit program if ifstream could not open file
        if ( !inUnWordFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if

        string unword;

        while(inUnWordFile>>unword)
        {
                unecessaryWords.push_back(unword);
        }
}
// a set function to set the private data memebr endOfSentence vector
void DocumentAnalyzer::setEndOfSentence()
{
        ifstream inEndSentFile( "endsentence.txt", ios::in ); // declairing the output file
        // exit program if ifstream could not open file
        if ( !inEndSentFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if

        string endsent;

        while(inEndSentFile>>endsent)
        {
                endOfSentence.push_back(endsent);
        }
}
// a set function to set the private data memebr prefix vector
void DocumentAnalyzer::setPrefix()
{
```

```
        ifstream inPrefixFile( "prefix.txt", ios::in ); // declairing the output file
        // exit program if ifstream could not open file
        if ( !inPrefixFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if

        string pref;

        while(inPrefixFile>>pref)
        {
                prefix.push_back(pref);
        }
}
// a set function to set the private data memebr capitalLetters vector
void DocumentAnalyzer::setCapitalLetters()
{
        ifstream inCapFile( "capittalletters.txt", ios::in ); // declairing the output file
        // exit program if ifstream could not open file
        if ( !inCapFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if

        string cap;

        while(inCapFile>>cap)
        {
                capitalLetters.push_back(cap);
        }
}
// a set function to set the private data memebr documentSentencesCount, which
represents the number of sentences within a text
void DocumentAnalyzer::setDocumentSentencesCount()
{
        documentSentencesCount=0;
}
// a get functionthat returns the private data memebr documentSentencesCount,
which represents the number of sentences within a text
int DocumentAnalyzer::getDocumentSentencesCount() const
{
        return documentSentencesCount;
}
```

```
// setting the Word Significance vector
void DocumentAnalyzer::setWordSignificance()
{
	double sum=0.0;
	for(int i=0; i<validTerms.size(); i++)
	{
		sum=sum+wordCount[i];
	}
	for(int s=0; s<validTerms.size(); s++)
	{
		double temp=0.0;
		temp=(((static_cast<double>(wordCount[s])/sum))*100);
		wordSignificance.push_back(temp);
	}
}
void DocumentAnalyzer::getWordSignificance() const
{
	for(int i=0; i<wordSignificance.size(); i++)
	{
		cout<<wordSignificance[i];
	}
}
// a utility function that removes starting characters. It takes string by reference and
returns an integer
int DocumentAnalyzer::startingcharacter(string & str)
{
	int counter=0;
	string sub="00";
	sub=str.substr(0,1);
	for(int i=0; i< startEndCharacters.size(); i++) // a loop to check if the first lette
in the word is an unwatnted starting character
	{
		if (sub==startEndCharacters[i])
		{
			counter=1; // if the first letter in the word is an unwatnted starting
character a counter is set to 1
			str=str.substr(1,str.length()-1);
		}
	}
	if (counter == 1)
		return 1; // the function returns 1 if the first lette in the word is an
unwatnted starting character
	else
		return 0; // the function returns 0 if the first lette in the word is not an
unwatnted starting character
```

```
}
// a utility function that removes ending characters. It takes string by reference and
returns an integer
int DocumentAnalyzer::endingCharacter (string & str)
{
	int counter=0;
	string sub="00";
	sub=str.substr(str.length()-1,1);
	for(int i=0; i< startEndCharacters.size(); i++) // a loop to check if the last lette
in the word is an unwatnted ending character
	{
		if (sub==startEndCharacters[i])
		{
			counter=1; // if the last lette in the word is an unwatnted ending
character a counter is set to 1
			str=str.substr(0,str.length()-1);
		}
	}
	if (counter == 1)
		return 1; // the function returns 1 if the last lette in the word is an
unwatnted ending character
	else
		return 0; // the function returns 0 if the last lette in the word is not an
unwatnted ending character
}
// a utility function that removes possisive characters. It takes string by reference and
returns an integer
int DocumentAnalyzer::possisveCheck(string & str)
{
	int counter=0;
	if (str.length()>1)
	{
		string last="00";
		string beforelast="00";
		last=str.substr(str.length()-1,1);
		beforelast=str.substr(str.length()-2,1);
		if (last=="s") // nested if conditions to test if the last two letters of a
word are 's
		{
			if (beforelast=="'"||beforelast=="’"||beforelast=="‘")
			{
				counter=1;
				str=str.substr(0,str.length()-2);
			}
		}
```

