



INTERNATIONAL
HELLENIC
UNIVERSITY

Mitigating concept drift in data mining applications for intrusion detection systems

Koutrouki Evgenia

SID: 3307160006

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Communications and Cybersecurity

DECEMBER 2017
THESSALONIKI – GREECE



INTERNATIONAL
HELLENIC
UNIVERSITY

Mitigating concept drift in data mining applications for intrusion detection systems

Koutrouki Evgenia

SID: 3307160006

Supervisor:

Prof. Georgios Ioannou

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Communications and Cybersecurity

DECEMBER 2017

THESSALONIKI – GREECE

ABSTRACT

The phenomenon of concept drift is defined as the unexpected behavior of a data stream under changing environments. It is considered one of the major problems of data mining applications. Intrusion detection systems are correlated with data mining due to the fact that they collect, monitor and analyze data, so as it is expected, they also experience concept drift. By analyzing the different types of data mining, machine learning, intrusion detection systems and concept drift, a new method is developed with the aim to mitigate this phenomenon in data mining applications for intrusion detection systems.

Koutrouki Evgenia

31/12/2017

Contents

ABSTRACT	3
CHAPTER 1	
INTRODUCTION	6
1.1 Context.....	6
1.2 Problem Statement.....	7
1.2.1 Concept Drift.....	7
1.2.2 Machine Learning and Data Mining	9
1.2.3 Intrusion Detection Systems	10
1.3 Aims & Objectives	11
1.4 Dissertation Layout.....	11
CHAPTER 2	
LITERATURE REVIEW.....	12
2.1 An overview of data mining techniques.....	12
2.1.1 Classification.....	12
2.1.2 Regression	14
2.1.3 Association	15
2.1.4 Clustering.....	16
2.2 Data Mining Techniques in Intrusion Detection Systems	17
2.2.1 k-Nearest Neighbor Algorithm.....	17
2.2.2 Naïve Bayes Algorithm.....	18
2.2.3 Decision Tree Algorithm.....	19
2.3 Developed Methods for Mitigating Concept Drift in Intrusion Detection Systems.....	20
2.3.1 MineClass Algorithm.....	20
2.3.2 Early Drift Detection Method	21
2.3.3 DWM Algorithm.....	22
2.3.4 FAE Algorithm.....	23
CHAPTER 3	
METHODOLOGY	25
3.1 Software Setup	25
3.1.1 Waikato Environment for Knowledge Analysis (WEKA)	25
3.1.2 Massive Online Analysis (MOA).....	26
3.1.3 Eclipse	26
3.1.4 Operating System.....	27
3.2 Experimental Dataset	27
3.3 Algorithm Implementation	28

3.3.1 Objective	28
3.3.2 The New Method Hypothesis.....	29
3.3.3 Classifier Behavior.....	30
3.4 Data Analysis	30
3.4.1 Output Information.....	30
3.4.2 Testing Method	31
3.4.3 Evaluation Methods	31
CHAPTER 4	
RESULTS AND ANALYSIS	33
4.1 DATASET PRESENTATION	33
4.2 RESULTS IN WEKA ENVIRONMENT.....	34
4.1.1 Hoeffding Tree	35
4.1.2 Naïve Bayes	37
4.1.3 J48.....	39
4.1.3 lazy IBk	41
4.3 RESULTS IN MOA ENVIRONMENT.....	43
CHAPTER 5	
CONCLUSIONS	45
REFERENCES.....	46
APPENDIX	48

CHAPTER 1

INTRODUCTION

This first chapter is the one where the introduction on data mining, machine learning and intrusion detection system takes place. Along with that, the problem of concept drift is presented and analyzed. A preview of the methodology used is also explained.

1.1 Context

When discussing about technology, the words machine learning continuously comes up. Those two terms are paired together because with technology evolving and becoming such a big part of people's everyday lives, the need of automating several processes is a critical issue. Numerous techniques have been developed to automate both systems and communications but also new problems appear along with that.

Securing sensitive data is also related to automated processes. There are different types of systems that are built for detecting and preventing attacks that aim sensitive data. That kind of systems are called intrusion detection systems (IDS). Intrusion detection systems use machine learning algorithms to become smarter and faster when it comes to detecting a malicious movement.

Data that are continuously received over a long period of time are known as data streams. As those sequential data are arriving they are processed and analyzed by the algorithms so that useful information can be acquired [1]. The data streams can be processed either offline or online. In offline training the whole training dataset is available when the model is training. When the model is trained and the training process has finished, it can be used for predictions. In online training the data keep arriving and the processing is sequential. The training dataset is not fully complete during the training of the model. The model keeps training as new training data keep coming up. The trained model is used for predictions before the training dataset is whole [2].

Data mining algorithms have to deal with those high speed streaming data and as it is expected problems appear with this process. The algorithms must have the ability to absorb the new

streaming data and adapt to the unpredicted changes in the streaming environment. This unexpected behavior of data distribution as it evolves through real time conditions is the phenomenon defined as concept drift.

Intrusion detection systems use data mining techniques to be more accurate in the detection of attacks and be able to prevent them as well. With data streams arriving one after another, intrusion detection systems must cope with concept drift so that they can be robust and able to adapt fast when the change occurs. In this paper a new method for handling concept drift is presented and explained. The main tool that is used is Weka, an analysis software that allows experiments to take place from the University of Waikato and the dataset that is used is the KDD '99 Dataset, which is a dataset related with network intrusion detection.

1.2 Problem Statement

In the following parts the problem of concept drift is stated and an explanation on machine learning, data mining and Intrusion Detection Systems is provided, in order to understand the relation between those terms.

