Spring 5-2020

FRAUD DETECTION USING DATA-DRIVEN APPROACH

Arianit Mehana

Computer Science and Engineering program

FRAUD DETECTION USING DATA-DRIVEN APPROACH

Bachelor Degree

Arianit Mehana

May/2020

Prishtinë

Computer Science and Engineering program

Bachelor Thesis
Academic year 2017/2018

Arianit Mehana

FRAUD DETECTION USING DATA-DRIVEN APPROACH

Mentor: Dr. Krenare Pireva Nuci

May/2020

This paper has been compiled and submitted to meet the partial requirements
for the Bachelor Degree

ABSTRACT

The extensive use of internet is continuously drifting businesses to incorporate their services in the online environment. One of the first spectrums to embrace this evolution was the banking sector. In fact, the first known online banking service came in 1980. It was deployed from a community bank located in Knoxville, called the United American Bank. Since then, internet banking has been offering ease and efficiency to costumers in completing their daily banking tasks.

The ever increasing use of internet banking and the large number of online transactions, increased fraudulent behaviour also. As if fraud increase wasn't enough, the massive number of online transactions further increased the data complexity. Modern data sources are not only complex but generated at high speed and in real time as well. This presents a serious problem and a definite reason why more advanced solutions are desired to protect financial service companies and credit card holders.

Therefore, this thesis aims to construct an efficient fraud detection model which is adaptive to costumer behaviour changes and tends to decrease the fraud manipulation, by detecting and filtering fraud in real-time. In order to achieve this aim, a review of various methods is conducted, adding above a personal experience working at a Banking sector, specifically in Fraud Detection office. Unlike the majority of reviewed methods, the proposed model in this thesis is able to detect fraud in the moment of occurrence using an incremental classifier.

The evaluation on synthetic data, based on fraud scenarios selected in collaboration with domain experts that replicate typical, real-world attacks, shows that this approach correctly ranks complex frauds. In particular, our proposal detects fraudulent behaviour and anomalies with up to 97% detection rate while maintaining a satisfying low cost.

ACKNOLEDGEMENTS

First of all, I am extremely grateful for my family which motivated and supported me throughout my whole career.

I am also thankful to my mentor, Dr. Krenare Pireva, for her guidance through each stage of the process.

Last but not least, I would like to thank my workplace supervisor and other colleagues for inspiring my interest in this domain with useful information and advices.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

FP- False Positives

FN- False Negatives

CRM- Customer Relationship Management

MRF- Markov Random Field

Bel- Belief

Pl- Plausibility

CBLOF- Cluster-Based Local Outlier Factor

HBOS- Halifax Bank of Scotland

DBSCAN- Density-Based Spatial Clustering of Applications with Noise

GNN- Graph Neural Networks

GCM- Global Constants Module

GUIM- Core/Graphical User Interface Module

DBIM- Database Interface Module

LAL- Learning Algorithms Library

LAIM- Learning Algorithm Interface Module

RMSE- Root Mean Squared Error

AIS- Artificial Immune Recognition algorithm

ID3- Iterative Dichotomiser 3

FS- Fraud Score

PCA- Principal Component Analysis

TL- Legal Transactions

TF- Fraud Transactions

MMPP- Markov Modulated Poisson Process

RM- Rule-based Model

SVM- Support Vector Machine

CHAID- Chi-Square Automatic Interaction Detection

DT- Decision Tree

RBF- Radial Basis Function

HT- Hoeffding Tree

NBU- Naïve Bayes Updateable

KNN- K-Nearest Neighbours

WEKA- Waikato Environment for Knowledge Analysis

AFDM- Active Fraud Detection Model

1 INTRODUCTION

The Internet has been around for decades. Many people have been using it to facilitate their lives and expedite their daily tasks. Of all the aspects of daily life that have benefitted from the internet, the banking sector has been especially effective at capitalizing on internet's features. It has introduced many attractive ways to increase the scope of its financial services. The emergence of internet banking has allowed banks to offer their customers relatively convenient and flexible banking, also known as e-banking [1].

Although there are many advantages of online banking, security issues often discourage customers from using it. This is evolving as many customers have found that the use of online banking could leave their financial assets at risk due to fraudulent activity.

Fraud is defined as wrongful or criminal deception intended to result in financial or personal gain, or to damage another individual without necessarily leading to direct legal consequences [2]. This ever-growing market urged the need for particular attention to a counter mechanism in order to tone the losses down, which only in the last decade have managed to increase 56,5% globally [3].

So far, there are different approaches from a number of researchers that in one way or another have proposed techniques to detect these activities, but there is lack of research on detecting these activities in real-time situations. Therefore, this thesis tackles this gap, by proposing a fraud detection approach which uses *instance-incremental learning.* This methodology increments its knowledge instance by instance which actively and adaptively recognizes such activity in order to prevent it from reaching the final state.

The rest of this thesis is organized as follows: *Chapter 2* presents an analyse on the actual fraud detection mechanisms. The main drawback of all these examined systems is described in *Chapter 3*. *Chapter 4* contains the methodology of the system, while *Chapter 5* presents the detailed development process of the proposed model. In *Chapter 6-7* are presented the results, respectively the evaluation methods for the proposed model, compared to actual approaches. This leads to the last *Chapter*, that concludes this thesis and unveils plans for future work.

2 LITERATURE REVIEW

This chapter offers a brief explanation of the banking sector and the online banking environment. It also breaks down the most frequent challenges that this environment faces and the countermeasures that are used. While there is a large number of methodologies used for detecting fraud, the most successful ones can be found elaborated below.

2.1 Banking Sector

A bank is a financial institution, licensed by a central bank, that handles banking activity such as deposit, credit and financial transactions. Banking is one of the key drivers of a country's economy. It uses deposits to provide loans which costumers use for personal or business purposes. Most of the products that a bank offers, include an interest amount. The interest is the most traditional method of revenue generation including transaction fees and financial advices. There is a variety of bank types including [1]:

- Commercial banks are the most widely-spread banks, they provide services to private individuals and businesses.
- Community banks are smaller than commercial banks. They focus on the local market and provide more personalized services.
- Private banks are banks that manage the assets of high-net-worth individuals.
- Investment banks are a different type of banks that provide investment management and advise corporations on capital market activities.
- Merchant banks are classified as banks that provide capital to firms in the form of shares rather than loans.
- Islamic banks are a form of banking that respects the concepts of Islamic law. Therefore, they avoid interest charges.

All these mentioned banks that may operate in a country are regulated by the Central banks.

Central banks are usually owned by the government and charged with regulatory responsibilities. They provide liquidity to the banking system and act as the lender in event of a crisis.

2.2 Bank Services

There is a variety of services a bank offers that are separated into two categories: individual banking and business banking. Individual banking services include: current accounts, saving accounts, loans, debit and credit cards etc. These services assist individuals in managing their finances. Unlike individual baking, business banking helps business owners differentiate their professional finances from personal ones. Business banking services do not differ from individual ones, expect that they are offered with different interest rates and conditions. Thanks to online banking, nowadays, most of these services can be managed from a computer, tablet or smartphone. Online banking, also known as internet banking or e-banking is an electronic system that allows customers to conduct most of the banking activities such as viewing account balances, obtaining statements, checking recent transactions, transferring money, applying for a credit product and so much more. It has revolutionized the banking industry, giving the costumer much more accessibility in an instant time without the need of visiting a branch.

2.2.1 The problems arisen with online services

Along with the possibilities offered to the user, there comes a big challenge which is information security. Mostly, this sensitive data is accessed using phishing, which is a criminal activity that uses social engineering techniques and enables phishers to fraudulently acquire sensitive information, by masquerading as a trustworthy person in an electronic communication [2]. Phishers use different techniques to achieve their purpose which may include: “man in the middle”, deceptive, and malware attacks.

During the “man in the middle” attack, the attacker places himself in between the bank and the costumer while the costumer is using his online banking account. Therefore, the attacker can either steal the information or change it to benefit his/her purpose.

In deceptive attacks the phisher sends a deceptive message to the costumer in order to lure him/her to interact immediately. If the costumer interacts, he/she gets redirected to a legitimate-looking page where he/she is asked to enter his/her sensitive information. This information is then used for fraudulent activity. Lastly, malware-based phishing refers to software programs that the attackers install on costumers’ computers, which are later used to handle the information needed.

2.2.2 Online banking attack countermeasures

The best practice to eliminate any kind of fraudulent action would be to stop them before they occur. This process is known as fraud prevention. Fraud prevention is the proactive mechanism with the goal of disabling the occurrence of fraud. Most of the financial institutions have a number of techniques that they use to prevent fraud. One of them is the use of personalized emails. A personalized email is structured to possess personal information of the costumer that it refers to, which is not the case with deceptive emails. This helps the costumer understand the origin of the email.

Some other precautions are protection software and two-factor authentications. Protection software is very effective against malware attacks, but in most cases this measure gets bypassed. That is why the two-factor authentication is also used. Combined they counter most of the phishing attacks. Two-factor authentication requires two different types of evidence to establish the identity, which makes it a very difficult step to be bypassed by fraudsters. With all the improvements in the prevention process, expert fraudsters not rarely manage to breach the security system of the bank. These cases should be detected in order to block them from reaching their final state.

2.3 Fraud detection using data-driven techniques

Fraud detection systems come into play when the fraudsters surpass the fraud prevention systems and start a fraudulent transaction. Accordingly, the goal of a fraud detection system is to check every transaction for the possibility of being fraudulent regardless of the prevention mechanisms, and to identify fraudulent ones as quickly as possible after the fraudster has begun to perpetrate. In order to accomplish that, financial institutions use a variety of techniques that can be grouped in three main categories [2]:

- Traditional techniques.
- Machine learning techniques.
- Hybrid techniques.

Besides fraud detecting, the mentioned techniques are widely used in other areas of the banking sector such as [2]: customer retention, marketing, risk management and CRM.

2.3.1 Traditional techniques for fraud detection

Traditional techniques are probably the oldest and most time-proof ones. They consist of defining certain rules and label actions that match them, as anomalous and potentially worth checking. These rules are defined by experts of the financial institutions; therefore, the efficiency of the system depends fully on the them. An example of this kind of approach is presented in [4].

The author in [4] proposed a rule-based system to help alert banks and other financial institutions in case of fraudulent activity on consumer credit, particularly with credit cards. Main focus is to detect fraud during the authorization process, in order to allow the institution to communicate with the client and compile the decision accordingly. The system is flexible and can be re-defined anytime. The system in [4] defined rules based on predictive fraud variables and thresholds, identified in complete harmony with the bank representatives. The predictive fraud values and thresholds are presented in *Table 1*.

Table 1. Predictor fraud variables with thresholds identified [4]

Predictor variables as defined in rule base
number of transactions last 24 hours
number of transactions last hour
number of transactions over $1000
number of transactions over $500
number of transactions at the same merchant
change in dollar value of previous transactions
transaction time of day

According to these values, the final rules were constructed as:

Table 2. Example of rules [4]

Final Rules
If dollar value of transaction is greater than $1000 then investigate further. If not, review next transaction.
If time of day is between 8pm and 6am, investigate further. If not, review next transaction.
If time since last transaction is less than 30 minutes, then investigate further. If not, review next transaction.
If customer at maximum balance ever, call customer. If not, investigate further
If there have been previous purchases within last 24 hours at the same merchant, call customer. If not, review next transaction.

After construction, the system classifies each active account on the dataset as either fraudulent or legitimate. In order to classify accounts, transfer information for each one is downloaded every hour. Accounts that fulfil any of the rules, will be written in a report that will be send to the fraud department to be investigated further. Analysts will then try to

communicate with the costumer and conclude the decision for that account. Accounts will not be blocked unless there is a fraud report from the costumer.

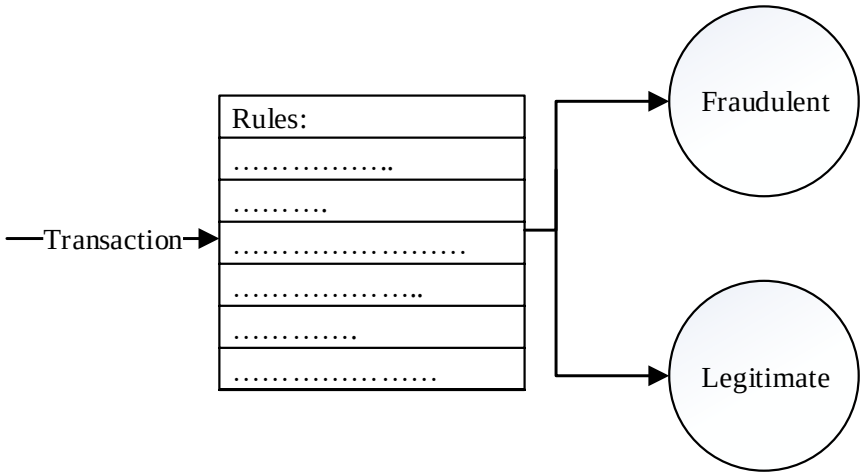


Figure 1. Architecture of the rule-based system

The model's performance was measured based on classification accuracy and the cost of misclassification. The cost of misclassification was defined in terms of 'good accounts disturbed'.

Table 3. Test results of the expert system [4]

Classification results		
	Legitimate accounts	Fraud accounts
Classified as legitimate	10933	113
Classified as fraudulent	1199	465
Total	12132	578

As shown in *Table 3*, from a total of 12710 accounts, 11398 were correctly classified. Which means that this expert model was able to classify with an 89.68% accuracy overall and a misclassification cost of 2112 accounts.

This system is a very fast and efficient approach that affects the dataset immediately. It also doesn't require a lot of hardware specifications to run which is a point of interest for the market. Although, it is fast and easy to run as a technique, generating reports every hour makes it hard to detect fraudulent behaviour at the moment of initiation. A better solution would be to analyse transactions instead of accounts. This way, an alert will be created the moment a suspicious transaction happens. In case a transaction is caught as suspicious, only that transaction would be blocked and not the whole account.

2.3.2 Machine Learning techniques for fraud detection

Machine learning techniques are concerned with general pattern recognition or the construction of universal approximations of relations in the data in situations where no obvious a priori analytical solution exists [5].

Learning process can be done in a supervised environment or an unsupervised one. In supervised learning, the aim is to learn a mapping from the input to an output, whose correct values are provided by a supervisor. In unsupervised learning, there is no such supervisor and there is only input data. The aim is to find the regularities in the input.

Every machine learning system should be trained and tested before evaluation. Usually 70% of the data in a dataset is used for training and the remaining part for testing. That is why each dataset is divided in two parts.