```
	}
	if (counter == 1)
		return 1; // if the last two letters of a word are 's, the function returns 1
	else
		return 0; // if the last two letters of a word are not 's, the function
returns 0
}
// a utility function that changes a plural forms of a word. It takes a sting by reference
and returns an integer
int DocumentAnalyzer::pluralCheck(string & str, int pos)
{
	static vector<string> temp;
	string tempString="00", tempStringIes="00";
	int counter=0;
	if(str.length()>1)
	{
		string last="00", last2="00";
		last=str.substr(str.length()-1,1);
		last2=str.substr(str.length()-2,2);

		if (last=="s"&&last2!="ss")
		{
			bool partOfUnecessaryWors=true;

			for(int i=0; i<unecessaryWords.size(); i++)
			{
				if(str==unecessaryWords[i])
				{
					partOfUnecessaryWors=false;
					break;
				}
			}
			if(partOfUnecessaryWors==true) // if the last letter of a word is
s, the user is prompter to define if the word is in the plural form or not. If yes, he is
prompted to enter the singular form.
			{
				bool pluralCheckBool=true;
				if(temp.size()>=1)
				{
					for(int v=0; v<temp.size(); v++)
					{
						if(str==temp[v])
						{
							pluralCheckBool=false;
							str=temp[v+1];
```

```
                                        break;
                                }
                        }
                }
                if(pluralCheckBool==true)
                {
                        char choice='0', confirm='0';

                        cout<< "Is the following word in the plural form?
<"<< str <<">"<<endl;

                        if(words.size()>=3)
                        {
                                if(pos==0)
                                        cout<< "The Word was mentioned in
the following context <"<<words[pos]<<" "<<words[pos+1]<<"
"<<words[pos+2]<<">."<<endl;
                                if(pos>0&&pos<words.size()-1)
                                        cout<< "The Word was mentioned in
the following context <"<<words[pos-1]<<" "<<words[pos]<<"
"<<words[pos+1]<<">."<<endl;
                                if(pos==words.size()-1)
                                        cout<< "The Word was mentioned in
the following context <"<<words[pos-2]<<" "<<words[pos-1]<<"
"<<words[pos]<<">."<<endl;
                        }

                        cout<< "Please enter the appropriate number
corresponding to your choice\n"
                                << "<y> for Yes\n" << "<n> for No\n" <<
endl;
                        cin>>choice;

        while(choice!='y'&&choice!='n'&&choice!='Y'&&choice!='N')
                        {
                                cout<<"You have entered an invalid
choice.\n"<<endl;
                                cout<<"Please limit your choice between
<y> or <n>"<<endl;
                                cin>>choice;
                        }
                        if(choice=='y'||choice=='Y')
                        {
                                temp.push_back(str);
                                tempString=str.substr(0,str.length()-1);
```

```
                                        tempStringIes=str.substr(str.length()-3,3);
                                        if(tempStringIes=="ies")
                                        {
                                                tempString=str.substr(0,str.length()-
3);
                                                tempString.append("y");
                                        }
                                        cout<<"Is this the singular form of the
word? <"<<tempString<<">\n"
                                                <<"Please enter the appropriate
number corresponding to your choice\n"
                                                << "<y> for Yes\n" << "<n> for No\n"
<< endl;
                                        cin>>confirm;

        while(confirm!='y'&&confirm!='n'&&confirm!='Y'&&confirm!='N')
                                        {
                                                cout<<"You have entered an invalid
choice.\n"<<endl;
                                                cout<<"Please limit your choice
between <y> and <n>"<<endl;
                                                cin>>confirm;
                                        }
                                        if(confirm=='y'||confirm=='Y')
                                        {
                                                temp.push_back(tempString);
                                                str=tempString;
                                        }
                                        else
                                        {
                                                string newWord="00";
                                                char check='0';
                                                cout<< "Please enter the singular
form from the previouse word with no spaces in between\n" << endl;
                                                cin>>newWord;
                                                cout<<"Is the word you have entered
is <"<<newWord<<">"<<endl;
                                                cout<<"Please enter your choice
below:\n" <<"<y> for Yes\n" << "<n> for No\n" <<endl;
                                                cin>>check;

        while(check!='y'&&check!='n'&&check!='Y'&&check!='N')
                                                {
                                                        cout<<"You have entered an
invalid choice.\n"<<endl;
```