1.2.1 Concept Drift

As the definition of concept drift states, the unexpected behavior of a data stream under changing environments is a major problem in data mining. In data mining applications for intrusion detection systems, concept drift forces the system to make less accurate predictions over time. That is the result of the dynamically changing and non-stationary environments [2]. The adaptation of the system when new data are incorporated is the most important operation of a machine learning algorithm and the correct and early identification of the small time windows which the concept drift occurs is crucial.

There are different time windows that concept drift occurs and that is the reason why there are different types of drifts. The four known occurring types are sudden (or abrupt) drift, gradual drift, incremental drift and reoccurring drift [3]. Sudden concept drift is easier to detect in the data stream than gradual concept drift. The change in sudden drift is instant between concepts and the algorithm can identify it easier and adapt quicker than the second type. Gradual concept drift

introduces a more complicated problem. The drift, in this case, is expressed slowly and without a certain pattern. The system is not able to detect this drift quickly due to its slow speed of change and the lack of sudden concept change. In incremental drift the changes are also slow, but in this case this slow changes between the intermediate concepts are sudden and more obvious than the changes in gradual drift. Finally, in the reoccurring drift, old or new concepts often reappear over time and that is why it is also known as seasonal drift.

Types of concept drifts

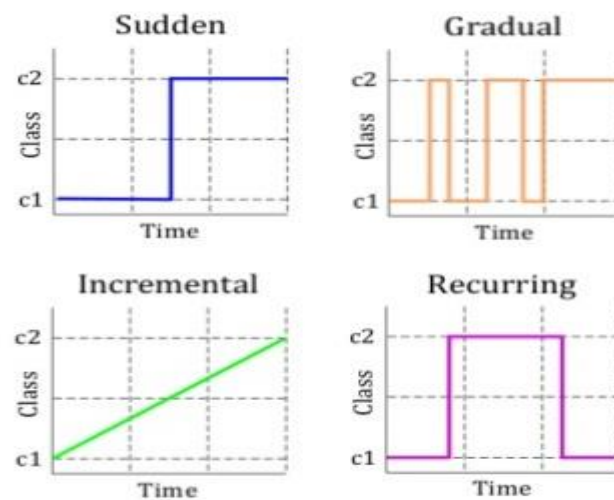


Figure 1. Types of Concept Drift [9]

Although there is no comprehensive survey for handling concept drift [2] there are techniques that have been developed so that systems can cope with it. Different learning modes exist that are mentioned in the first part of the introduction (offline and online learning) and different strategies.

According to the strategies that exist, they are divided into evolving and trigger-based when the model needs to be updated and they are also divided into using ensembles or a single classifier [3]. On one way, when the evolving strategy is used, the learning models will continue evolving through time and keep updating. On the other way, when a trigger-based strategy is used, the data streams that are arriving are monitored and if there is a suspicion of change then an alert is triggered by the triggering mechanism and the update process begins by taking into account the latest data in the system and dropping the old ones [3].

The learning system, as it is stated can use ensembles or single classifiers for the learning process. If the ensemble technique is used, the system has the ability to remember some of the concepts. An ensemble contains multiple single classifiers that are isolated one from the other and that is why the accuracy of this technique is higher than using a single classifier only. The technique falls both in the two previous strategies mentioned above and can be either evolving or trigger-based. If only a single classifier is used, the system can only make a prediction for a single model at every time. When this technique is applied, the system keeps in memory only the last updated model.

1.2.2 Machine Learning and Data Mining

Machine learning is one of the most promising areas of technology. It is defined as the complex computation processes of automatic pattern recognition and intelligent decision making based on training data [5]. From those computations a model is created, known as learning model or learning algorithm. In simpler terms, the machine learning method uses the patterns from the training dataset so that it can learn the form of a classifier [5].

Machine learning can be categorized into two learning groups, supervised learning and unsupervised learning. In supervised learning the algorithm is trained after pairs of input and output are given (those data are called labeled data). Those data are used so that a learning model to be trained. The algorithm uses the input data in a way that it can predict the output data using the minimum resources. In unsupervised learning the algorithm is trained only with input data because no output data are given in the training dataset (in this case the data are unlabeled). While the model is training, it brings together key features that appear in the training data and uses them to create a pattern so that the data to be summarized into clusters with similar key features. Unsupervised learning is difficult to assess in comparison with supervised learning because of the lack of labeled data.

The performance of the machine learning algorithms depend on the performance of the evaluation metrics that are used in each case, the difficulty of the problem and how well the model is trained.

Data mining can be defined as the process of analysis of large databases in order to find the knowledge in them [6]. In simpler words, data mining uses analysis tools to look for useful

patterns in the data provided. Data mining uses machine learning algorithms to achieve this purpose.

Data mining can be categorized into two groups, supervised and unsupervised, just like machine learning. The difference between those groups is exactly the same with the machine learning groups. In supervised data mining, labeled data are used in the training dataset to create a model and in unsupervised mining, unlabeled data exist in the training set, so the model is created by grouping the data based on similar key features.

1.2.3 Intrusion Detection Systems

When using an intrusion detection system, the goal is for the system to understand which of the incoming data are intrusive and which are non-intrusive and separate them [7]. Intrusion detection systems use data mining so that they are capable of predicting future results based on the given data and also to be adaptable when they deal with streaming data and must respond instantly to threats.

In intrusion detection the two most basic categories are: misuse detection and anomaly detection. Misuse detection uses labeled data to train the machine learning algorithm. Models are automatically updated according to the input arriving in the system. A misuse detection intrusion system can only identify attacks that are already known and attacks that are very similar to those known attacks (attacks that most of the instances they have are similar to a known attack). So the system in this case makes its decisions based on the labeled data that are provided, so the decision is made based on prior knowledge. Misuse detection systems have a high percentage of accuracy when it comes to the already known attacks [8].