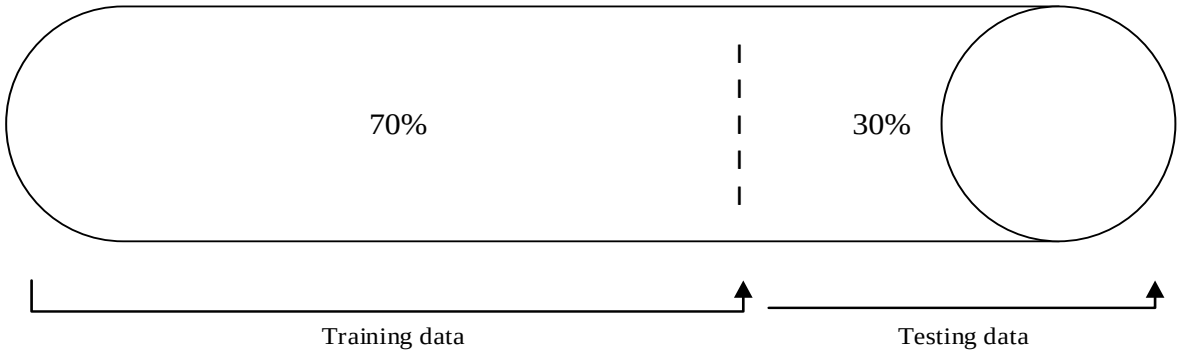


Figure 2. Dataset split practice [5]

2.3.2.1 Supervised machine learning techniques

For supervised learning, best performing methods are: Naïve Bayes, KNN, graphs, artificial neural networks and support vector machines [6]. The defining characteristic of these approaches is that they take the target variable as an input to the function. Some of these methods are presented in [7-10].

Both researchers in [7] and [8] use belief propagation to denote the final score for an instance whether that is a device or an account. However, their learning methodology differs.

The proposed system in [7] runs two analyses on device basis (as shown in *Figure 3*), anytime a new transaction is made.

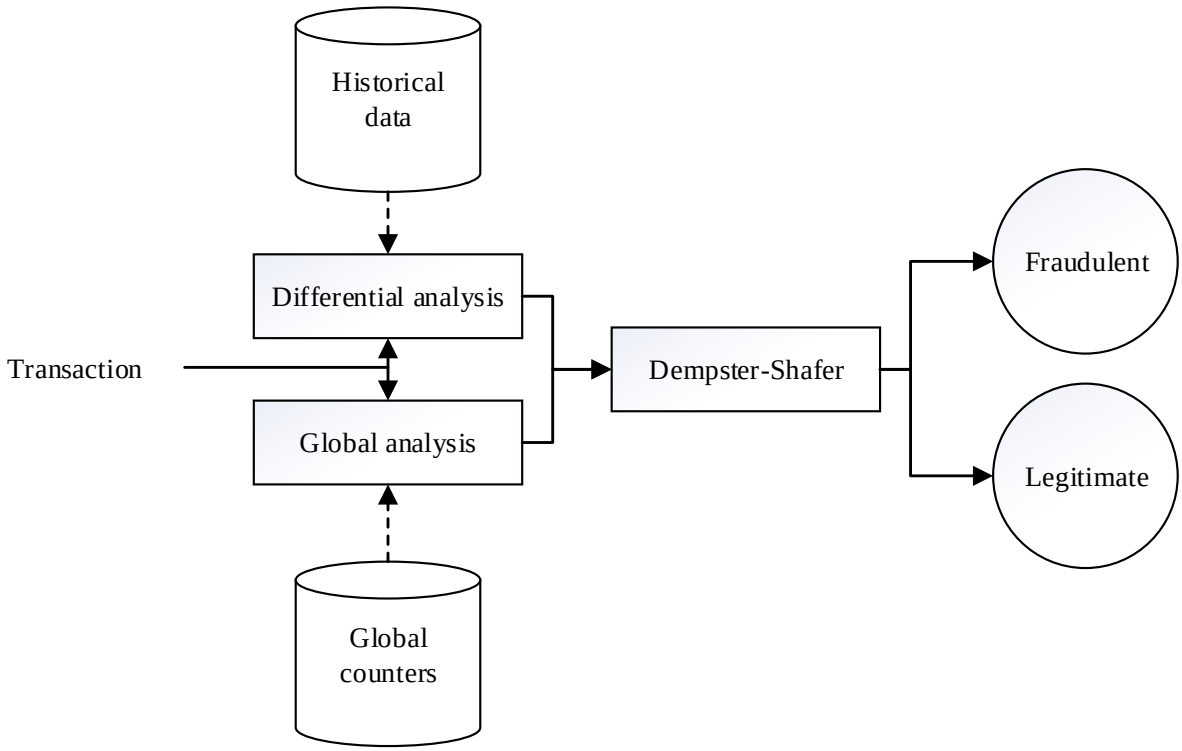


Figure 3. Architecture of the system [7]

The first analyse is called differential analyse because it highlights transactions that deviate from the average user's behaviour. As shown in *Figure 4*, this is achieved using two buffers. The first buffer contains all the transactions made in the actual session, which represent the

current usage pattern. The second one contains the most recent transactions. It includes transactions from an institution-defined earlier date, until the date the system is run (excluding the latest session transactions), which represent the average usage pattern. The deviation is calculated using a statistical method, the result of which is a probabilistic value that gives a fraudulent belief for that device.

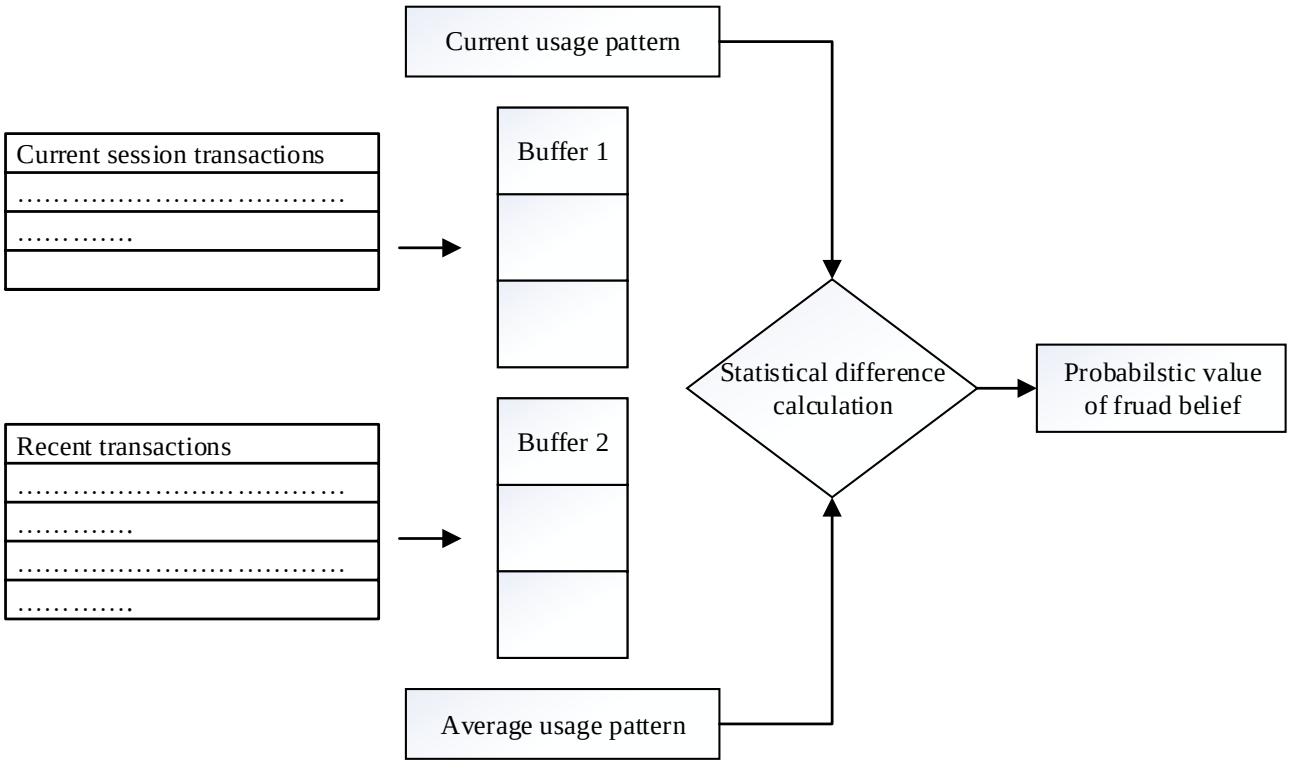


Figure 4. Differential analysis process

If there is a difference between the current and average usage pattern, that device is then passed to the global analyse. In this step, the particular device that attempted the transactions is checked whether it owns other accounts and the nature of those additional accounts. This analyse is done to investigate further about that device and to strengthen or weaken the evidence of fraud from the first analyse. After this analyse, the devices are then listed either in the white list or the black list. The black list contains devices classified as fraud while the white list contains those classified as legitimate. Other devices that are waiting to be

classified are kept in the suspect list. The classifying is done using an exponentially decaying function which is expressed as [7]:

$$P(t) = Pmax \cdot e^{-\lambda t}, \quad (1)$$

where,

P_{max} is the maximum probability value assigned to the device. This function depends on the number of different accounts accessed by the device (N), since the probability of being a fraud increases with this number.

These two evidences concluded from the analysis are than combined using a mathematical theory of evidence which is called the *Dempster-Shafer theory*. This theory, with the help of the *Belief function* and the *Plausibility function* as expressed below:

$$Pl(H) = 1 - Bel(-H), \quad (2)$$

$$U(H) = Pl(H) - Bel(H), \quad (3)$$

defines the final suspicion score. Based on this score, an account on a given device can be detected as fraudulent or legitimate [7]. No matter how high the evidence of fraud is, if there is no costumer fraud report during the actual day, that device will be listed as non-fraudulent. Otherwise, if the costumer reports fraudulent activity, all the particular accounts related with that device will be blocked.

In the other system presented in [8], the dataset is learned using the Markov Random Field model. MRF model is an undirected graph where nodes represent costumers and edges represent transactions.

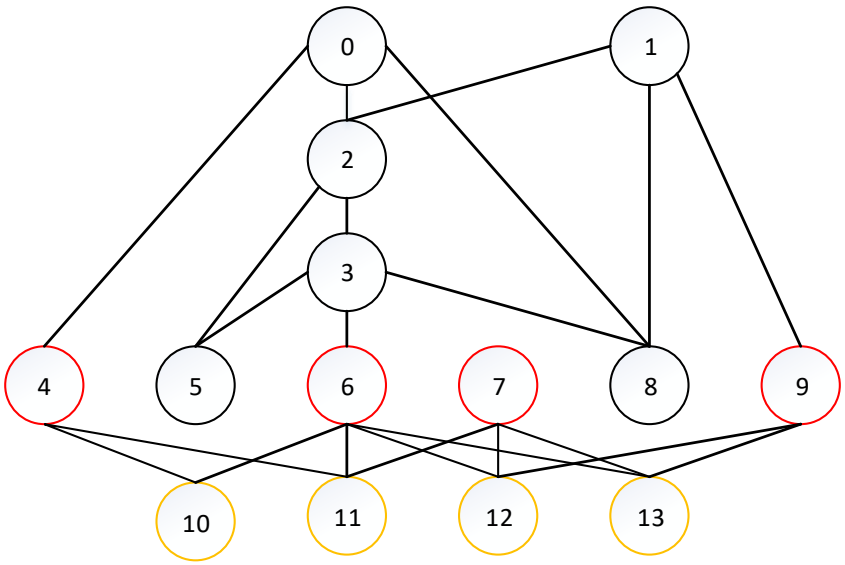


Figure 5. MRF graph in NetProbe [8]

The model presented in this system has managed to understand a new layout of fraud networks. Fraudsters (red nodes in *Figure 5*) cooperate with accomplices (yellow nodes in *Figure 5*) that don't perform any fraudulent activity. This way, these last ones try to appear completely legitimate to the system. If the dependency between nodes would not be calculated, most of the accomplices would continue performing without being noticed.

The dependency between a node and its neighbours is represented by the *Propagation matrix (Ψ)* where *Ψ(i,j)* equals the probability of a node being in state *j* given that it has a neighbour in state *i*.

Each node communicates with other nodes via message passing $m_{ij,}$ which is the key factor that affects a node's belief of being fraudulent b_i:

$$b_i(\sigma) = k \prod_{j \epsilon N(i)} . m_{ij}(\sigma) \qquad (4)$$

where m_{ij} is the message vector sent by node *i* to *j*

N(i) is the set of nodes neighbouring *i*

k is a normalization constant.

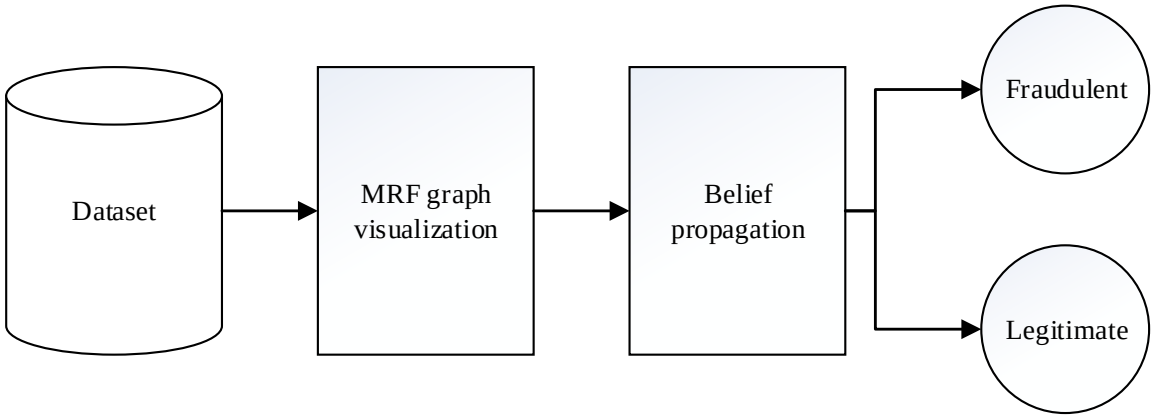


Figure 6. Architecture of NetProbe [8]

After the dataset is visualized in the graph and belief propagation is calculated for each node, the system presents fraud scores for each costumer so that they can be noticed and investigated further by the bank.

While the system proposed in [7] has not been validated yet, the authors in [8] ran the system in a synthetic dataset where accuracy was measured by precision and recall. Precision is the number of right guesses for fraudulent nodes while recall is the number of right guesses of nodes belonging to a fraudster.