```
                                                cout<<"Please limit your
choice between <y> and <n>"<<endl;
                                                cin>>check;
                                        }
                                        while(check=='n'||check=='N')
                                        {
                                                cout<< "Please enter the
singular form from the previouse word with no spaces in between\n" << endl;
                                                cin>>newWord;
                                                cout<<"Is the word you have
entered is <"<<newWord<<">"<<endl;
                                                cout<<"Please enter your
choice below:\n" <<"<y> for Yes\n" << "<n> for No\n"<<endl;
                                                cin>>check;

        while(check!='y'&&check!='n'&&check!='Y'&&check!='N')
                                                {
                                                        cout<<"You have
entered an invalid choice.\n"<<endl;
                                                        cout<<"Please limit
your choice between <y> and <n>"<<endl;
                                                        cin>>check;
                                                }
                                        }
                                        temp.push_back(newWord);
                                        str=newWord;
                                        counter=1;
                                }
                        }
                        if(choice=='n'||choice=='N')
                        {
                                tempString=str;
                                temp.push_back(str);
                                temp.push_back(tempString);
                        }
                }
        }
}
if (counter == 1)
{
        return 1; // if the form of the word was changed, the memeber function
returns a 1
}
else
```

```
		return 0; // if the form of the word was not changed, the memeber function returns a 0
}
// a utility function that changes a word from its plural form into its singular form without user's feedback. Returns 1
// the tested word has changed, returns 0 if it is unchanged.
int DocumentAnalyzer::checkPlural(string & word)
{
	bool isUnecessary=false;
	bool hasChanged=false;
	for(int i=0; i<unecessaryWords.size();i++)
	{
		if(unecessaryWords[i]==word)
		{
			isUnecessary=true;
			break;
		}
	}
	if(!isUnecessary)
	{
		if(word.substr(word.length()-1,1)=="s")
		{
			if(word.substr(word.length()-2,2)!="as"&&word.substr(word.length()-2,2)!="is"&&word.substr(word.length()-2,2)!="os"
				&&word.substr(word.length()-2,2)!="us"&&word.substr(word.length()-2,2)!="ss")
			{
				if(word.substr(word.length()-3,3)=="ies")
					word=word.substr(0,word.length()-3)+"y";
				else if(word.substr(word.length()-2,2)=="es")
				{
					if(word.substr(word.length()-4,4)=="sses"||word.substr(word.length()-3,3)=="xes")
						word=word.substr(0,word.length()-2);
					else
						word=word.substr(0,word.length()-1);
				}
				else
					word=word.substr(0,word.length()-1);

				hasChanged=true;
			}
		}
	}
```

```
        if(hasChanged)
                return 1;
        else
                return 0;
}
// a utility function the tests is a word is at the ned of the sentence or not. It takes a
tring by reference as argument and returns an integer
int DocumentAnalyzer::endOfSentenceCheck(string & str, int pos)
{
        int counter=0;
        if(str.length()>1)
        {
                string last="00";
                bool endOfSentenceBool=true;
                last=str.substr(str.length()-1, 1);
                for (int i=0; i<endOfSentence.size(); i++) // aloop to test if the last letter
of the word is considered as an end of sentence character
                {
                        if(last==endOfSentence[i])
                        {
                                endOfSentenceBool=false;
                        }
                }
                if(endOfSentenceBool==false)
                {
                        string newWord="00";
                        counter=1;
                        newWord=str.substr(0, str.length()-1);
                        str=newWord; // modifing the passed argument by removing the
end of sentence chracter

                        if(last==".") // testing if the end of sentence was a period or not
                        {
                                for(int k=0; k<prefix.size(); k++) // making sure thatthe
period was not used for a prefix
                                {
                                        if(str==prefix[k])
                                                counter=0;
                                }
                                if(counter==1 && words.size()>(pos+1))
                                {
                                        for(int h=0; h<words[pos+1].length(); h++)
                                        {
                                                string q="00";
                                                q=words[pos+1].substr(0,1);
```

```
                                        for(int g=0; g<startEndCharacters.size();
g++)
                                        {
                                                if(q==startEndCharacters[g])
                                                {
        words[pos+1]=words[pos+1].substr(1, words[pos+1].length()-1);
                                                        break;
                                                }
                                        }
                                }
                        }
                        if(counter==1 && words.size()>(pos+1)) // making sure
thatthe period was not used for abbreviation
                        {
                                string first="00";
                                first=words[pos+1].substr(0, 1);
                                for(int z=0; z<capitalLetters.size(); z++)
                                {
                                        if(first==capitalLetters[z])
                                        {
                                                counter=1;
                                                break;
                                        }
                                        else
                                                counter=0;
                                }
                        }
                }
        }
        if(counter==1)
                return 1; //if the  last letter was an end of sentence, the function returns
1
        else
                return 0; //if the  last letter was not an end of sentence, the function
returns 0
}
// a utility function to check if the word is an unwanted word or not. The function
takes a string as an argument and returns an integer
int DocumentAnalyzer::unecessaryWordsCheck(string & str)
{
        int counter=0;
```