Anomaly detection intrusion systems, on the other hand, operate with unlabeled data. So the machine learning algorithm does not have a training dataset that contains known attacks. In this case the training of the algorithm is based on the behavior of the data. The system observes the data and defines by itself a normal data behavior, so if there is a behavior that does not conform to the rest, the system raises an alarm. With anomaly detection the system can identify previous unknown attacks and that is a great advantage.

1.3 Aims & Objectives

The aim of this dissertation is to develop a new data mining model, through analyzing datasets consisting of labeled data, in order to mitigate a phenomenon known as concept drift, in data mining applications for intrusion detection systems.

For accomplishing this aim clear objectives are set to help defining the developing and research process. Those are presented below.

- To locate successfully existing concept drift in the dataset.

There are various techniques detecting concept drift in data stream mining, therefore the most suitable ones for the system are going to be displayed and explained. This procedure is needed so that different aspects to be examined.

- Development and proposition of a new method.

Having examined all this previous work carefully, the new method will be created in a way that can bring correct results and manage to mitigate the concept drift problem.

- Maximizing the system's accuracy and minimizing the execution time.

The system should be able to adapt rapidly in changing patterns, false positive alerts should be reduced in order the system's prediction ability becomes more accurate and different set of variables will be used to reach the most efficient level of operation.

1.4 Dissertation Layout

This dissertation is structured with the following way. In the second chapter there is a presentation of previous work done on this specific matter along with an explanation of data mining techniques in general and data mining techniques that are related to intrusion detection systems. Also, methods that have been already developed to address concept drift are explained. In the third chapter, a demonstration of the new developed method for mitigating concept drift is made. In the fourth chapter, the testing phase is presented and the results of the experiments are introduced as well. In the last chapter of this dissertation a discussion about the results is taking place and the conclusions are displayed.

CHAPTER 2

LITERATURE REVIEW

In this chapter a well-documented review of previous work related to the subject is presented. The first section describes the most basic and common used techniques in data mining. The second section concerns data mining algorithms that can be used in an intrusion detection system. Last, in the third section, known detection techniques that are used to mitigate concept drift in intrusion detection system are presented.

2.1 An overview of data mining techniques

In data mining different techniques are used for different operations, according to the kind of applications that the technique targets. Nowadays, too many techniques exist because different companies develop their own for the purpose they want to serve [10]. However, there are basic techniques that every new one is based on. Those techniques are going to be analyzed in this section in the length of operation, testing, measures that they use, algorithms that are based on and where they are most commonly applied.

2.1.1 Classification

Classification is a supervised mining predictive task for predicting a value of categorical variable (also known as target). It is supervised due to the fact that it uses the class itself in every training example and predictive because it predicts an outcome at a current moment. Among data mining techniques, classification is the one that is the most commonly used. In classification an item is assigned to one of a set of classes. This happens because the goal in the technique is to make accurate predictions about the target class from the data for every example [11].

To understand classification better there are some points that need to be made from the beginning. First, the historical data used in each case are split in two datasets. The first dataset

is used in order the predictive model to be created and the second dataset is used so that the model can be tested. Second, the process is divided in two phases, the learning phase and the testing phase. Third, classifications are rules created in order to be applied in a new set of data so that a prediction to be made or a decision to be taken. Knowing all the above, it is time to find out how classification works.

The task begins by taking the first dataset of the historical data, also known as the training set, so that the learning phase can be initiated. In this phase the classifier is built. In this dataset the outcome (which is an attribute) is known and other different attributes are defined. The goal in this phase is to combine all the attributes correctly so that the algorithm to understand how the certain value of the outcome is derived from the given data. In simpler words, the model must understand how the prediction was made so that the decision can be done. From this process a set of rules is created, also known as classification rules. In the second phase of the task the rules must be tested, so the second dataset is used, also known as the testing dataset. In the testing dataset the outcome does not exist, but the rest of the attributes exist. The classification rules created, now must be tested so that their accuracy to be estimated. So the rules are applied in the testing dataset and if the outcome is correct then they are considered solid and useful and can be used for future predictions. For this to be done value of the outcome is compared to the previous dataset's known values. Testing dataset must be consistent with the training dataset that is used to build the prediction model that is the reason that both datasets come from the same historical one [11].

In order to test the data there are different test metrics that are used such as accuracy (the ability of the classifier to predict correct the target values compared to the known values), speed (how fast the classifier can compute the target values), scalability (how fast the classifier is constructed from the data provided), lift (the ratio of the classifier's predictions due to the original known target from the training data), confusion matrix (a table that show the correct and false predictions that the classifier predicted compared to the original values) and robustness (the correct predictions that the classifier is able to make when the data is provided) [5]. According to the data that will be examined there are different algorithms that can be used that will provide better results in each case.

2.1.2 Regression

Regression is a supervised mining predictive task for predicting numeric value. It is supervised due to the fact that it uses known target at the beginning of the task and predictive because, like classification, it predicts an outcome at a current moment. Regression is used in predictive analytics, just like classification, but the difference is that while classification items are assigned to one of a set of classes, regression is used for predicting continuous numbers or a range of them. When a feature is given and its value must be predicted, regression is used in order to predict the value based on the values of the rest features in the dataset [12].

There are different types of regression such as Linear and Nonlinear regression and Multivariate Linear and Multivariate Nonlinear regression [12]. The historical data provided in a regression task are divided in two datasets. The first dataset is used to build the model and the second dataset is used to test it. Again, like classification, there are two phases: the training phase and the testing phase. With those in mind, a description of a regression task is provided.