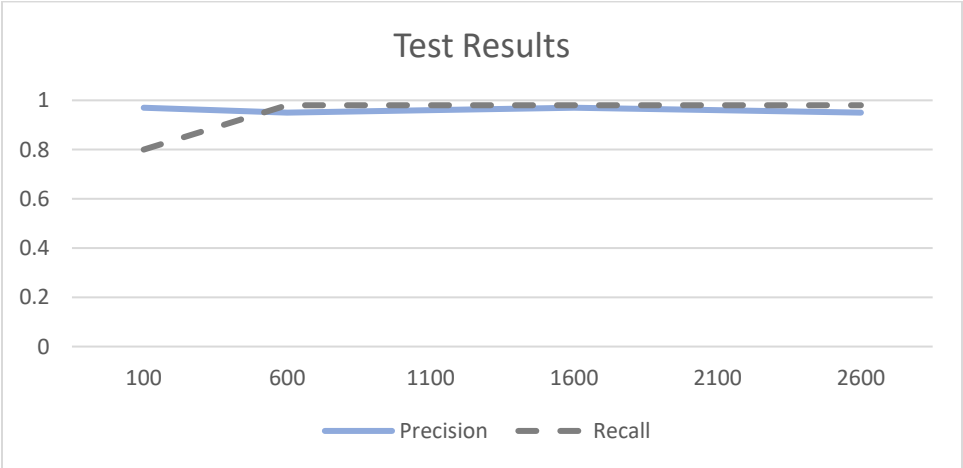


Figure 7. NetProbe Test Results [8]

As we can see from *Figure 7*, with the increase of the costumers the recall has also a tendency of increase, while the precision is stable at nearly 1 (0.98 average). This means that 98% of the nodes (costumers) are being correctly classified.

While both of these systems in [7] and [8] implement similar approach on fraud scoring, the method proposed in [8] resulted to be a better classifying method. For instance, local profiling in [7] is created based on the most recent customer's transactions, not all of the customer's transactions as in [8]. When creating a user(local) profile all of the transactions should be aggregated, in order to create the most accurate spending pattern of that costumer. Another drawback in [7] is the use of the exponentially decaying function. The decaying function in [8] is time variant. The higher the time difference between the time of the transaction and the scoring time, the lower the transaction fraud score will be. And if the time window closes without costumer report, devices by default are ranked as non-fraudulent. In order to eliminate the time variance, devices should be classified using anomaly detection algorithms such as CBLOF and HBOS. Additionally, in absence of costumer report, a device should never be rated as non-fraudulent and so allowed to perform. On the contrary, the system should try to establish a communication with the costumer. And if there is no costumer response, that suspicious transaction should be blocked. Although, the approach in [8] can be seen as the successor of the one in [7], it presents some deficiencies. The use of graphs, limits the scalability of the system. Both the time and space requirements of NetProbe increase proportionally to the number of edges in the graph. This increase is caused by the propagation matrix. A solution would be to automatically learn this matrix from available data which can be done using graph neural networks (GNN) which are used in [9].

Figure 8, depicts a system named Cardwatch [9]. Cardwatch uses neural networks approach in order to detect fraud on credit card transactions. It is a system organized in five modules [9]:

- Global Constants Module (GCM):

  The purpose of this module is to gather all the global variables declared in the system.

- Core/Graphical User Interface Module (GUIM):

  It is a module that serves as a container for all GUI-related routines, including the call-back code or auxiliary functions for widget control. Moreover, this module handles the creation of neural network description files, which are then accessed by the LAIM module and forwarded to the LAL module [9].
- Database Interface Module (DBIM):

  This module handles the communication between the database and the remaining modules. It contains the code for such operations as initialization, opening and modification of databases.
- Learning Algorithms Library (LAL):

  This module provides the neural network learning algorithms. It is independent of the core part of the system while retrieving transaction data or marking fraudulent records. This keeps the interfaces to the core highly efficient [9].
- Learning Algorithm Interface Module (LAIM):

  This module provides a bridge between the core and the neural network library. It has two functions inside: train and test with method dependent calls to LAL module.

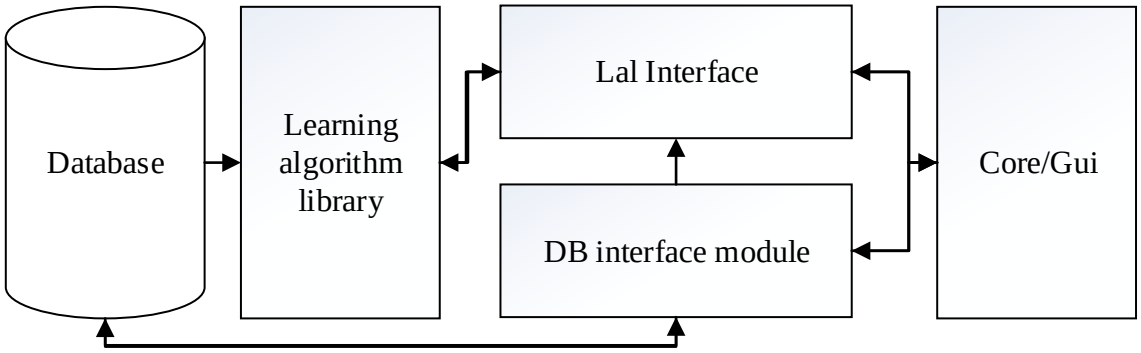


Figure 8. Architecture of Cardwatch [9]

Learning rules defined in LAL include: conjugate gradient, backpropagation with momentum, and batch backpropagation with momentum. After the user decides on of these rules, the transactions are classified as fraud or legitimate using the root mean square error (RMSE) described as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(t_n - o_n)\text{^}2}\,, \tag{5}$$

where *N*-the number of transactions

t-time of transaction *n*

o-output of *n.*

While in [9], selecting a learning rule is mandatory, the system in [10] has a self-regulating characteristic that staves off this need.

[10] is an Artificial Immune Recognition System that uses the artificial immune recognition algorithm (AIS) to detect fraudulent behaviour on user basis. AIS is a class of adaptive or learning computer algorithms, inspired by the function of the biological immune system.

Before implementing AIS in this specific system, a dataset analysis function is added in order to examine the dataset characteristics. That function is called alpha index [10]:

$$alpha_{ij} = \frac{N_{ij}}{N_f} \tag{6}$$

$alpha_{ij}$ is the fraud ratio for the j_{th} value of i_{th} field, and N_{ij} shows the number of fraudulent records having j_{th} value for the i_{th} field, and N_f is the whole number of fraudulent records in the dataset.

Alpha indexing the values on a dataset helps AIS algorithm produce the fraud and legal detectors. Fraud transaction detectors demonstrate the fraud patterns in the dataset. While legal transaction detectors demonstrate legal user behaviour. Legal transaction detectors are generated specifically for each user because each one has different shopping habits.

When testing a particular user's new transaction, to determine its class (either normal or fraud), the k neighbours are chosen from all fraud detectors, and the same user's legal detectors.

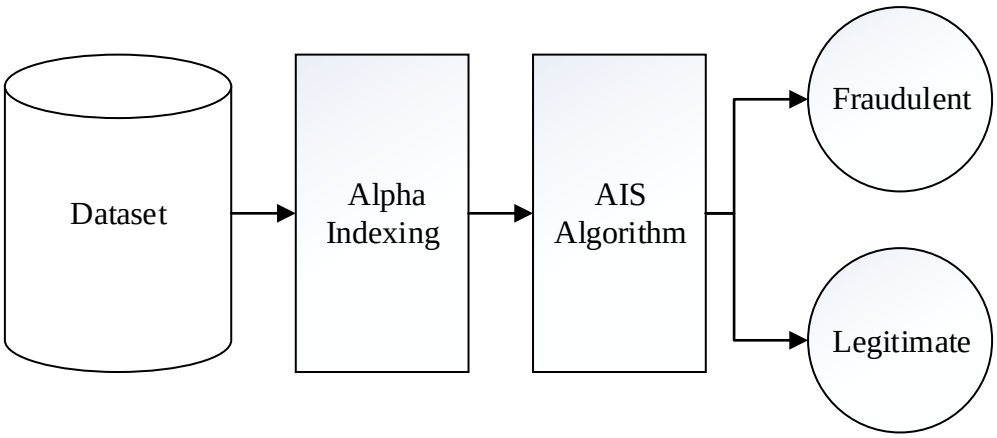


Figure 9. Architecture of AIRS

If the new transaction doesn't comply with the legal detectors but goes along with a fraud one, it is ranked as fraudulent. Otherwise, if it suits with any legal detector, it is ranked as legitimate.

The system in [9] was tested on a dataset consisting of 112 transactions, were 53 were fraudulent ones.

Table 4. Test results of Cardwatch [9]

Fraud detection			
	Actual	Detected	Percentage
Legal Transactions	59	67	100
Fraudulent transactions	53	45	85
Total	112	112	

As we can see from Table 4, the system managed to detect all legal transactions but missed 8 fraudulent ones. This resulted in the fraud detection rate of 85%.

A better result was achieved in the system presented in [10]. The testing in this approach was made on a dataset consisting 0.2% fraudulent transactions. This system managed to detect 100% of fraudulent transactions with 10% false positive rate.

The low detection rate in [9] is primarily a product of unsophisticated neural network learning techniques and system input limitations. To satisfy this gap, delta learning rule may be used. It is a supervised learning rule that works with multiple input units and minimizes error by reducing the difference between the actual and expected output.

These limitations are not present in the methodology in [10]. The system in [10] choses the best learning rule adaptively and can accept a relatively big number of inputs. While, it detects all of the fraudulent transactions, it disturbs a number of legal users as well. This happens because of the outdated detectors. The detectors that have not been productive for a defined period of time should be deleted in order to lower the rate of false positive alerts.

2.3.2.2 Unsupervised machine learning techniques

In the other hand, unsupervised learning techniques include: k-means clustering, hierarchical clustering, neural networks, anomaly detection and decision trees. Most unsupervised algorithms aim to find a clustering or group structure in the data, which has to be interpreted by the researcher [6]. Some examples of unsupervised approaches are presented in [11-13] research work.

In the approach presented in [11], fraud is detected by constructing user profiles and excluding deviations using anomaly detection algorithms. Initially, for each user a local, global and temporal profile is created as shown in *Figure 10*.

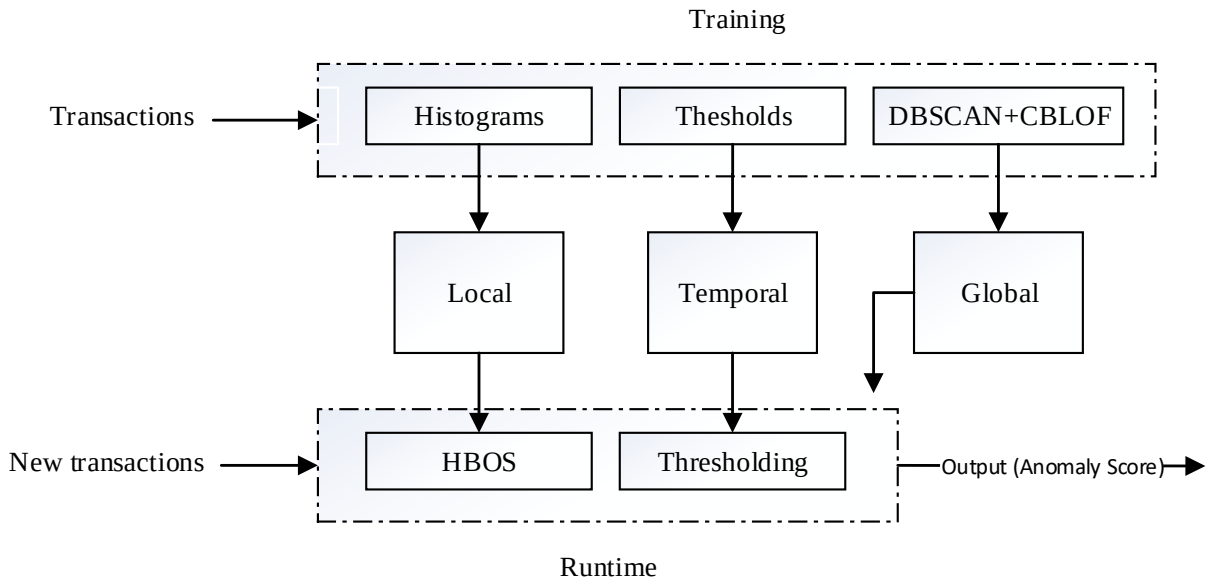


Figure 10. Architecture of BankSealer [11]

Local profiling is done to characterize each user's spending pattern. This profile is created by calculating the empirical marginal distribution of the features of each user's transaction visualized in a histogram. After a profile is created, each new transaction of that account is scored using *HBOS* which is an anomaly detection algorithm. This profiling can be done on those accounts that have performed at least three transactions. Accounts that don't fulfil this condition are denoted as under-trained or new accounts. In these cases, global profiling is considered.

Global profiling uses *agglomerative hierarchical clustering* to group accounts based on their similarities, in a big cluster. The groups with a sufficient density are than grown into clusters using an iterative version of *DBSCAN* which is a density-based clustering algorithm. This profiling assigns each account a global anomaly score using *unweighted-CBLOF*. The more a user deviates from the dense cluster of "normal" users, the higher the global anomaly score will be [11].

Temporal profiling is made to deal with frauds that exploit the repetition of legitimate-looking transactions over time [11]. This profiling is made only for users that have a sufficient amount of past transactions, because those are the accounts that are most-likely

fraudulent. During training of this profile, the total amount and the maximum daily number of transactions is aggregated for a pre-defined time window. At runtime, the difference between the current window values and the average ones combined with a particular threshold, produces the anomaly score.

After an account has been evaluated by the three profiles mentioned above, a final score is calculated using an exponential discount factor in terms of a time window W.

The authors in [11] ran tests in different scenarios such as: information stealing, transaction hijacking, stealthy fraud and mixed frauds. These test scenarios were done on a dataset consisting 9 months of data from a bank with 718,927 transfers.

Table 5. Test results of BankSealer [11]

Correctly ranked frauds(%)				
Users: / Fraud Scenario:	Overall	Well-trained	Under-trained	New
Information Stealing	96	96	99	93
Transaction hijacking	65	46	90	60
Stealthy fraud	59	44	90	44
Mixed fraud	71	73	85	58
Total	73			

Table 5 summarizes the percentage of correctly ranked transactions overall, for well-trained users only, for undertrained uses only, and finally for new users only. Overall fraud detection rate resulted to be 73%. As we can see from *Table 5*, this system was not so efficient in detecting stealthy fraud. Stealthy fraud is a fraudulent behaviour where the fraudster tries to blend in with the user's behaviour.

This deficiency could be reduced with the implementation of a rule-based filtering along with this system presented in [11], based on the fact that a rule based system doesn't depend on user behaviour.

Unlike this system, the systems presented in [12] and [13] use k-means clustering in order to group the users. The one presented in [12] is called BOAT. Besides k-mean clustering, boat implements decision trees and the boat algorithm as shown in *Figure 11*.