```
	for(int i=0; i<unecessaryWords.size(); i++) // a loop to check if the word is considere as an unecessary word or not
	{
		if(str==unecessaryWords[i])
			counter=1;
	}

	if(counter==1)
		return 1; // if the word was found to be unecessary, the function returns 1
	else
		return 0; // if the word was not found to be unecessary, the function returns 0
}
// a member function that utilizes the diffeent utility functions of the DocuentAnalyzer class to process all words and fill the following private data member
// vector <vector<string>> sentences that holds all sentences of the passed text
// vector <Word> validTerms that holds word objects. It includes words to be further processed
void DocumentAnalyzer::documentProcessing()
{
	vector<string> temp;
	vector <string> tempValidTerms;
	int termCounter=0;

	for(int i=0; i<words.size(); i++) // a loop that iterates through the vector of all words
	{

		bool fullProcess=true;
		bool endOfSentenceBool=true;
		int wordValidation=0, size=0;

		upperToLower(words[i]); // converting all upper case letters to lower ones. This is to make sure that if a word is included more than once with lower and upper case letters, they will be treated the same.

		// code that defines string senseNum and boolean isTagged=false, and checks if word[i] contains '\'
		// if word[i] contains '\', split word[i] at '\' into word[i] and senseNum and make isTagged=true

		// a loop used to make sure that the edited word has undergone all required processing aspects and is ready to be included in the rest of the private data members.
```

```
                // the choosen order of pocessing is set in the follwoing manner to save
processing time.
                while(fullProcess==true)
                {
                        bool startCharacter=true, endCharacter=true, possisive=true,
plural=true, sentenceTest=true;
                        int sC=0, eC=0, pS=0, pL=0, eS=0;


                        sC=startingcharacter(words[i]); // processing the word for
starting characters
                        if(sC==1)
                                startCharacter=false;
                        eC=endingCharacter(words[i]); // processing the word for ending
characters
                        if(eC==1)
                                endCharacter=false;
                        pS=possisveCheck(words[i]); // processing the word for possive
check
                        if(pS==1)
                                possisive=false;
                        pL=checkPlural(words[i]); // processing the word for plural check
                        if(pL==1)
                                plural=false;
                        eS=endOfSentenceCheck(words[i], i); // testing if the word is at
the end of a sentence
                        if(eS==1)
                        {
                                sentenceTest=false;
                                endOfSentenceBool=false;
                        }

        if(startCharacter==true&&endCharacter==true&&possisive==true&&plural==tr
ue&&sentenceTest==true)
                        {
                                fullProcess=false;
                        }
                }
                // code that appends to word[i] its senseNum before inserting word[i]
into the sentences vector of vector and
                // the validTerms vector
                temp.push_back(words[i]); // pushing back the word into a local vector
                if(endOfSentenceBool==false) // testing if the word was at the end of
sentence or not.
                {
```

```
                vector <int> sentnecCheckVector;
                int sum=0;

                documentSentencesCount++;

                for(int q=0; q<sentences.size(); q++)
                {
                        int sentenceEquality=0;

                        if(temp.size()==sentences[q].size())
                        {
                                for(int w=0; w<sentences[q].size(); w++)
                                {
                                        if(temp[w]!=sentences[q][w])
                                        {
                                                sentenceEquality=1;
                                                break;
                                        }
                                }

        sentnecCheckVector.push_back(sentenceEquality);
                        }
                }
                if(sentnecCheckVector.size()>=1)
                {
                        for(int e=0; e<sentnecCheckVector.size(); e++)
                                sum=sum+sentnecCheckVector[e];

                        if(sum==sentnecCheckVector.size())
                        {
                                sentences.push_back(temp); // if the word was at
the end of a sentence, the local vector is pushed back into the private data member
                                temp.clear();
                        }
                        else
                                temp.clear();
                }
                else
                {
                        sentences.push_back(temp); // if the word was at the end
of a sentence, the local vector is pushed back into the private data member
                        temp.clear();
                }
        }
```