The task begins with the training phase where the training dataset is used in order to build the model. A regression algorithm is applied in the dataset so that an estimation of the value of the target to be made as a result of the variables that are used for the prediction. The regression model is built from the combination of the training data, the prediction values and the predicted outcome. After the model is created, it can be applied in the testing dataset so it can be tested. In the training phase of the task it is also included the computation of the variables that can be used to minimize the error percentage in the process. Keep in mind that both datasets are configured in the same way because they derive from the historical dataset. So next comes the testing phase where the testing dataset is used. In this dataset the value of the target is, as it is expected, unknown. To test the model statistics are computed in order to determine the deflection of the testing results from the original known values. If the performance of the model is accurate then it is ready to be used to other new datasets.

The test metrics that are used to estimate the model's error percentage are the Root Mean Squared Error (also known as RMSE) and the Mean Absolute Error [12]. Of course, there are many regression algorithms and that leads to different ways to estimate errors in them.

2.1.3 Association

Association is an unsupervised mining descriptive task for finding association rules in a dataset and it is transaction-based. It is unsupervised because there is not a pre-established target in the data and descriptive due to the fact that it tries to establish correlation between variables and discover patterns amongst them. Association is usually applied in large datasets and binary datasets. It is one of the most recognizable techniques in data mining. Association rules are trying to discover repeated item sets among a dataset. In a transaction, a set of items is called an attribute [13].

To become more familiar with the association technique further explanation must be provided. Association is not meant to be used in numeric datasets but as stated before in datasets that contain items and transactions. A transaction can be declared as a subset of items in a larger item set. An association rule is an uncovered pattern among related items. So, for pattern identification the items must be of the same type. In association mining a rule must be claimed as interesting. This is also known as rule interestingness. For a rule to be interesting a threshold of the two basic association measures (confidence and support) must be supported. The threshold is set by the user. A rule that does not satisfy the minimum threshold of the above measure is not considered interesting. If it does satisfy the threshold then the rule is called strong. If all possible association rules are found then it is considered to have completeness. Of course, in each case different thresholds are being set to match the type of the application and data. So association rules differ according to the dataset that is examined. Below, an analytical explanation of the two measures mentioned is given.

Support is a percentage derived from the transaction dataset provided in each case. It describes the transactions (item sets) that the applied association rule satisfies. It is the first step of an association task. A rule that has a high percentage of support is more likely to be effective. Confidence is a percentage derived also from the transaction dataset and it describes how frequently the association rule has been found to be reliable, in other words true among the item sets. It is the second step of an association task. A rule that has a high percentage of confidence is considered interesting because its prediction capability is high. So a strong rule that meets the minimum thresholds of the two measures declares that if a certain item exists there is high probability of another related item to exist.

Other measures that are used to test association rules are lift (measures if a rule is independent) and conviction (measures how often an item exist without the item that's related to don't) [13].

2.1.4 Clustering

Clustering is an unsupervised mining descriptive task for finding relative sets of data into clusters (also known as natural groups). It is unsupervised due to the fact that there is not at target in the data and the algorithm is learning on its own and descriptive because the algorithms searches for relationships that exist among the dataset. Clusters are defined as groups or classes of data objects or classes that are highly similar to one another in the same cluster and highly dissimilar in groups of data objects in a different cluster [14].

So a clustering algorithm uses one or more variables to begin with so that it can identify a cluster. Between the data there might be relationships that are not noticeable before the clustering task begins. The difference from classification is that a classification model divides data that are assigned in classes that are already defined. Clustering is mostly used in datasets that contain very large piles of data and in them exist too many attributes.

A clustering task does not learn from an imported example but from observing the dataset. That is and the main advantage of clustering because algorithms can identify a single feature in order relationships and create the clustering model. The resulted clusters must be of high quality. So one basic measure is the centroid distance. This means that every cluster has a center and every object in the cluster is divergent from the center a certain Euclidean distance. Centroid distance is the average of all those distances. If an object is close to the cluster centroid it is assigned to this cluster. One important information is that clustering due to the way the task works is highly used in anomaly detection because it can spot objects that do not belong in any clusters. Clustering is a task that can be reversed and used that way. In a dataset the algorithm can suppose that a cluster exists in a particular point and after to use the variables to make a comparison and find out if its assumption is true [15].

In clustering the other measures that are used for establishing a model's efficiency are, like association, support and confidence [14].

2.2 Data Mining Techniques in Intrusion Detection Systems

Having discussed the basic data mining techniques in the previous section, it is now time to see which of them apply in an intrusion detection system. An intrusion detection system is in the second line of defense, as the first is user prevention techniques. Through it, a computer system is monitored and can also be controlled for certain actions. Data mining is important in intrusion detection because data streams are continuously keep coming and they must be monitored and processed in real time. So data mining helps to make the intrusion system more “clever” and adapt with ease in those real time changes. The most common types of techniques for intrusion detection are classification techniques.

2.2.1 k-Nearest Neighbor Algorithm

The k-nearest neighbor (also known as k-NN) algorithm is one of the most famous non parametric supervised algorithms and it is used both for classification and regression methods. It is non parametric because it does not require previous or specific details about how the current training set is structured [16]. k-NN is best suited for small datasets because in the large ones it takes too much processing time. That is why it is also defined as lazy [17].

At first, in the training set of the data, the algorithm is searching for the k closer objects to the object it wants to test. By searching, k-NN calculates the distance (usually the Euclidian) between the new data point and the k objects closer to it. The algorithm wants to classify this new testing data point by predicting a class for it.

The three basic elements of the k-NN algorithm are the already existing data points which are labeled, a distance or another measure so that the distance between the objects to be calculated and finally, how many the nearest neighbors are (k value).

For an unlabeled data point to be classified, the algorithm must calculate the distance to all the labeled objects, identify which are the nearest neighbors around it (k neighbors) and also identify the labels of the classes of those neighbors so that to used them for assigning a class label to the unlabeled data point. When all the above actions are done, the data point will be classified according to the majority class of the labeled neighbors [17].