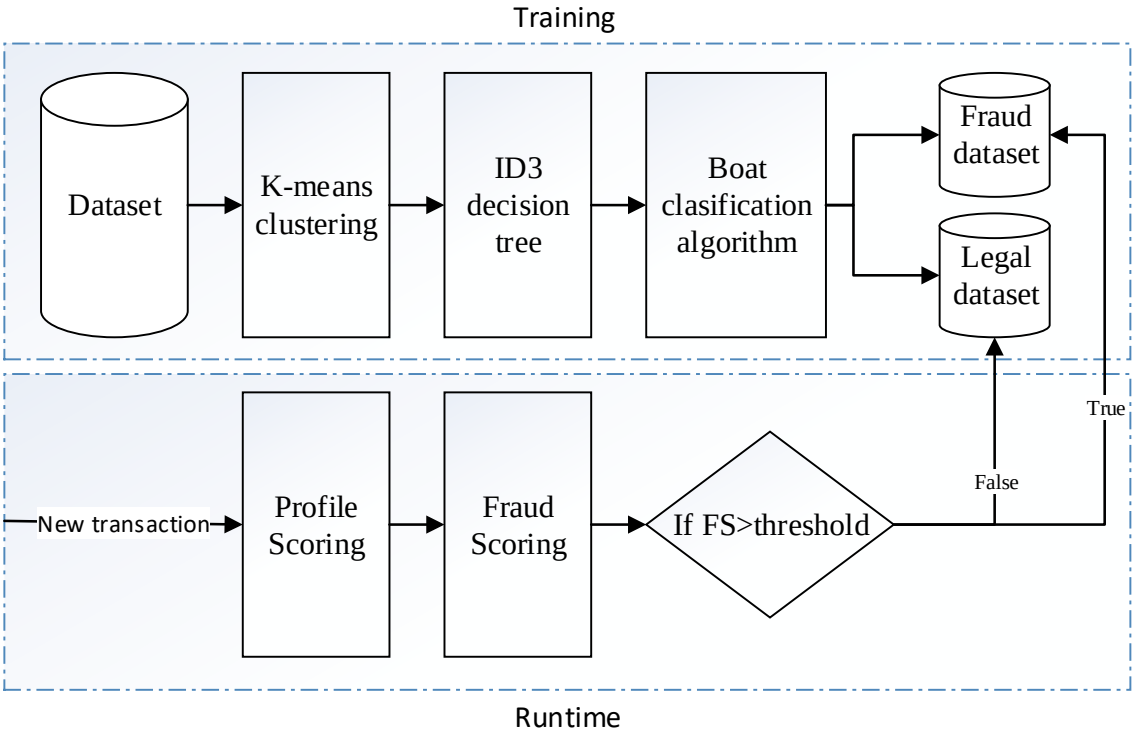


Figure 11. Architecture of BOAT [12]

K-means algorithm clusters all of the accounts by minimizing the sum of squares of distances between each data point and the centroid of the cluster to which they belong. Afterwards, the best attributes from the cluster are extracted using the ID3 algorithm. ID3 is a decision tree algorithm which classifies each account based on the extracted attributes using the function of *Entropy*, described as [12]:

$$Entropy(D^1) = \sum_{i=1}^{c} P_i \log_2 P_i, \tag{7}$$

where p_i is the probability of the subset D^1 belonging to class i.

The decision tree must be updated any time a new transaction is performed, which is why the proposed system used the BOAT algorithm. BOAT is a scalable algorithm that can incrementally update a decision tree when the training dataset changes dynamically [12]. All other decision tree algorithms require separate database scans for each level of the tree but BOAT algorithm constructs several levels of the tree in a single scan over the database.

At runtime, each new transaction goes through two stages. In the first stage, the transaction is compared with the legitimate transactions stored in the database and the profile score is computed. If any deviation from the normal behaviour is observed, it passes to the second stage. Second stage confirms if the transaction is due to fraudulent activity or due to short term change in spending behaviour by comparing it with the fraud history database, and calculating the deviation score. If the deviation score is higher than the user specific threshold, the transaction is denoted as fraudulent, and is inserted in the fraud history database. Otherwise, it is concluded as a short term change in the user's behaviour and the transaction is allowed.

In the other approach, presented in [13], the authors incorporated Principal Component Analysis (hereafter: PCA) algorithm together with a sophisticated version of k-means (*Figure 12*). PCA is a data analysing method which transforms correlated variables into uncorrelated ones. This method aims to represent transactions described by different attributes in a smaller subspace than initial one, and so that the least possible information is lost [13]. Using this algorithm, a matrix is built for every account. It possesses *n* transactions with *p* respective attributes:

$$X(n,p) = \begin{bmatrix} x_1 \dots x_{1p} \\ \dots \dots \dots \\ t_{n1} \dots t_{np} \end{bmatrix} \quad (8)$$

After PCA, SIMPLEKMEANS unsupervised classification scheme has been applied to classify the transactions. This algorithm consists in picking up randomly k initial points, and assigning transactions to their closest similar point. This way each transaction is identified as fraudulent or legal and the following matrix is built:

$$T = \begin{bmatrix} t_{11} \dots t_{1p} \\ \dots \dots \dots \\ t_{n1} \dots t_{np} \end{bmatrix}, \quad (9)$$

T represents all the transactions of a bank account and each transaction $T_j = \{t_{j1}, t_{j2} \dots t_{jp}\}$ is described by p characteristics. T contains both legal TL and fraudulent TF transactions.

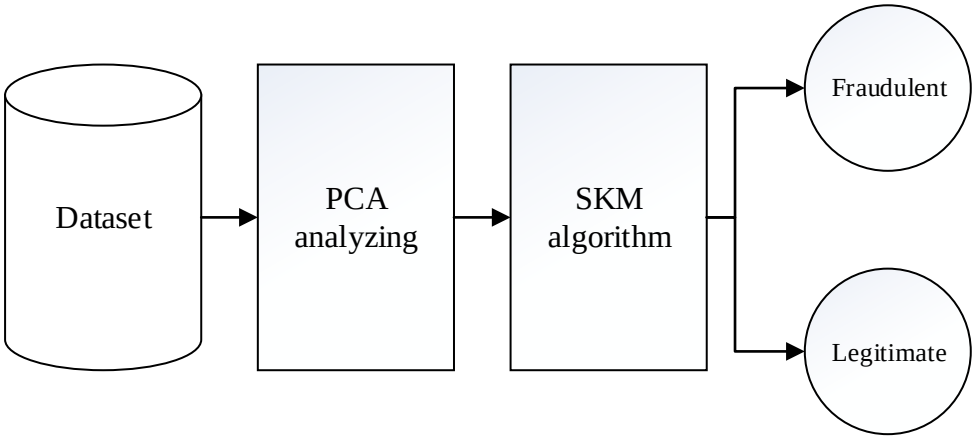


Figure 12. Architecture of the system [13]

For evaluation purposes both systems in [12] and [13] were tested using synthetic data. The authors in [16] used a Markov Modulated Poisson Process (hereafter: MMPP) to generate data. After running their system, they managed to get 85% fraud detection rate with a 10% false positive rate. In the other hand, the authors in [13] created a simplified dataset of five accounts.

Table 6. Test results of the system [13]

RESULTS FOR 5 DIFFERENT BANK ACCOUNTS		
	Legitimate transactions	Fraud transactions
Classified as legitimate	37	0
Classified as fraudulent	1	12

Total	38	11

As seen in *Table 6*, from five accounts we have a 100% fraud detection rate with one legal transaction misclassified as fraudulent.

Although, the results are acceptable, both of these systems present room for improvement. For instance, the system in [12] detects fraud by comparing a transaction with the fraud database which is populated with fraud transactions from all users. This approach is not correct because a fraud behaviour for one user may be completely normal for the other. So, transactions should be compared with the same user's history and not the general group.

The other system resulted in a mismatch of a small dataset with 49 transactions, which presents a lot of risk and concern when applied on complex datasets. A solution, especially when working with matrixes, is to apply a number of algorithm iterations before deciding the final scores.

2.3.3 Hybrid techniques for fraud detection

Hybrid techniques are a combination of two or more computational techniques which provide greater advantages on fraud detecting than individual ones. Most common combinations in hybrid methodologies include [6]:

- Traditional techniques with machine learning ones
- Supervised machine learning techniques with unsupervised ones

An example of each of these categories is presented in [14] and [15], respectively.

In [14], a hybrid model for credit card fraud detection is presented. The proposed approach combines elements of traditional and machine learning methodologies, aiming to compensate for the individual deficiencies of the methods. To detect fraud this system incorporates one-class classification and rule-based methodology at account level.

In the first stage, the system constructs a model for each of the accounts in the dataset.

The model of an account *i* consists of a set of descriptors that quantify the time-ordered series:

$count_i(t_{si})$ and $amount_i(t_{si})$.

S_i is the number of transactions of an account i in the dataset that consists only legitimate transactions. Using these descriptors, we can understand:

how much an account has spent on average:

$$amount\ on\ average_i = \frac{1}{S_i}\sum_{j=1}^{S_i} amount_i(t_j)\,; \tag{10}$$

how much does an average deviate from costumers' *amount on average:*

$$amount\ spread_i = \sqrt{\frac{1}{S_i - 1}\sum_{j=1}^{S_i}\left(amount_i(t_j) - amount\ on\ average_i\right)^2}; \tag{11}$$

how many transactions on average have been made:

$$count\ average_i = \frac{1}{S_i}\sum_{j=1}^{S_i} count_i(t_j)\,; \tag{12}$$

and how much a number of transactions deviates from the average:

$$count\ spread = \sqrt{\frac{1}{S_i - 1}\sum_{j=1}^{S_i}\left(count_i(t_j) - count\ on\ average_i\right)^2}. \tag{13}$$

Based on these parameters, different groups with similar attributes are formed. For each group, minimum and maximum boundaries are set according to the client request.

When a new transaction is done, the system compares its attributes with the average ones and denotes a score, referring to the possibility of that account to be compromised:

$$score_{amount} = \frac{1}{1 + \exp\left(-\frac{|amount(t_{new}) - amount\ on\ average_i|}{boundary\ amount_i}\right)}; \quad (14)$$

and

$$score_{count} = \frac{1}{1 + \exp\left(-\frac{|count_i(t_{new}) - count\ on\ average_i|}{boundary\ count_i}\right)}, \quad (15)$$

then both scores are combined in the final score:

$$score = score_{amount}\ x\ score_{count} \quad (16)$$

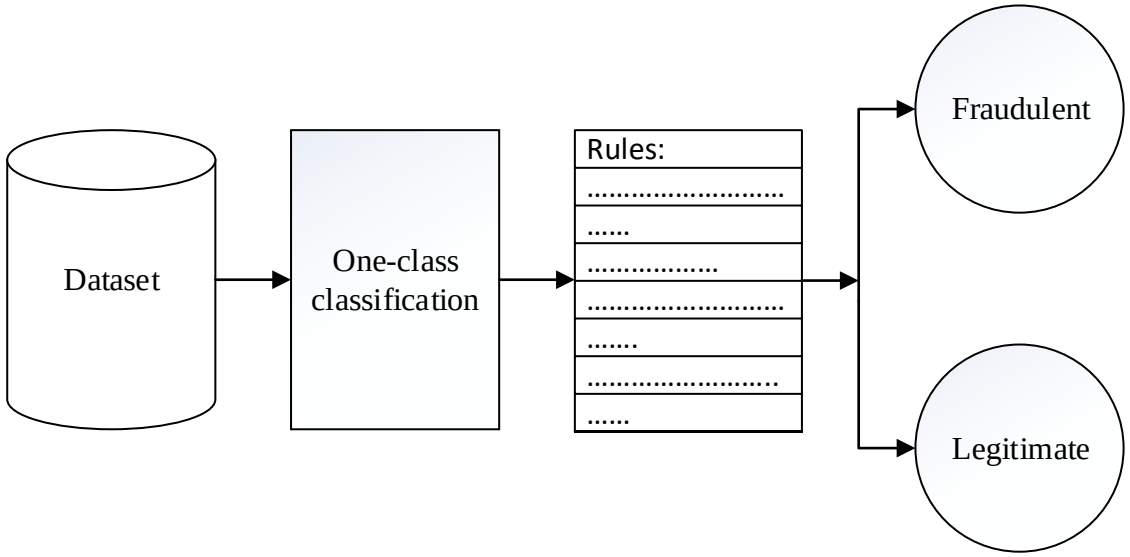


Figure 13. Architecture of the hybrid model [14]

If an account score exceeds a prescribed threshold than it goes through second level of refinement as shown in *Figure 13*. If any account contravenes any of the rules in the second level, it gets flagged as suspected to be fraudulent. Those flagged accounts can then be investigated further by the responsible authorities.

For testing purposes, three separate tests were made on a dataset consisting 10,000 transactions where 1555 were fraudulent. First test was done using the rule-based model, second one was done using the one class classification and the third one was done using the hybrid model as presented in this paper.

Table 7. Test results of the hybrid model [14]

	Rule model	Classification model	Missed by the RM	Hybrid model
Fraud detection rate (%)	92	54	5	97
False positive rate (%)	16	3	-	9,5

As seen in *Table 7*, the fraud detection rate of the rule model is higher than that of the classification model. In the other hand, the false positive rate of the classification model is lower than that of the rule model. Additionally, the last one managed to detect a number of fraud accounts that were missed by the rule model. The result of the combination of these models is an 97% fraud detection rate and a false positive rate of 9,5%.

The advantages of a hybrid model are also exposed in [15]. The authors of this paper presented a model that combines supervised methodologies like decision trees with unsupervised ones like SVM-s in order to prosper a new approach towards credit card fraud detection (*Figure 14*).

As a first step, each account in this system is classified using three different decision tree algorithms: ID3, C5.0 and CHAID. Starting from the root node of the decision tree, accounts are split into binary-split child nodes using an input attribute which separates them best. As the tree is grown, the resultant tree may over fit the training data, containing possible errors

or noise. That is the why this system continuously checks whether removal of some nodes, starting from the leaf ones, make a significant effect on the tree's classification performance. This operation is called as pruning.

After the tree is fully constructed, a new observation is done by SVM. Unlike the decision tree methods, SVM tries to find a hyperplane to separate two classes while minimizing the classification error. SVM's basic idea is eliminate previous classification errors made by the decision tree by transforming the attributes to a higher dimensional feature space and finding the optimal hyperplane in that space that maximizes the margin between the classes. This is achieved using the SVM kernels such as: polynomial kernel, sigmoid kernel, radial basis kernel and linear kernel. As a result, each record is denoted as either fraudulent or legal.