```
                wordValidation=unecessaryWordsCheck(words[i]); // performing the
unecessary word check
                if(wordValidation==0) // checking if the word is a valid one or not
                {
                        bool validTermCheck=true;
                        for(int y=0; y<tempValidTerms.size(); y++)
                        {
                                if(words[i]==tempValidTerms[y])
                                        validTermCheck=false;
                        }
                        if(validTermCheck==true)
                                tempValidTerms.push_back(words[i]); // if the word is a
valid term; it is inlcuded into a local vector.
                }
        }

        for(int a=0; a<tempValidTerms.size(); a++) // a loop to find the number of
repetitions of the valid word
        {
                for(int b=0; b<words.size(); b++)
                {
                        if(tempValidTerms[a]==words[b])
                                termCounter++;
                }
                validTerms.push_back(tempValidTerms[a]); // including the created
object into the private data member
                wordCount.push_back(termCounter);
                termCounter=0;
        }
        temp.clear();
        tempValidTerms.clear();
        sorting(validTerms, wordCount); // performing a sort algorithm for the private
data memeber that includes instances of objects of the Word class
        setApprovedValidTermsandWordCount();
}
// a utility member function that converts all upper caseletters to lower ones. This is
to make sure that if a word is included more than once with lower and upper case
letters, they will be treated the same.
void DocumentAnalyzer::upperToLower(string & str)
{
        string tempString;
        tempString.clear();
        int stringSize=0;
        stringSize=str.length();
```

```
        for(int t=0; t<stringSize; t++) //a loop to iterate through a stringconvering all
upper case letters to lower ones
        {
                string sub="00";
                sub=str.substr(t,1);
                if(sub=="A")
                        sub="a";
                if(sub=="B")
                        sub="b";
                if(sub=="C")
                        sub="c";
                if(sub=="D")
                        sub="d";
                if(sub=="E")
                        sub="e";
                if(sub=="F")
                        sub="f";
                if(sub=="G")
                        sub="g";
                if(sub=="H")
                        sub="h";
                if(sub=="I")
                        sub="i";
                if(sub=="J")
                        sub="j";
                if(sub=="K")
                        sub="k";
                if(sub=="L")
                        sub="l";
                if(sub=="M")
                        sub="m";
                if(sub=="N")
                        sub="n";
                if(sub=="O")
                        sub="o";
                if(sub=="P")
                        sub="p";
                if(sub=="Q")
                        sub="q";
                if(sub=="R")
                        sub="r";
                if(sub=="S")
                        sub="s";
                if(sub=="T")
                        sub="t";
```

```
                if(sub=="U")
                        sub="u";
                if(sub=="V")
                        sub="v";
                if(sub=="W")
                        sub="w";
                if(sub=="X")
                        sub="x";
                if(sub=="Y")
                        sub="y";
                if(sub=="Z")
                        sub="z";
                tempString.append(sub);
        }
        str=tempString; // modifyingthe initial passed string
}
// a utility function to perform a sorting algorithm
void DocumentAnalyzer::sorting(vector<string> & vecS, vector<int> & vecInt)
{
        for(int i=0; i<vecS.size(); i++)
        {
                int maxVal=0, maxPos=0;
                string maxString="00";
                maxVal=vecInt[i];
                maxPos=i;
                maxString=vecS[i];
                for(int l=(i+1); l<vecInt.size(); l++)
                {
                        if(vecInt[l]>maxVal)
                        {
                                maxVal=vecInt[l];
                                maxPos=l;
                                maxString=vecS[l];
                        }
                }
                vecInt[maxPos]=vecInt[i];
                vecInt[i]=maxVal;
                vecS[maxPos]=vecS[i];
                vecS[i]=maxString;
        }
}
// a member function to return the sentences stored in the private data member
sentences
void DocumentAnalyzer::getSentences() const
{
```

```
        cout<<"The sentences within the edited text after editting are: \n"<<endl;

        for(int i=0; i<sentences.size(); i++) // a loop to iterate within the main vector
        {
                for(int k=0; k<sentences[i].size(); k++) // a loop to iterate within each
vector of strings stored at each position in the main vector
                {
                        cout<<sentences[i][k]<< " ";
                }
                cout<<"\n"<<endl;
        }
}
vector<string> & DocumentAnalyzer::getInitialWordList()
{
        return words;
}
void DocumentAnalyzer::setApprovedValidTermsandWordCount()
{
        for(int i=0; i<validTerms.size(); i++)
        {
                if(wordCount[i]>2)
                {
                        approvedValidTerms.push_back(validTerms[i]);
                        approvedWordCount.push_back(wordCount[i]);
                }
        }
}
vector<string> & DocumentAnalyzer::getApprovedValidTerms()
{
        return approvedValidTerms;
}

vector<int> & DocumentAnalyzer::getApprovedWordCount()
{
        return approvedWordCount;
        }