Considering the performance of the k-NN classifier there are some important issues. The selection of the number of k neighbors is one of them. For a very small k , the classifier may result to sensitivity in noise objects. For a very large k , the classifier may result to contain many neighbor objects from other classes. Having this in mind, it is apparent that the classifier operates better in smaller datasets because in larger ones the processing time for testing all the objects is too long. Another issue is the choice of the measure for distance. The classifier can use different measures for distance but the best choice is a measure that considers, apart from small distance, also the likelihood between the object and the classes. The most common measure used is the Euclidean distance [17].

The k-NN classifier is based only in the examples that are provided to it and is a classifiers that, when used in intrusion detection systems, can accurately detect the intrusion in the system and also have a low false positive rate in its results [16].

2.2.2 Naïve Bayes Algorithm

Naïve Bayes is an algorithm commonly known for its simplicity to be constructed. It is based on the Bayes Theorem (that is where it got its name too) and it uses conditional probabilities to classify the unlabeled objects. Naïve Bayes does not need parameters that are too complicated and that why its interpretation is relatively easy even by unskilled, in classifier technology, users. That is why it is also known as idiot's Bayes [18]. This classifier can be applied in very large datasets and have decent results. Naïve Bayes is also an algorithm that is robust to noise and it is used to deal with supervised classification problems (both binary and multiclass).

A great advantage of the Naïve Bayes classifier is that it operates very fast and that it is because of its ability to scan the data with only a single pass. That makes both the training phase and classification very fast. A disadvantage of the classifier is that by its nature it makes an assumption that all the features are independent statistically. Although, besides this assumption, Naïve Bayes has very good results when compared with other classifiers [18].

As it was stated, the classifiers makes the assumption that the features in the dataset are independent. Although this original assumption, it has been proved that Naïve Bayes provides good results even in dataset that functional dependencies are present. That is because the classifier does not interact directly with the degree of the feature dependencies that are measured [19].

The aim of Naïve Bayes classifier is to develop a way of assigning the objects to their classes based only on the new object's vector of variables. This aim derives from the fact that the set of labeled features that is provided to the algorithm have each a known vector of variables.

2.2.3 Decision Tree Algorithm

Decision tree algorithm is a classifier that is based, like Naïve Bayes, in conditional probabilities. The main task of this classifier is to generate rules so that it can classify the unlabeled objects to the classes. By rule, the classifier defines a statement, based on the conditional probabilities, that is easy to be interpreted by the database so that the last can label a set of related records. With the rules the decision model becomes more transparent [20].

A decision tree is constructed in the form of nodes and edges. The first node of the tree is called root node and has only outgoing edges. There are two more kinds of nodes: the internal or test nodes and the terminal or leaf nodes. The first are nodes that have both incoming (one edge) and outgoing edges (two or more). The second do not have outgoing edges (that is and the reason why they are called terminal) but have one incoming edge. The root and test nodes represent the attributes than an object will have and the terminal nodes represent the class label [20].

The process of classifying a new object using a decision tree algorithm is simple. A number of sequencing questions are being asked to the new object. Depending, in every step, on the answer of the first question, the next relevant question is being asked. Those set of questions and answers are building the decision tree. When the process is finished and the last question has been answered, the decision tree is ready to easily classify the unlabeled object to a class label. The paths between the nodes of the tree are forming the algorithms rules. The decision of how a testing node will split is based on the value of the attributes [20]. The algorithm decides which of the given attributers has a better of closer value to the unlabeled object and follows the one that suit it better.

For measuring the complexity of the decision tree some of the following metrics are used: tree depth, number of attributes used, total number of nodes and total number of terminal nodes [28].

A decision tree algorithm has advantages such as it can be used to solve binary classification problems and also multiclass classification problems. The interaction between the user and the

model is very small and that does not stop the classifier from being accurate. Decision tree is a fast classifier both in an accurate and timely manner [21].

2.3 Developed Methods for Mitigating Concept Drift in Intrusion Detection Systems

Mitigating concept drift successfully has been an important task for the scientific world. There is not a developed method that eliminates it completely yet. In this section, related work for the already developed concept drift methods is presented. Most of them are novel methods and focus on detecting concept drift early so that the intrusion detection system can adapt quickly and other focus on enhancing already existing methods by developing different detection tests. Specifically all the methods deal with online streaming data.

2.3.1 MineClass Algorithm

MineClass algorithm (Mining novel classes in data streams) is an algorithm that focuses on two parts: detecting new classes (also known as novel classes) and handling concept drift in streaming data. Detecting new classes that might evolve later on the data stream is a feature that not many of the techniques that already exist can perform [22]. The algorithm does not use a single traditional classifier to identify the new evolving classes because a traditional classifier can identify correctly only the classes that were used to train it in the training data.

There are two terms that the algorithm relies on, and are also mandatory to be met, cohesion and separation. With cohesion meaning that all the data points that belong to the same class must have a close distance to each other and with separation meaning that all these data points must have a far distance from all data points that belong to different classes. That is and the main concept that this technique is based on. So in simpler words, the MineClass algorithm is detecting automatically the appearance of a new class in a dataset that suffers from concept drift by measuring the cohesion between the testing data points and the separation of the last from the training data points.

The MineClass algorithm is introducing a novel class detector and proposing its combination with a traditional classifier. This serves the purpose of discovering the novel classes that might appear

without having to manually retrain the data where the new class has been added so a new completed model to occur [22]. The goal of this proposal is not only to be able to identify the deviation of a data point from the normal behavior but also to detect if a group of outlying data points (known as outliers) have mutual connection.