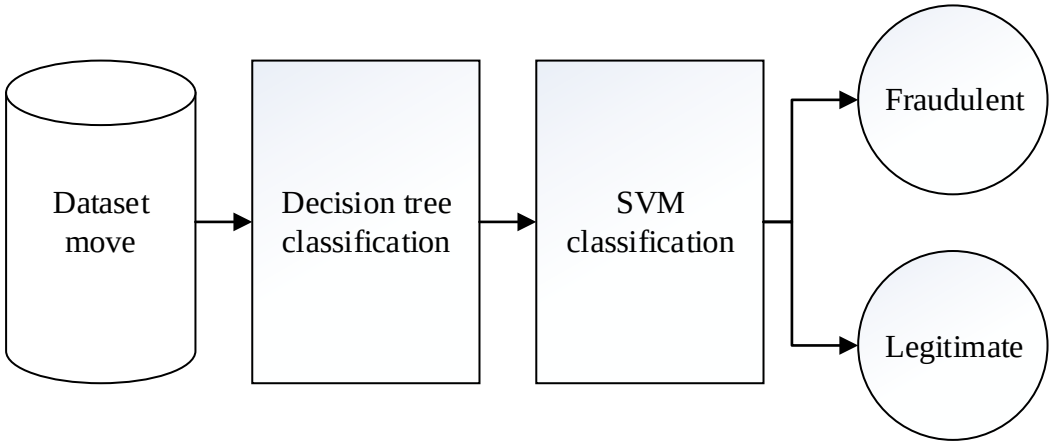


Figure 14. Architecture of the hybrid system [15]

To evaluate the accuracy of this system, three test were made on different datasets. Each dataset had a different fraud to legal account ratio as seen in *Table 8*.

Table 8. Test results of the hybrid model [15]

Classifier model		Ratio Fraud: Legal					
		1:1		1:4		1:9	
		Train	Test	Train	Test	Train	Test
DT methods	ID3	90.01	86,79	92,13	92,53	93,69	94,69
	C5.0	92.71	91,08	97,44	92,81	99,15	94,52
	CHAID	92,51	89,37	92,92	92,53	94,32	94,76
SVM methods	RBF	99,78	83,02	98,00	88,97	98,75	93,08
	Polynomial	99,78	83,02	98,00	88,97	98,75	93,08
	Sigmoid	99,78	83,02	98,00	88,97	98,75	93,08
	Linear	99,78	83,02	98,00	89,19	98,75	93,08

From these results we can conclude the beneficial of combining decision trees with support vector machines. While decision trees perform better in test environment, SVM methods are superior in training environment. Resulting in a better overall train and test and a fraud detection rate of nearly 97%.

Both of these approaches in [14] and [15] present ingenious systems that by using the combined forces of individual methodologies, detect fraud at the highest rate possible while maintaining a decently low false positive rate. Undoubtedly, these ensemble combinations are the way to go for any fraud detection system. Although the mentioned approaches proudly presented a 97% fraud detection rate, they lack the instant reaction technology.

To be able to detect fraudulent activity in the moment of occurrence, a model should analyse on transaction level instead of analysing on account level. Analysing on transaction basis is not only beneficial for the financial institution but also for the costumer, based on the fact that the sooner the fraudulent activity is detected the lower the losses will be.

2.4 Instance-incremental classifiers for fraud detection

Instance-incremental methods are truly incremental in the sense that they learn from each training example as it arrives. Thus, they can essentially learn indefinitely. This category includes lazy learners and incremental learners such as Naive Bayes Updateable and Hoeffding Trees.

Hoeffding tree is a decision tree algorithm for streaming data, where instead of reusing instances, it waits for new instances to arrive. The most interesting feature of the Hoeffding Tree is that it builds a tree that probably converges to the tree built by a batch learner with sufficiently large data. The pseudocode of this algorithm is shown in *Figure 15.* As inputs, this classifier takes the data stream and the confidence parameter δ. From the root node until the tree is complete, this algorithm grows the tree by splitting leafs until each example is in its class. This is done using the Hoeffding bound:

$$G(best\ attribute) - G(second\ best) > \sqrt{\frac{R^2 ln1/\delta}{2n}} \tag{17}$$

which takes the value:

$$\epsilon = \sqrt{\frac{R^2 ln1/\delta}{2n}}, \tag{18}$$

as a confidence interval for the estimation of the entropy at a node, where R is the range of the random variable, δ is the desired probability of the estimate not being within ϵ of its expected value, and n is the number of examples collected at the node. In the case of information gain, the entropy is in the range [0,…,log n_c] for n_c class values. If the value in (18) is smaller than the difference between the split gain G of the best and second best attribute $G(best\ attribute) - G(second\ best)$ at a particular node , the algorithm splits on the best one.

sort (x, y) to leaf l using HT

update counts n_{ijk} at leaf l

if *examples seen so far at l are not all of the same class* ***then***

compute G for each attribute

if $G(best\ attribute) - G(second\ best) > \sqrt{\frac{R^2 ln 1/\delta}{2n}}$ ***then***

split leaf on best attribute

for *each ranch* ***do***

start new leaf and initialize counts

return HT

Figure 15. Hoefdding Tree pseudocode [16]

Although the output of this tree is asymptotically nearly identical to that of a no incremental learner using infinitely many examples, it grows slow which may affect the performance.

K-Nearest Neighbours(KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used for batch and instance learning. It assumes that similar instances exist in close proximity, and it groups them by calculating the distance usually done with the Euclidean distance (as in WEKA). The pseudocode for this algorithm is shown in *Figure 16.* It sorts the collection of distances in the ascending order and returns the mode of the corresponding K labels.

Calculate the distance of examples (x,y)

Add the distance and the index of the example.

if *examples seen so far are not on the same class* ***then***

Sort the ordered collection of distances and indices

Pick the first K entries from the sorted collection

Get the labels of the selected K entries

return the mode of the K labels

Figure 16. KNN pseudocode [16]

The number of neighbours (k) depends upon the data; generally, larger values of *k* reduce the effect of noise on the classification, but make boundaries between classes less distinct [16]. Although this algorithm is simple and easy to implement it gets significantly slower as the number of examples or independent variables increase.

Naive Bayes Updateable is a classification algorithm known for its low computational cost and simplicity. As an incremental algorithm, it is well suited for the data stream setting. It is based on Bayes' theorem which tells how the probability of an event is modified after accounting for evidence:

$$\Pr(c|d) = \frac{\Pr(c)\Pr(d/c)}{\Pr(d)} \tag{19}$$

where *Pr(c)* is the prior, the initial probability of event *c*, *Pr(c|d)* is the posterior, the probability after accounting for *d*, *Pr(d|c)* is the likelihood of event *d* given that event *c* occurs, and *Pr(d)* is the probability of event *d*. It is based on the definition of conditional probability, by which $Pr(c \cap d) = Pr(c)Pr(d|c) = Pr(d)Pr(c|d)$. The Naive Bayes model is built as follows: Let $x_1, \ldots, x_k$ be k discrete attributes, and assume that x_i can take n_i different values. Let C be the class attribute, which can take n_C different values. Upon receiving an unlabelled instance $I = (x_1 = v_1, \ldots, x_k = v_k)$, the Naive Bayes classifier computes a "probability" of *I* being in class c as:

$$\Pr(C = c|I) \cong \Pr(C = c) \prod_{i=1}^{k} \Pr(x_i = v_i|C = c) \tag{20}$$

$$= \Pr(C = c) \prod_{i=1}^{k} \frac{\Pr(x_i = v_i \wedge C = c)}{\Pr(C = c)} \tag{21}$$

The values $Pr(x_i = v_j \wedge C = c)$ and $Pr(C = c)$ are estimated from the training data. Thus, the summary of the training data is simply a 3-dimensional table that stores for each triple (x_i, v_j, c) a count $n_{i,j,c}$ of training instances with $x_i = v_j$ and class *c*, together with a 1-dimensional table

for the counts of $C = c$. This algorithm is naturally incremental: upon receiving a new example (or a batch of new examples), simply increment the relevant counts [16]. Predictions can be made at any time from the current counts.

Analysing these approaches and all preciously mentioned papers, created a clear overview of what the problem persists. Literature review helped in identifying the gap which actually exists in these systems. Therefore, the following chapter will present the problem definition together with the aim and the objectives to tackle.

3 PROBLEM DECLARATION

Most of the approaches in fraud detection, including all of the elaborated papers in [4, 7-15], detect fraud by learning from batches of the dataset. This particular type of operation is called *batch learning*. In this setting, a learning method is trained every *w* new instances form the batch, and when that batch is complete, it is given to a classifier to train on. The main disadvantages of these methods are that they:

- require a parameter w specifying the batch-size;
- are forced to delete trained models to make room for new ones;
- cannot learn from the most recent examples until a new batch is full;

The dependence on the batch, limits the ability of the system to react instantly. Additionally, having to delete trained models may affect these methods' ability to learn the complete concept and not being able to learn from new examples immediately may affect their ability to respond to a new concept.

3.1 Aim and Objectives

This thesis aims to present a system that detects fraud in a real-time manner, with the lowest potential cost. Its advantages against other analysed approaches in section 2 will present an attractive point for the market and arouse further research in the fraud detection domain. This thesis will propose a model using *instance-incremental learning*, from which, the best performing learner will be chosen. Main objectives of this thesis are as follows:

O1: Applying a comprehensive research on existing approaches
O2: Using the aggregated knowledge to construct the new proposed model
O3: Evaluating the achievements on a practical experiment
O4: Comparing this systems ability with other similar approaches

4 METHODOLOGY

The presented model is built purposely on a synthetic dataset, keeping confidentiality intact. This thesis uses a dataset [17] as secondary data for simulation purposes. This dataset contains mobile transactions based on a sample of real transactions extracted from one month of financial logs from a mobile transaction service. The original logs were provided by a multinational company, who is the provider of the mobile financial service which is currently running in more than 14 countries all around the world. Additionally, in order to evaluate the selected dataset and also the proposed model, a primary data collection was conducted using interview methods. The interviews were organised with domain experts from local banks. Initially, this dataset was pre-processed with a number of filters and scaled down to a smaller and more balanced version from the initial one. Pre-processing the data helped in creating a better understanding for the upcoming analysis and structuring of the model, that was done using WEKA software. WEKA is an open source machine learning software that can be accessed through a graphical user interface, standard terminal applications, or a Java API. It is widely used for teaching, research, and industrial applications, contains a plethora of built-in tools for standard machine learning tasks, and additionally gives transparent access to well-known toolboxes such as scikit-learn, R, and Deeplearning4j. Additionally, WEKA offers a number of incremental classifiers including: *Hoefding tree*, *Knn lazy learner* and *Naïve Bayes Updateable*, all of them described in section 2. This model is build using the last mentioned classifier while *Hoefding tree* and *KNN* were also separately tested in order to compare the results. *Naïve Bayes Updatable* is the successor of *Naïve Bayes*, that can handle evolving data, which is always the case with any transaction dataset as in [17]. It is efficient and requires less memory usage compared to its predecessor. For better segregation with other approaches, this model will be named as "Active Fraud Detection Model", hereafter mentioned as AFDM. For testing purposes, this model was validated using *k-fold cross-validation*, which is a resampling procedure used to evaluate machine learning models. This procedure has a single parameter called *k* that refers to the number of groups that a given data sample is to be split into. These *k* groups are then recursively used for testing and training. The results of the tests are evaluated using the confusion matrix and cost.

5 EXPERIMENTAL DESIGN

This chapter sheds some light at the new AFDM, which is described in details. It is divided into two sections. The first section presents the pre-processing steps that were applied in the dataset, in order to refine it for classification. The classification and the methodology followed for building the AFDM can be found in the second section. Each action is particularly presented, conducting a transparent communication.

5.1 Data pre-processing

Analysing data that has not been carefully screened for particular problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis. Based on the scale that pre-processing affects the final results, a number of filters were applied in this dataset before proceeding with any classifying duty.
The particular synthetic dataset used in this study, in its raw form, contained an astonishing number of 6362620 instances. Each instance with 10 attributes including:

- “step” - maps a unit of time in the real world. In this case 1 step is 1 hour of time.
- “type” - cash-in, cash-out, debit, payment and transfer.
- “amount” - amount of the transaction in local currency.
- ”nameOrig” - customer who started the transaction.
- ”oldbalanceOrg” - initial balance before the transaction.
- ”newbalanceOrig” - new balance after the transaction.
- ”nameDest” - customer who is the recipient of the transaction.
- ”oldbalanceDest” - initial balance recipient before the transaction.
- ”newbalanceDest” - new balance recipient after the transaction.
- ”isFraud” - This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behaviour of the agents aims to

profit by taking control or customers' accounts and try to empty the funds by transferring to another account and then cashing out of the system.

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
1	PAYMENT	9839.64	C1231006815	170136	160296.36	M1979787155	0	0	0
1	PAYMENT	1864.28	C1666544295	21249	19384.72	M2044282225	0	0	0
1	PAYMENT	11668.14	C2048537720	41554	29885.86	M1230701703	0	0	0
1	PAYMENT	7817.71	C90045638	53860	46042.29	M573487274	0	0	0

Figure 17. Raw Dataset

From all these records, only 8213 were fraudulent. This difference between the two classes created an unbalanced dataset to be classified, which is why under-sampling was necessary. In order to do so, a new dataset was sampled containing records with the "step" smaller than 48. Which resulted in a dataset of 1660 records, where 580 were fraudulent.

Additionally, the last column of the dataset, also called as the class, was transformed from binary values: 0 and 1, into text fields: "Legal" and "Fraud". This was mainly done per weka requirements.

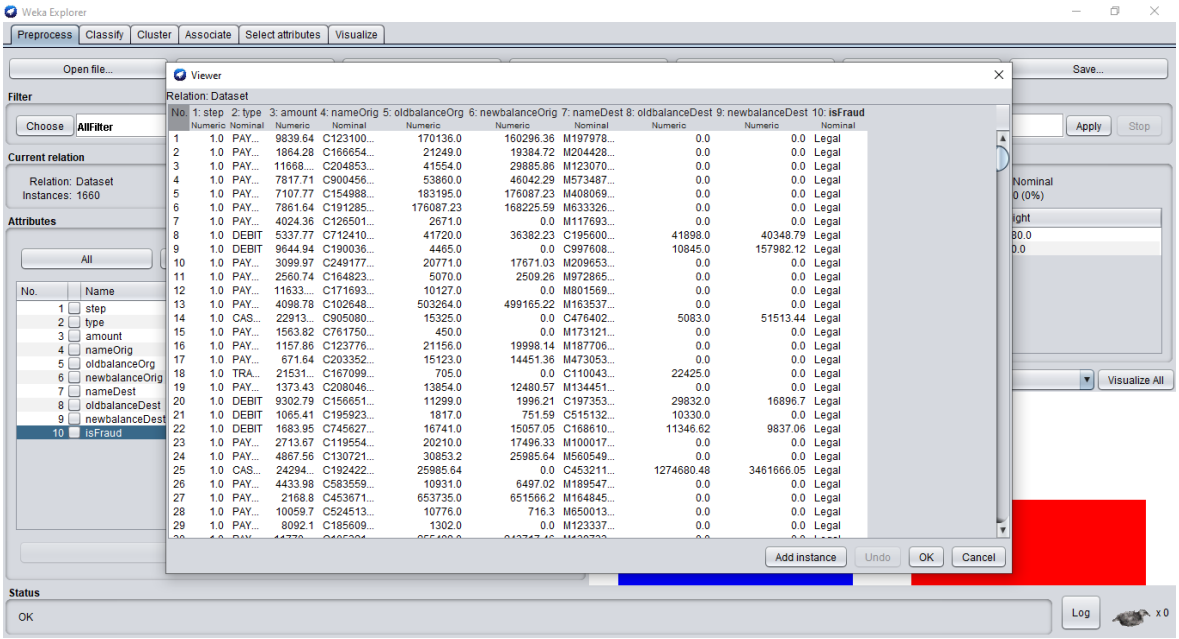


Figure 18. Dataset in weka

As a first step, after adapting the dataset to be accessible from weka, the missing values were corrected by replacing them with the corresponding mean of that attribute(column), using the unsupervised attribute filter “ReplaceMissingValues” as seen in *Figure 19*.