// Project Main for Finding Potential Collocations within the Inputted Text
#include <fstream> // file stream
using std::ifstream; // input file stream
using std::ofstream; // output file stream
#include <iomanip>
#include <cstdlib>
```

```
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
//#include <math>
using namespace std;
// including the altime header file
#include "atltime.h"
#include "DocumentAnalyzer.h"
#include "CollectionAnalyzer.h"
// Global Function to calculate the mean
double mean(vector<int> v)
{
	double sum=0;
	for(int i=0; i<v.size(); i++)
		sum+=v[i];
	double mean=sum/v.size();
	return mean;
}
// Global Function to calculate standard deviation
double stdDev(vector<int> v, double mean)
{
	double sum=0;
	for(int i=0; i<v.size(); i++)
		sum+=pow((mean-v[i]),2)/v.size();
	double stdDev=sqrt(sum);
	return stdDev;
}
//Main Function
int main()
{
	//Declairing local variables
	string iS, name="00";
	int wordscount=0, validWindow=0, threshold=0, boarder=0;
	double average=0.0;
	const char *namePtr = 0;
	vector<int> wordCt;
	double avg=0.0;
	double sDev=0.0;
	CTime startTime, endTime;
	//the user is prompted to input the file name
	cout<<"Please enter the file name that contains your data to be analysed."
<<endl;
	cout<<"Make sure that the file is placed within the folder of this project\n in
the Visual Studio Directory."<<endl;
```

```
        cout<<"Make sure that the file name is spelled correctly, case-sensitive \n and
includes the extension <*.dat> or <*.txt>." <<endl;
        cin>>name;
        namePtr= name.data ( ); // casting the string into a constant character pointer
to be used
        ifstream inClientFile( namePtr, ios::in ); // declairing the input file
        // exit program if ifstream could not open file
        if ( !inClientFile )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        ofstream outClientFile2( "tfid Frequency.txt", ios::out ); // declairing the output
file
        // exit program if ifstream could not open file
        if ( !outClientFile2 )
        {
                cerr << "File could not be opened" << endl;
                exit( 1 );
        } // end if
        const char *filePtr = 0;
        string fileNameStr;
        int loopCounter=0;
        CollectionAnalyzer C1;// creating an instance of the class collectionanalyzer
        while(inClientFile>>fileNameStr)
        {
                loopCounter++;
                filePtr=fileNameStr.data();
                ifstream inDataBaseFile( filePtr, ios::in ); // declairing the input file
                // exit program if ifstream could not open file
                if ( !inDataBaseFile )
                {
                        cerr << "File could not be opened" << endl;
                        exit( 1 );
                } // end if
                string tempStr;
                while(inDataBaseFile>>tempStr)
                {
                        string ex, space=" ";
                        getline(inDataBaseFile, ex);
                        iS.append(tempStr);
                        iS.append(ex);
                        iS.append(space);
                }
```

```
                DocumentAnalyzer D1(iS); // creatig an instance of a
DocumentAnalyzer class object
                D1.documentProcessing(); // performing document processing
operations on inputed text

                if(loopCounter==1)
                {
                        C1.setInitialCollection(D1.getApprovedValidTerms());
                        C1.setInitialFrequency(D1.getApprovedWordCount());
                }
                if(loopCounter>1)
                {
                        C1.searchLoop(D1.getApprovedValidTerms(),
D1.getApprovedWordCount(), loopCounter);
                }

                D1.~DocumentAnalyzer();
                iS.clear(); // clearing out the initial string to be ready to recieve a new
one
        }
        C1.approvedMatrix();
        C1.processOriginalSpace();
        C1.TermFrequencyWeight();
        C1.implementNewSpace();
        for(int o=0; o<C1.getInitialCollectionSize().size(); o++)
        {
                outClientFile2<<left<<setw(25)<<C1.getInitialCollectionSize()[o]<<" ";
                for(int p=0; p<C1.getpFrequencySize().size(); p++)
                {

        outClientFile2<<left<<setw(10)<<C1.getpFrequencySize()[p][o]<<" ";
                }
                outClientFile2<<endl;
        }
        return 0;
        }
```