Coming to the second part, that is handling concept drift, an approach of ensemble classification is adapted. An ensemble of M classifiers (in this case, decision tree and K-NN) is maintained in order to classify the unlabeled data. There is a division of the data stream in parts (named chunks) of equal size. Every one of these chunks holds a part of the memory and it is processed online. As new models are built, they can replace, if there is a need, one of the already existing. The ensemble of classifiers is evolving and follows the updated concept.

The process the algorithm follows is the next: when a new chunk arrives, its separation is tested. If they are well separated, a filter is applied the data so that to eliminate outliers that may exist because of concept drift. Then, the cohesion of the outliers is tested and the new class is set. After that, the training process continuous and the new model is built. The already existing model is replaced by the new one. The ensemble of classifiers is also update along with the model with the new instances that have been labeled from the training. That is how the already built model is continuously updated with novel classes existing in it.

2.3.2 Early Drift Detection Method

Early drift detection method (or EDDM) is technique for handling concept drift in evolving data streams by calculating the estimated distribution of the distances between errors of classification [23]. It is a method that can be handled with two ways: applied inside an incremental and online algorithm and also by using it to wrap a batch learning algorithm.

Examples are generated because distribution changes over time. So assuming that the data come from a stationary source might be false. That is why the detection should also be adapting over periods of time. Concept drift is easier to be detected if the change in the data distribution is abrupt. The real challenge, that this algorithm is trying to face, is the detection of concept drift when the data distribution is experiencing a slow gradual change. EDDM has a good result percentage in this latter case.

The method that is used in EDDM in order to detect concept drift is by measuring the distance between classification errors. This provides the algorithm with a faster adaptation over time without having an increased rate of false positive results. In this method two thresholds are defined: the drift level (concept drift exists after this threshold is met) and the warning level (maximum distance between errors).

The EDDM algorithm process the examples in a sequence of trials with the examples arriving one at a time. When the new example arrives the algorithm classifies it with help from the already existing model. The training model adapts if the warning threshold for the distance between classification errors is met and it starts searching for concept drift in the data distribution. That is how the model will predict more accurate. The warning threshold in EDDM is set for a minimum of 30 errors. After those errors have occurred, EDDM uses the second threshold to search for concept drift in the data. Given that, it is apparent that when the distance between the errors is small the model operates well but if the distance between the errors is large the model needs to adapt. The learning algorithms that are used here are a decision tree and two K-NN algorithms [23].

2.3.3 DWM Algorithm

DWM (or Dynamic Weighted Majority) algorithm is a general ensemble method used in online learning for handling concept drift and is an extension of the Dynamic Majority Algorithm [24]. This algorithm is using prediction strategies known as experts to boost its performance. It supports four mechanisms aiming to handle concept drift. The first one is that the online learners of the ensemble are trained. Second, those learners have weights that are based on how they perform. Third, the learners can be removed also based on how they perform and fourth, depending on the ensemble's global performance, new experts can be added.

The experts in the ensemble are weighted, new can be added and old can be removed, as it was mentioned above. DWM relies on those weights through this addition and deletion process. The entire algorithm's performance is considered as global. Based on the global performance, DWM is adding or removing experts. If a mistake is produced by the global algorithm, then an expert is added. If a mistake is produced by an expert, then DWM assigns a lower weight on it. Last, if, during the whole training process, an expert performs inadequately (this is something that can be observed from the expert's low weight) DWM will remove it from the ensemble.

The set of experts in the algorithm are assigned with a certain weight. The number of training examples inserted have a class label and a feature vector each. The algorithm uses a multiplicative factor (β) that is using when it want to decrease the weight of an expert, because of its incorrect predictions. A parameter (θ) is used as a threshold and when an expert drops below it, by performing poorly, it gets removed. DWM also has a parameter p , which checks how often an expert is created or removed [24].

The way DWM proceeds is actually simple. It begins with forming its ensemble, which, at first, contains only one learner (the learning algorithms in this example are Naïve Bayes and Incremental Tree Inducer). The learner has the maximum weight that can be assigned (that is 1) by the algorithm. For its first prediction, the learner uses a default class or previous background knowledge. Then, DWM provides the learner with an example (or more than one) coming from the data stream and waits for the learner to classify it. If the learner makes a false prediction, then its weight is decreased with the help of the multiplicative factor. This first prediction of the learner, is also the global prediction of the algorithm due to the fact that there is only a single learner. It is apparent, if this global prediction is false, a new learner will be created and added to the ensemble, with its weight set to the maximum level. Then, DWM provides the ensemble with a new example to be trained. When the training process comes to an end the algorithm will produce the new global prediction.

When the ensemble contains many experts, DWM gets from each one of them a classification and if an expert makes an incorrect prediction then its weight is decreased. Despite of the accuracy of the predictions of the learners, the algorithm uses them (the predictions) combined with their weights so that it can summarize their weights to compute one for every class. The class with the highest weight will be considered more important and its prediction will be set as global [24].

2.3.4 FAE Algorithm

FAE (or Fast Adapting Ensemble) algorithm is an algorithm that can cope with concept drift by adapting quickly and it has good performance when dealing with repeating patterns [25]. The algorithm can handle abrupt and gradual changes related with concept drift. The examples derived from the data stream are divided in parts of equal size (chunks) and an advantage of the algorithm is that in the learning process more than one examples can be processed

simultaneously without having to wait for an example to end and then add a new one. As a result the classification mechanism can adapt faster. The classifiers used are also divided into active and inactive during testing the data. This happens because of their behavior and relevance with the data in present.

The global decision of FAE is made by the decisions of its active classifiers (here Hoeffding Tree and VFDT are used). FAE stores all learners that are inactive because of their relation to older concepts and can be found useful if those concepts reoccur. The inactive status can be rapidly change to active if a certain concept appears. The importance of a classifier also depends on two parameters in the algorithm which describe the importance of the learner's behavior when new or old data are processed, respectively. According to the parameter that has the higher value each time, the most suited classifiers will be used.