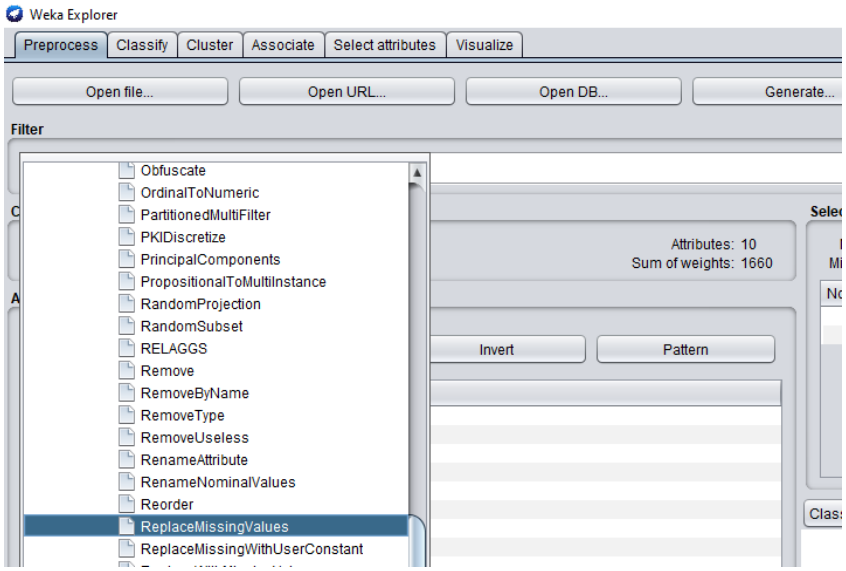


Figure 19. Filter for replacing missing values

Secondly, all the numeric attributes were normalized. Normalization is the process of scaling the values of an attribute to a range [0,1] while maintaining the relative distance between them. This is mainly done in order to aid the classifying algorithm to better perceive the correlation between attributes.

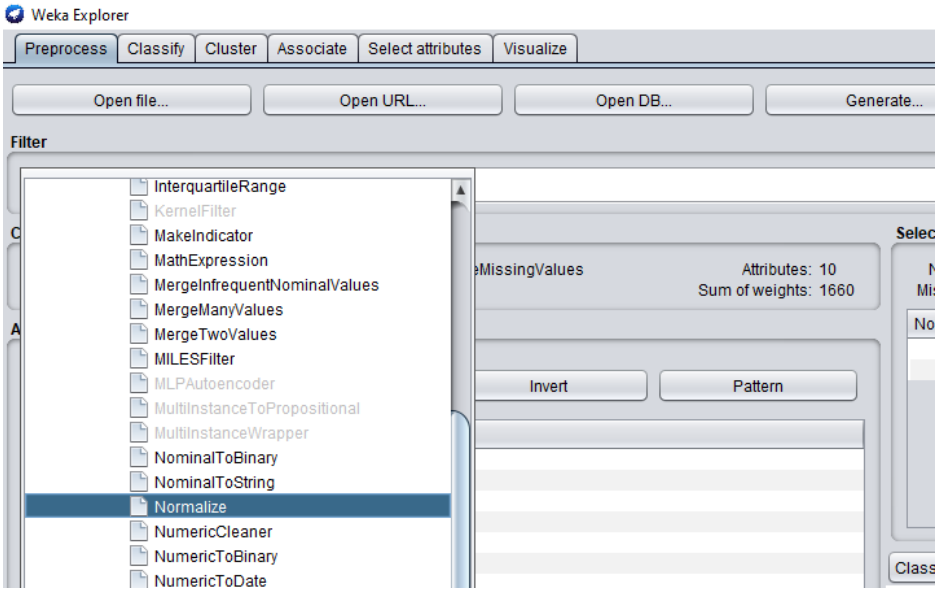


Figure 20. Filter for normalizing attribute values

As a final step, "nameOrig" attribute was removed from the dataset. This attribute was concluded as irrelevant because it corresponds to the name of the issuing account which doesn't lead to the final decision. As a result of these steps the dataset evolved in the form seen below:

Viewer

Relation: Dataset-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filters.unsupervised.attribute.Normalize-S1.0-T0.0-w

No.	1: step	2: type	3: amount	4: oldbalanceOrg	5: newbalanceOrig	6: nameDest	7: oldbalanceDest	8: newbalanceDest	9: isFraud
	Numeric	Nominal	Numeric	Numeric	Numeric	Nominal	Numeric	Numeric	Nominal
1	0.0	PAY...	9.8309...	0.0085547841...	0.01605004112...	M197978...	0.0	0.0	Legal
2	0.0	PAY...	1.8555...	0.0010684429...	0.00194093960...	M204428...	0.0	0.0	Legal
3	0.0	PAY...	0.0011...	0.0020894196...	0.00299239035...	M123070...	0.0	0.0	Legal
4	0.0	PAY...	7.8089...	0.0027081903...	0.00461009001...	M573487...	0.0	0.0	Legal
5	0.0	PAY...	7.0990...	0.0092114172...	0.01763113824...	M408069...	0.0	0.0	Legal
6	0.0	PAY...	7.8529...	0.0088540241...	0.01684397348...	M633326...	0.0	0.0	Legal
7	0.0	PAY...	4.0156...	1.3430331359...	0.0	M117693...	0.0	0.0	Legal
8	0.0	DEBIT	5.3290...	0.0020977664...	0.00364285432...	C195600...	0.0012711132...	0.00116461185...	Legal
9	0.0	DEBIT	9.6362...	2.2450928312...	0.0	C997608...	3.2901864092...	0.00455993476...	Legal
10	0.0	PAY...	3.0912...	0.0010444081...	0.00176935245...	M209653...	0.0	0.0	Legal
11	0.0	PAY...	2.5520...	2.5492991386...	2.51245419356...	M972865...	0.0	0.0	Legal
12	0.0	PAY...	0.0011...	5.0920616129...	0.0	M801569...	0.0	0.0	Legal
13	0.0	PAY...	4.0900...	0.0253051377...	0.04998006385...	M163537...	0.0	0.0	Legal
14	0.0	CAS...	0.0229...	7.7057217554...	0.0	C476402...	1.5420947457...	0.00148686399...	Legal
15	0.0	PAY...	1.5550...	2.2626915432...	0.0	M173121...	0.0	0.0	Legal
16	0.0	PAY...	1.1491...	0.0010637667...	0.00200235968...	M187706...	0.0	0.0	Legal
17	0.0	PAY...	6.6291...	7.6041520462...	0.00144697560...	M473053...	0.0	0.0	Legal
18	0.0	TRA...	0.0215...	3.5448834176...	0.0	C110043...	6.8033591726...	0.0	Legal
19	0.0	PAY...	1.3647...	6.9660730310...	0.00124964572...	M134451...	0.0	0.0	Legal
20	0.0	DEBIT	9.2940...	5.6813670548...	1.99875110021...	C197353...	9.0505155334...	4.87699809288...	Legal
21	0.0	DEBIT	1.0566...	9.1362456311...	7.52546745789...	C515132...	3.1339442699...	0.0	Legal
22	0.0	DEBIT	1.6752...	8.4177153610...	0.00150762170...	C168610...	3.4423692867...	2.83933092613...	Legal
23	0.0	PAY...	2.7049...	0.0010161999...	0.00175186021...	M100017...	0.0	0.0	Legal
24	0.0	PAY...	4.8588...	0.0015513616...	0.00260187187...	M560549...	0.0	0.0	Legal
25	0.0	CAS...	0.0242...	0.0013066108...	0.0	C453211...	0.0386716126...	0.09991618910...	Legal
26	0.0	PAY...	4.4252...	5.4963291686...	6.50529046199...	M189547...	0.0	0.0	Legal
27	0.0	PAY...	2.1600...	0.0328711256...	0.06523956192...	M164845...	0.0	0.0	Legal
28	0.0	PAY...	0.0010...	5.4183920154...	7.17211822947...	M650013...	0.0	0.0	Legal
29	0.0	PAY...	8.0833...	6.5467208650...	0.0	M123337...	0.0	0.0	Legal

Figure 21. Final Dataset

5.2 Classifying

Naïve Bayes Updateable was chosen as the most appropriate classifier for the purpose of this study: the evaluation of which can be found in *Chapter 7*. In order to achieve the best results, along with this classifier, the bagging technique and "CVParameterSelection" filter was used. Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms. It basically generates a number of new training sets in a number of iterations and outputs the aggregated average which also reduces variance and helps to avoid overfitting.

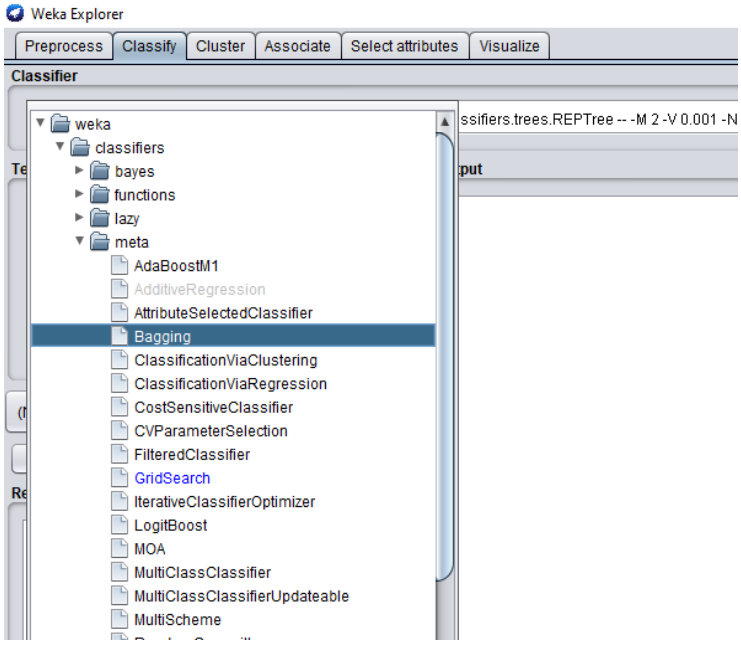


Figure 22. Bagging-meta classifier

In the customization panel of this meta classifier (*Figure 23*), all the parameters were left on default, except for the classifier field which was changed to our desired Bayes algorithm.

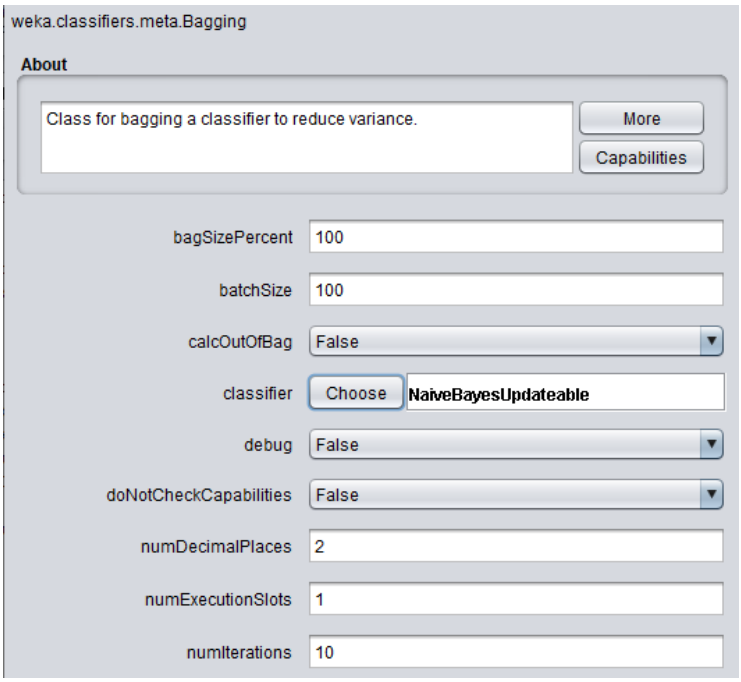


Figure 23. Customization panel of Bagging

Before applying this algorithm, its parameters were set using the "CVParameterSelection" filter, found under the meta category in weka:

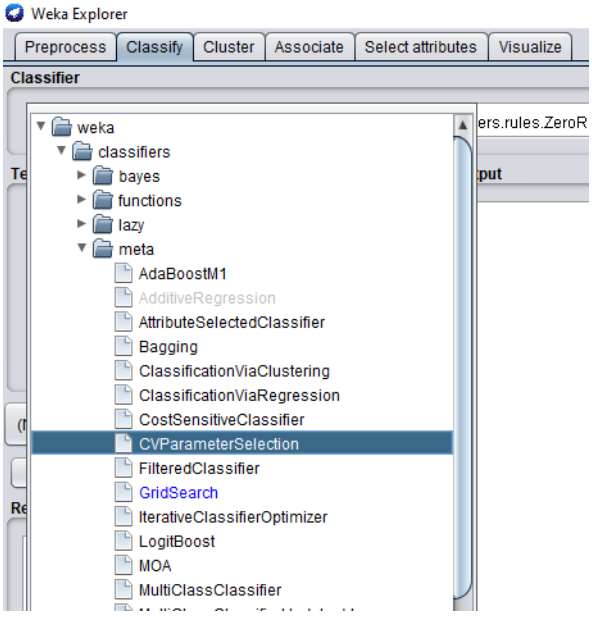


Figure 24. CVParameterSelection

In the customization panel of this classifier (*Figure 25*), using the java.lang.String field, the batch size of Naïve Bayes was set to 1. Setting the batch size to 1, ensures the algorithm to work instance by instance. Other parameters of Bayes are mostly cosmetic inputs, expect for the Kernel Estimator which was left in default state (False), because default distribution performed a better spread (*Figure 26*).

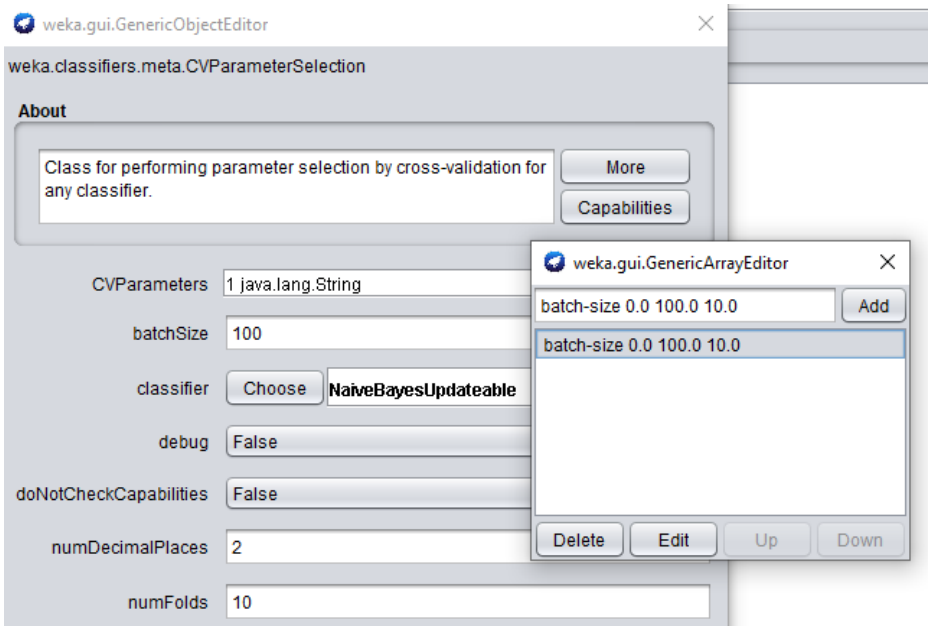


Figure 25. Customization panel of CVParameterSelection

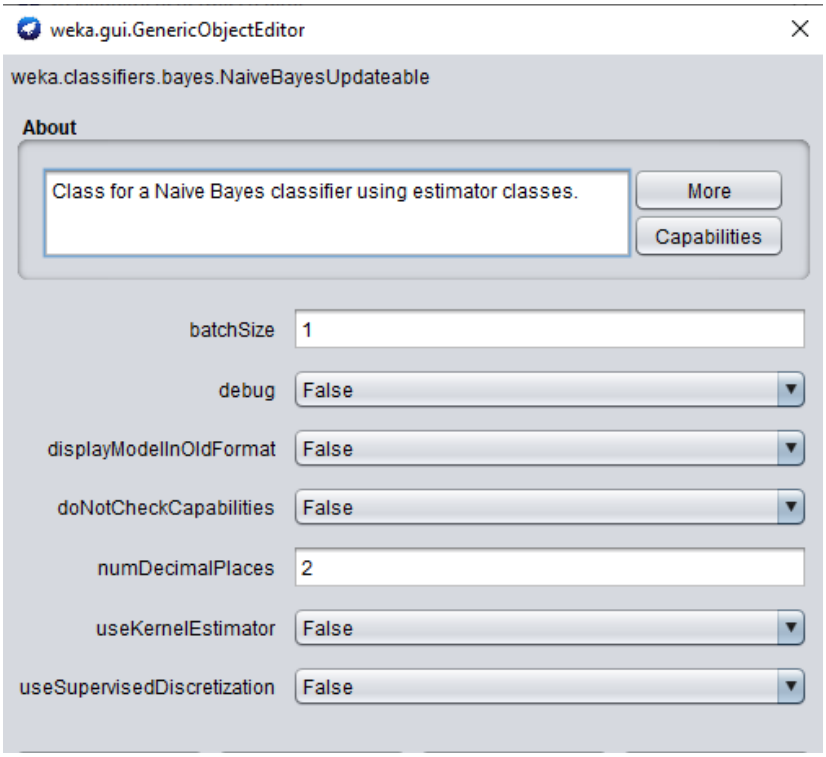


Figure 26. Customization panel of Naive Bayes Updateable

With the parameter selection as the final step, the model was ready to be applied in the dataset with its final form as:

weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.meta.CVParameterSelection -- -P "batch-size 1.0 100.0 10.0" -X 10 -S 1 -W weka.classifiers.bayes.NaiveBayesUpdateable -- -batch-size 1

6 ANALYSIS AND RESULTS OF AFDM

AFDM not only classifies instances in the moment of occurrence, it also uses each instance to immediately train the learner. Any time a new transaction is made, this model classifies it either as fraud or as legal. Afterwards it uses that knowledge and updates Naïve Bayes in order to make it effective for similar upcoming instances. Each step is enhanced using the bagging technique as shown in *Figure 27*.

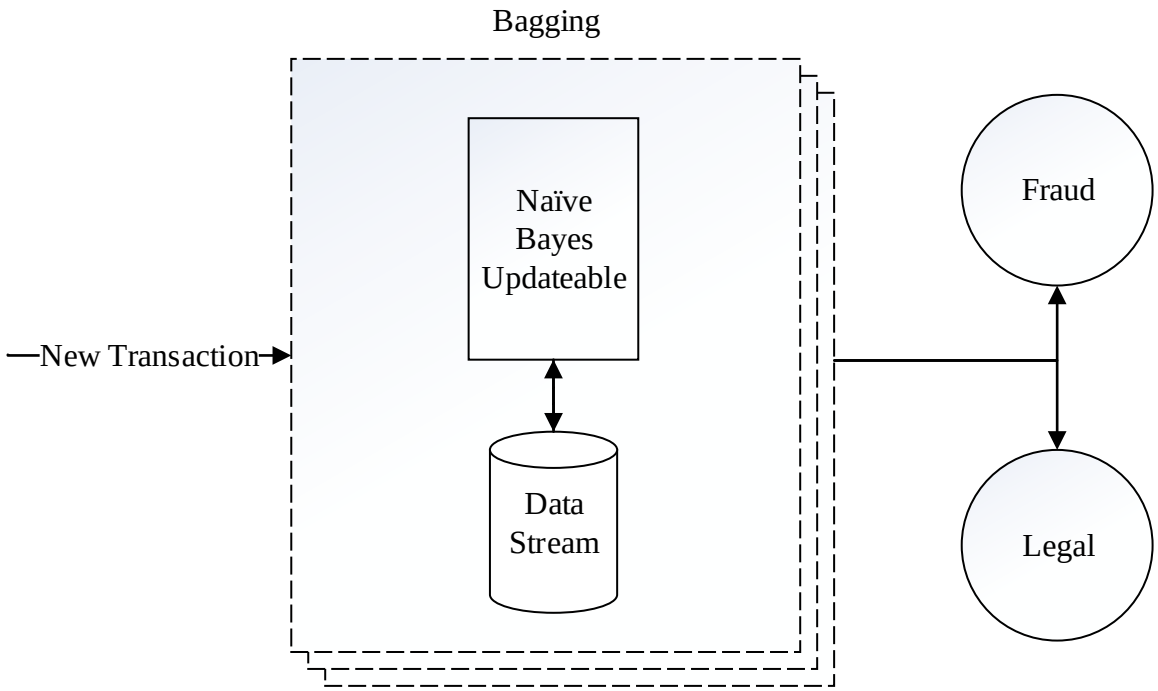
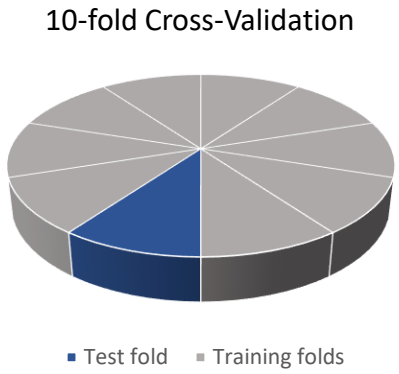


Figure 27. Architecture of AFDM

As mentioned in *Chapter 4*, for testing purposes the 10-fold cross-validation option was used. This type of validation separates the dataset in 10 equal folds. During this validation each fold must act, at least once, as a test example with the rest of folds as training examples. This requires the system to run 10 times. Additionally, in WEKA the model is run for the 11th time in order to output the aggregated result.



Using cross-validation ensured a better model estimate that had a lower bias than other options. Also, using this option prevented overfitting on the dataset and provided a general metric. For performance evaluation, all of the standard measures were observed including:

- **The Confusion matrix,** which is a table that summarizes the number of correctly and incorrectly classified instances among classes.
- **The Kappa statistic *κ*,** which is a sensitive measure for quantifying the predictive performance of the classifier.
- **The Precision,** which is the fraction of relevant instances among the retrieved instances.
- **The Recall,** which is the fraction of the total amount of relevant instances that were actually retrieved
- **The RMSE**, which is a measure used for calculating the differences between values predicted by the model and the values observed.

Each of these measures take values from 0 to 1. Higher values of these indicators mean better performance of the model except for the RMSE, which apparently is an error estimate.
In order to unveil the superiority of the AFDM model, an initial test was made using the baseline classifier: "ZeroR". This classifier trivially predicts the most-frequent class and gives the baseline accuracy.

```
Time taken to build model: 0 seconds

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

Correctly Classified Instances        1080               65.0602 %
Incorrectly Classified Instances       580               34.9398 %
Kappa statistic                          0
Mean absolute error                      0.4547
Root mean squared error                  0.4768
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances             1660

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000     1.000     0.651       1.000   0.788      ?        0.500     0.651     Legal
                 0.000     0.000     ?           0.000   ?          ?        0.500     0.349     Fraud
Weighted Avg.    0.651     0.651     ?           0.651   ?          ?        0.500     0.545

=== Confusion Matrix ===

    a    b   <-- classified as
 1080    0 |    a = Legal
  580    0 |    b = Fraud
```

Figure 28. ZeroR Model Output

Running this classifier on the dataset exhibited a 65% accuracy (*Figure 28*), by classifying all of the instances in the legal class (prior class). Therefore, all of the fraud instances were misclassified. Other indicators like *kappa statistic* (0) and *RMSE* (1) reached extreme values, showing the ineffectiveness of using this classifier.

After determining the results of this baseline classifier, we used its predictions in order to compare it with the presented model in this thesis. Running the AFDM model in these conditions proudly presented a much higher accuracy, reaching 97% (*Figure 28*).

```
Time taken to build model: 8.89 seconds

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

Correctly Classified Instances        1613               97.1687 %
Incorrectly Classified Instances        47                2.8313 %
Kappa statistic                          0.9387
Mean absolute error                      0.1685
Root mean squared error                  0.244
Relative absolute error                 37.0625 %
Root relative squared error             51.1679 %
Total Number of Instances             1660

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.960    0.007    0.996      0.960    0.978      0.940    0.987     0.994     Legal
                 0.993    0.040    0.931      0.993    0.961      0.940    0.987     0.957     Fraud
Weighted Avg.    0.972    0.018    0.973      0.972    0.972      0.940    0.987     0.981

=== Confusion Matrix ===

    a    b   <-- classified as
 1037   43 |    a = Legal
    4  576 |    b = Fraud
```

Figure 29. AFDM Output

As seen from the confusion matrix (*Table 9*), this model correctly classified 1613 instances, leaving 47 misclassified. From these 47 misclassified instances, only 4 were fraudulent.

Table 9. Confusion Matrix of AFDM

Confusion Matrix		
Classified as	**a**	**b**
a=Legal	1037	43
b=Fraud	4	476

Besides accuracy, this model puts much better numbers in other measures as well, compared with the baseline values.

Table 10. Summary of test results (Baseline accuracy)

Model **Measure**	*ZeroR*	*AFDM*
Accuracy	65	97.2
Kappa Statistic	0	0.938
Precision	0.651	0.973
Recall	0.651	0.972
RMSE	1	0.244

As seen from *Table 10*, kappa statistic, precision and recall of the AFDM model maintain "close to 1" values, while the RMSE value is relatively small. These values are undoubtedly attractive and speak for the effectiveness of this system in resolving the actual issues.

6.1 Evaluation of the dataset and AFDM from Domain Experts

In order to gather additional information on operation strategies and requirements of local fraud mechanisms, several interviews were conducted. The interviewees are employees of TEB SH.A. Their contributions in different sectors of the bank, including the fraud sector, make them valuable assets for the purpose of these interviews. According to the Head and the Manager of the Fraud Unit, fraudsters usually attack during weak timeframes such as: holidays, weekends, after working hours etc. This is the reason why a fraud detection model should be always active, which is how AFDM is built. The Manager of this unit emphasises this point, saying that "24/7 operation of the system is a must". Another compliance was reached when discussing whether a time window is needed for analyses. It was noted that each transaction should be evaluated separately and accurately, excluding the use of a time window. According to the Product Development Manager of this institution, fraud relatable

attributes of a transaction are: amount, beneficiary name, beneficiary account and other beneficiary information. The same answer was received from other interviewees, resulting in removal of an attribute mentioned in *Chapter 5*. This information also helped in evaluating the chosen dataset. All of the three subjects agreed that missing a fraud behaviour is way more harmful than disturbing a good costumer. Besides financial losses, this error also causes reputational damage.

Based on these conclusions, each interviewed subject welcomed the idea behind AFDM, especially the incremental learning approach. This methodology according to the Head and the Manager of the Fraud Unit would be highly effective in reducing cost and lowering false positive rates.

7 EVALUATION OF AFDM

Although comparing this model with a baseline classifier offered a pretty clear overview of what it can accomplish, AFDM was also evaluated against a couple of other batch and incremental classifiers in the same conditions. The flow and the results of this test process are unveiled in this chapter. The much more concurrent results, urged the need to use an additional performance measure, called cost. Cost is a value calculated from the sum of misclassified instances in both classes, multiplied by their relative weight.

$$\text{Cost} = \text{FP} * \text{weight} + \text{FN} * \text{weight} \tag{22}$$

Where, FP is the number of legal instances classified as fraud and FN is the number of fraud instances classified as legal. Consequences for classifying a fraud behaviour as legal are much worse than the opposite, which is why the weight for this misclassification has been set to 5. This was done using the cost-sensitive evaluation matrix in the classifier evaluation options.

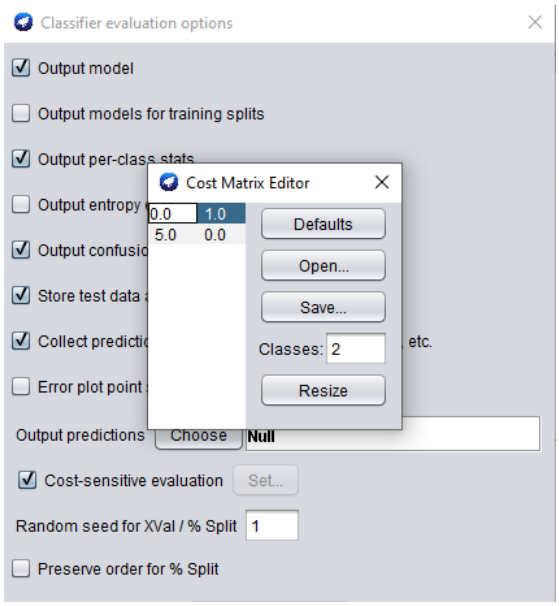


Figure 30. Cost Matrix Editor

With the cost matrix set, the test was initiated including classifiers as: J48, Logistic Regression, Hoeffding tree and KNN lazy learner. J48 and Logistic regression represented the batch category, while Hoeffding tree and KNN represented the incremental one.