New classifiers can be created and be added to the ensemble if the algorithm thinks are needed. There is a limit of classifiers that can exist in the storage. If the storage is full and a new learner must be added then one of them of the already existing is replaced by the new one. There are rules for this process and a further elaboration will be provided.

The algorithm assigns to all classifiers a weight and it is using those weights for making a decision. Those weights can be updated from the algorithm by experiencing an increase or a decrease. That is depending on the classifier's performance while testing the data. When the algorithm's process begins there is only one active learner in the ensemble with the maximum weight assigned.

The implementation of a drift detector for determining if a new learner is needed whether the data are experiencing concept drift is another advantage of this algorithm []. If the status of the drift does not change a new classifier is not necessary. Given that, memory is saved in the storage and the learners that already exist are favored. Because of this, the algorithm can track gradual drift too. Only if the drift detector produces two consecutively alerts a new classifier is created. If the storage is full the procedure must be followed is that the first classifier that will be removed is one that is inactive and its weight is the lowest. An active learner can be deleted if and only if an inactive one does not exist but it is important to remember that at every moment one of the learners must remain active [25].

It should be noted, that none of the above methods makes use of filters in the preprocessing part of the classification process and also, none of them try to rearrange the form of a dataset in order to achieve better results.

CHAPTER 3

METHODOLOGY

In this chapter, the detailed research plan is explained along with the development of the new method. In the first section the software that is used for the experiments is presented. After that the datasets that have been chosen are described. Following, there is a demonstration of the algorithms implemented and finally a thorough data analysis takes place.

3.1 Software Setup

There are different software that are used, so that the experiments to be conducted. The operation system that is used during those experiments is presented and the setup for writing code for WEKA is displayed as long with the WEKA and MOA.

3.1.1 Waikato Environment for Knowledge Analysis (WEKA)

The data mining open source software WEKA is developed by the University of Waikato. WEKA contains a varied collection of machine learning algorithms and it is best suited for applications written in Java [26]. It provides a variety of tools for employing all the basic data mining techniques and has access to SQL databases.

In WEKA there are four available modes in the GUI Chooser. These options are the explorer, the experimenter, knowledge flow and the simple CLI. For the purposes of this specific dissertation the three first are going to be used. The software reads arff files that are imported manually. The format of the datasets used are in arff format.

When the dataset is loaded there are numerous options that are used for its examination. In this case, there is a use of several filters that remove the noise from the dataset in the preprocessing step, where the dataset is prepared so that the data mining functions can be used on it.

3.1.2 Massive Online Analysis (MOA)

Massive Online Analysis, most commonly known as MOA, is an open source framework developed by the University of Waikato [27]. MOA implements algorithms using online data streams. The online and offline algorithms of the software are used, in this dissertation, for classification.

MOA supports bi-directional interaction with WEKA and is also suited for Java-based applications. The framework supports arff format for the datasets. It also provides a tool dedicated for concept drift observation and analysis. This tool will be used for comparing the original dataset to the processed dataset for observing the mitigation of concept drift.

3.1.3 Eclipse

Eclipse is an integrated development environment (IDE) and it is most commonly used for coding in Java [28]. Due to the fact that WEKA is compatible for Java applications, WEKA API will be used for setting up Eclipse.

For setting up WEKA API in Eclipse the following steps must be followed.

Setting up WEKA API in Eclipse	
Step 1	Create a new project in Eclipse with the name 'weka-api'
Step 2	Built a path by selecting the 'Configure Built Path' feature
Step 3	In the 'Libraries' tab, add the external JAR file of WEKA

Table 1. WEKA API in Eclipse

By importing the WEKA JAR file in Eclipse all the WEKA libraries are ready for use.

3.1.4 Operating System

The characteristics of the operation system used are the following:

Characteristics	
Operation System Edition	Windows 10 Home
System Type	x64
RAM	8 GB
CPU	Intel Core i5-5200 CPU @ 2.2GHz 2.2GHz

Table 2. OS characteristics

3.2 Experimental Dataset

To investigate the problem, one reliable datasets is selected. That is the KDD Cup 1999 Dataset, commonly known as KDD-99, which is the most widely used dataset in intrusion detection. The dataset includes network intrusions that are simulated in a military network environment. It was created by processing the 1998 DARPA Intrusion Detection System for tcpdump portions [29].

There are four types of attacks includes in KDD-99. Those are probing, denial – of – service (DOS), unauthorized access from a remote machine (R2L) and unauthorized access to local superuser with root privileges (U2R) [29].

KDD-99 dataset experiences concept drift. That is because the dataset was generated by hand-injected attacks in a military network, so that different kind of attacks to be produced. All the data in the dataset are labeled and it is an ideal training set for supervised learning algorithms.

In order to find a solution for coping with concept drift, those artificial data of the KDD-99 dataset are used for all the experiments and testing of the proposed algorithm. One small detail that should be mentioned is that WEKA cannot support the size of the whole KDD-99 dataset, so for the purposes of this dissertation a smaller percentage of the dataset is used. This choice had the result of lower improvement percentages in the evaluation part. Naturally, this was expected, due to the fact that the smaller dataset contains a much smaller amount of instances than the original and also fewer cases of concept drift.

3.3 Algorithm Implementation

Different algorithms are going to be applied in the dataset using different classifiers and variables so that the most optimal result to be presented. With help from those algorithms, the new method is being developed and presented, for addressing the concept drift problem.

3.3.1 Objective

By carefully examining the dataset for locating concept drift and the speed of adaptation of the model in the classification process, one idea was always coming up, which was the use of filters. In order to test this theory, the new method began developing. Taking the above into consideration, in the following figure the new method is represented in two simple steps.