Table 11. Summary of test results (Batch and Incremental)

Model **Measure**	*AFDM*	*J48*	*KNN*	*Logistic*	*Hoeffding tree*
Accuracy	97.2	97.1	93.6	75.9	64.8
Kappa statistic	0.938	0.935	0.857	0.384	0.007
Precision	0.973	0.971	0.936	0.794	0.578
Recall	0.972	0.971	0.936	0.760	0.649
RMSE	0.224	0.166	0.252	0.486	0.458
Cost	63	184	398	1903	2859

Table 11 expresses the tests results using the mentioned classifiers. Although, AFD model performed better than other classifiers, J48 presented abutting results. Actually, RMSE value of J48 is lower than that of the presented model, but it's the opposite with the cost measure. This is mainly because J48 did a better classifying on the legal class but missed a lot of fraud instances, which affect the cost more, based on their higher weight. KNN also managed to output a pleasing accuracy, while Logistic and Hoeffding Tree conveyed a poor performance compared to the AFDM. Putting AFDM aside, batch learners seem to do a better classifying compared with incremental ones, lacking the real-time prediction.

The learning methodology of incremental classifiers allows another evaluation measure to be performed, which is *knowledge flow*. This flow presents performance changes of a system with the increase of instances. In weka, this was done using the Knowledge Flow Environment.

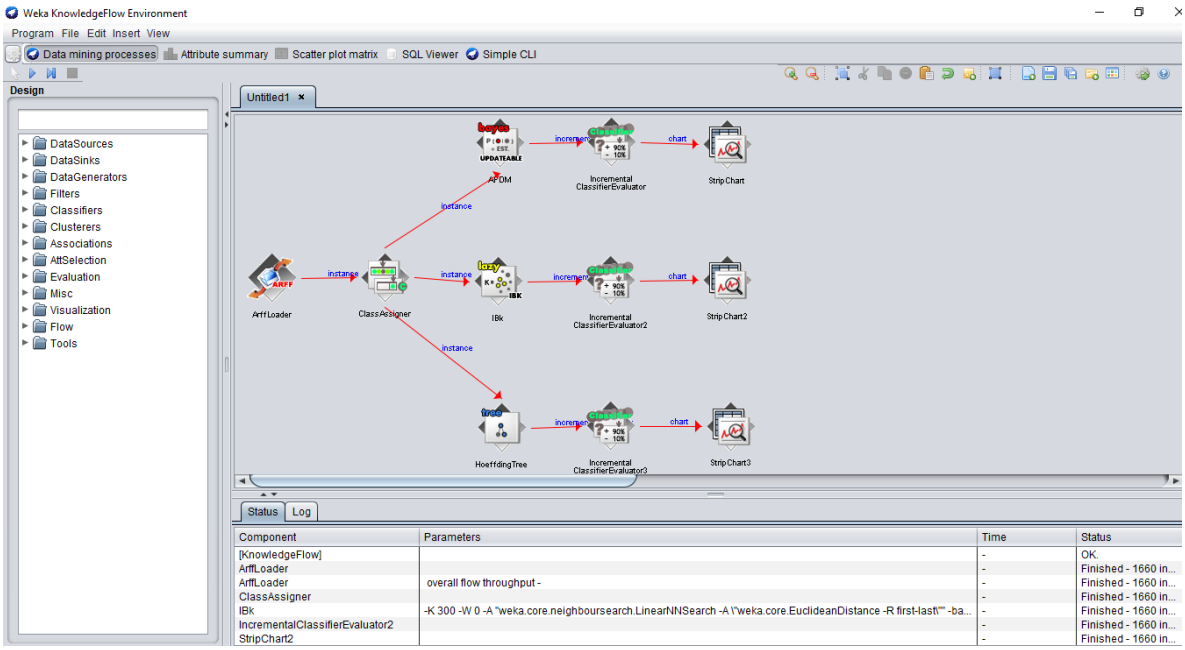


Figure 31. Knowledge Flow Environment

In this panel, the dataset was loaded using the "ArffLoader" and the class was differentiated using the "ClassAssigner". Afterwards, incremental classifiers from *Table 11* were applied and evaluated on charts using the "IncrementalClassifierEvaluator" as seen below:

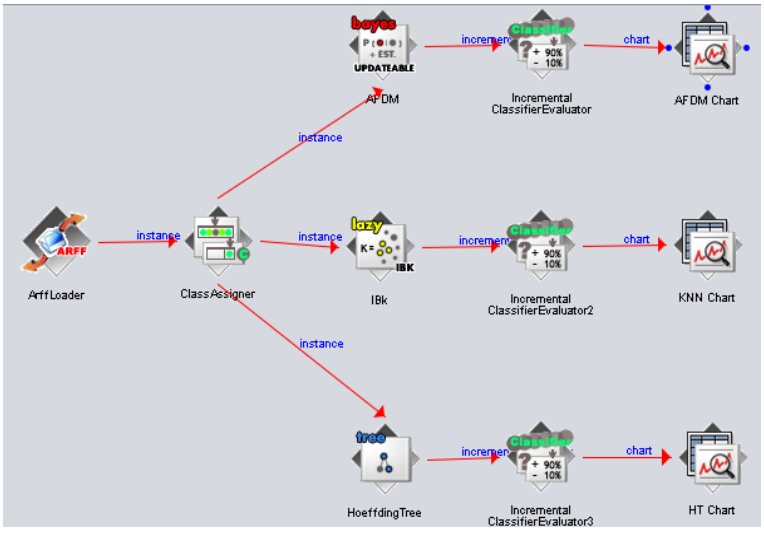


Figure 32. Knowledge Flow design

In *Figure 33* it is shown that accuracy and kappa measures of AFDM present much higher values throughout the increasing number of instances (axis x), compared with those of KNN or Hoeffding tree's. AFDM's accuracy is following a trend of increase, reaching almost 1 at the last instance, while other classifiers presented in *Figure 34-35* are decreasing their accuracy towards the last instance. RMSE is also much lower in the first chart compared with other charts, representing KNN and Hoefdding tree.

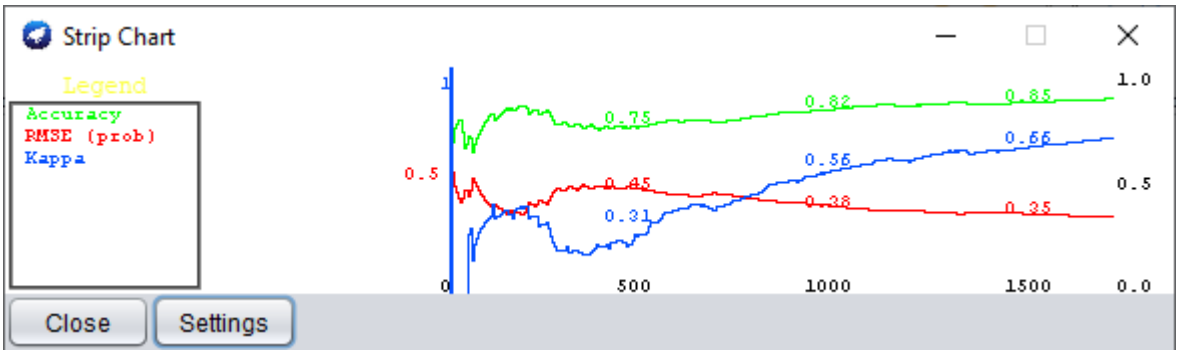


Figure 33. AFDM chart

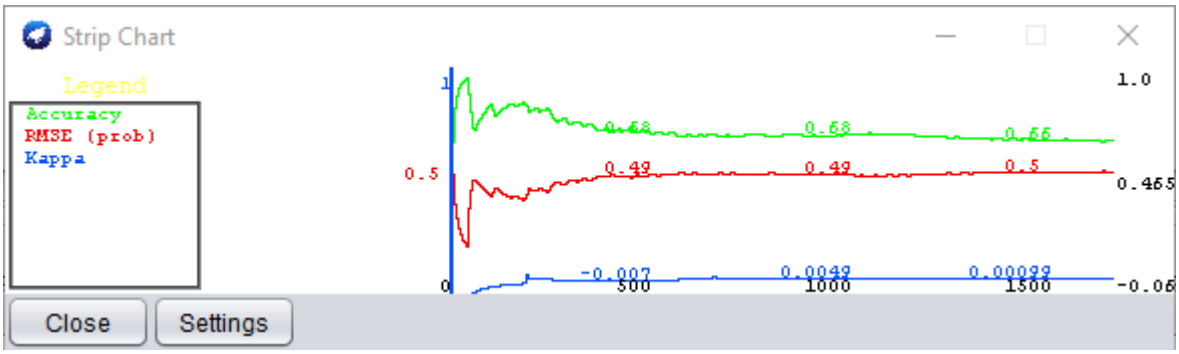


Figure 34. KNN chart

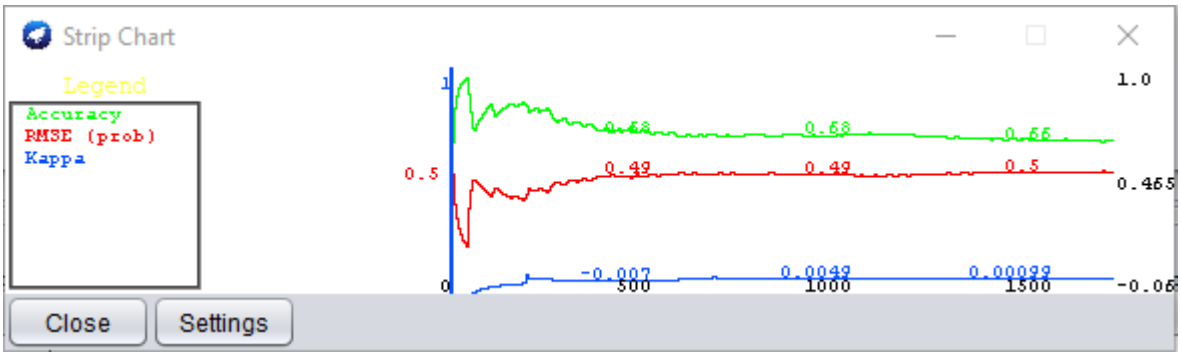


Figure 35. Hoeffding Tree chart

Observing these charts, presented in *Figure 33-35*, proved that AFDM not only produces a better output, it also maintains a much more consistent performance throughout the increasing number of instances.

8 CONCLUSSION AND FUTURE WORK

This thesis introduced a novel approach for real-time fraud detection in online banking transactions using incremental learning approach. AFDM is a supervised fraud detection model, built to aid the fraud unit in a financial institution. Its findings contributed to domain experts achieve a better understanding of the potential risk they are exposed to. The development of this model was made possible by gratifying the objectives precisely. Initially, the most successful fraud detection methodologies were studied. The disadvantages that surfaced from these models built problems that needed to be solved by the proposed AFDM. Unlike other approaches, this model is transaction-based. Analysing transactions instead of accounts not only allowed a more detailed dissection, it also served a better environment for post detection actions. Additionally, AFDM accounted all of the user transactions which is a must. This excluded the need of a time window and reduced the risk of missing a fraudulent behaviour. On top of that, a classifying algorithm like *Naïve Bayes Updateable* incremented its knowledge transaction by transaction. This learning methodology provided the ability to detect and respond in real time. It also allowed to learn new concepts of behaviour changes immediately.

The effectiveness of this model was evaluated on a dataset modelled from a mobile transaction service. The resemblance with a real dataset offered real-world scenarios and ensured valid results. Comparing these results with classifiers from different categories ultimately proved the significance of this model.

Given the good results and the consistency presented in the previous chapter, AFDM is undoubtedly an attractive pick for the services it offers in the fraud detection domain. Currently this model presents the framework for fraud detection classification. Among the directions for future work it is planned to implement global-based classifying and post detection actions. Global-based classifying will deal with new costumers that have a small number of transactions. On the other hand, post detection actions will enable the interaction with costumers. In case of a suspicious behaviour, the transaction will be blocked until the costumer proves the opposite.

9 REFERENCES

[1] Peterson, M. (2011). A brief history of internet banking.

[2] Alsayed, A., & Bilgrami, A. (2017). E-banking security: Internet hacking, phishing attacks, analysis and prevention of fraudulent activities. *Int. J. Of Emerg. Techn. and Adv. Activ*, *7*(1), 109-115.

[3] Gee, J., & Button, M. (2019). The Financial Cost of Fraud 2019: The Latest Data from Around the World.

[4] Leonard, K. J. (1995). The development of a rule based expert system model for fraud alert in consumer credit. *European journal of operational research*, *80*(2), 350-356.

[5] Chakraborty, C., & Joseph, A. (2017). Machine learning at central banks.

[6] Chitra, K., & Subashini, B. (2013). Data mining techniques and its applications in banking sector. *International Journal of Emerging Technology and Advanced Engineering*, *3*(8), 219-226.

[7] Kovach, S., & Ruggiero, W. V. (2011). Online banking fraud detection based on local and global behaviour. In *Proc. of the Fifth International Conference on Digital Society, Guadeloupe, France* (pp. 166-171).

[8] Pandit, S., Chau, D. H., Wang, S., & Faloutsos, C. (2007, May). Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web* (pp. 201-210).

[9] Aleskerov, E., Freisleben, B., & Rao, B. (1997). Cardwatch: A neural network based database mining system for credit card fraud detection. In *Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFEr)* (pp. 220-226). IEEE.

[10] Soltani, N., Akbari, M. K., & Javan, M. S. (2012, May). A new user-based model for credit card fraud detection based on artificial immune system. In *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012)* (pp. 029-033). IEEE.

[11] Carminati, M., Caron, R., Maggi, F., Epifani, I., & Zanero, S. (2015). BankSealer: A decision support system for online banking fraud analysis and investigation. *computers & security*, *53*, 175-186.

[12] Sherly, K. K., & Nedunchezhian, R. (2010, December). BOAT adaptive credit card fraud detection system. In *2010 IEEE International Conference on Computational Intelligence and Computing Research* (pp. 1-7). IEEE.

[13] Lepoivre, M. R., Avanzini, C. O., Bignon, G., Legendre, L., & Piwele, A. K. (2016). Credit card fraud detection with unsupervised algorithms. *J. Adv. Inf. Technol*, *7*(1).

[14] Krivko, M. (2010). A hybrid model for plastic card fraud detection systems. *Expert Systems with Applications*, *37*(8), 6070-6076.

[15] Şahin, Y. G., & Duman, E. (2011). Detecting credit card fraud by decision trees and support vector machines.

[16] A. Bifet, R. Gavalda, G. Holmes, and B. (2018). Machine Learning for Data Streams: with Practical Examples in MOA.

[17] Lopez-Rojas, E., Elmir, A., & Axelsson, S. (2016). PaySim: A financial mobile money simulator for fraud detection. In *28th European Modeling and Simulation Symposium, EMSS, Larnaca* (pp. 249-255). Dime University of Genoa.