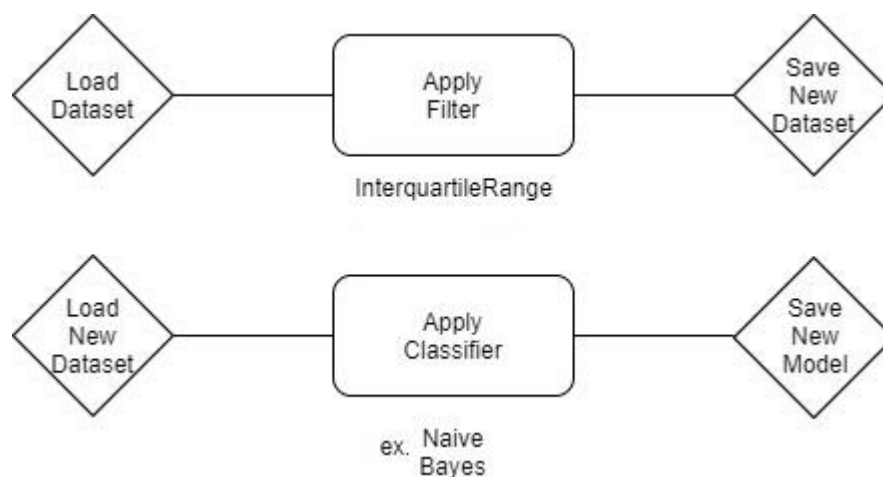


Figure. New Method representation

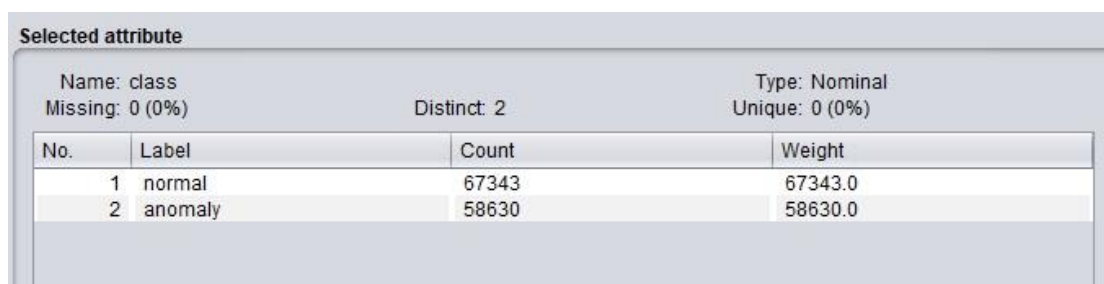
In the original dataset the Interquartile Range filter from WEKA is applied. The filtered dataset is saved and reloaded in WEKA so that the classifiers can be applied on it and get optimal results.

For the above process a Java code is developed, which can be used to automate the steps. The environment used for the code development is Eclipse with the weka.api configuration. The developed code is provided in the appendix.

3.3.2 The New Method Hypothesis

While developing the new method a fundamental hypothesis is introduced. To apply a filter on the dataset in the preprocessing part. The choice of the filter is neither arbitrary nor random. In order understand why the Interquartile Range filter is chosen, the following must be considered.

First, due to the fact that the KDD-99 dataset is labelled, it is known that it contains concept drift. The class attribute of the dataset has two labels, anomaly and normal. The weight of the instances between the two labels does not differentiate much, with 67343.0 in normal and 58630.0 in anomaly as it is also shown in the following figure. That means that with such a big anomaly weight there will be several concept drift changes.



The screenshot shows a 'Selected attribute' window in WEKA. It displays the following information:

- Name: class
- Missing: 0 (0%)
- Distinct: 2
- Type: Nominal
- Unique: 0 (0%)

No.	Label	Count	Weight
1	normal	67343	67343.0
2	anomaly	58630	58630.0

Figure 2. WEKA environment: class attribute weight

Second, from the introduction where concept drift is analyzed, it is known that concept drift forces the system to make less accurate predictions over time due to the instant or gradual change of the concepts included in the data. So the system must have the ability to identify in an early stage of time the presence of drift. Most those concept changes are observed in parts of data that extreme values and outliers exist.

Based on the two facts above, the Interquartile Range filter is chosen (the filter is applied with its default configuration). By definition, the filter detects all the extreme values and the outliers in the dataset based on their interquartile ranges and it doesn't include the class attribute in this process [WEKA]. The filter adds two more attributes in the list of the dataset's attributes, the outlier attribute and the extreme value attribute. So with the use of this filter the dataset is rearranged and the outliers along with the extreme values are defined clearly in it. As a result

when the new model is built the accuracy of the classifier improves. Also as it is shown in the evaluation chapter from the results of the experiments the classifier is able to detect quicker the presence of concept drift and that is why the system adapts faster.

3.3.3 Classifier Behavior

To support the hypothesis of the new method, tests were made using four different classifiers. Those are Naïve Bayes, J48, Hoeffding Tree and lazy IBk. The classification is performed, on WEKA, two times for each of the above classifiers, one with the non-filtered dataset and one with the new filtered dataset.

The decision to keep the outliers and the extreme values in the dataset after the filter is applied is conscious. The reason is that if the outliers are removed with another filter then the dataset doesn't not experiences concept changes because the number of instances is reduced by far and the dataset is partially clean. This has as a result to partially eliminate or fully eliminate the drift in the data because the data containing the drift have been removed. This move could be useful if the objective was to clean the dataset. With the current objectives, there is no point to remove the drifting data but the purpose is to thoroughly examine them and find a way to mitigate the drift they experience.

3.4 Data Analysis

Data Analysis is important due to the fact, that all the results can be evaluated with the same metrics so that the outcomes can be compared to each other. Here the test method for the classifiers is explained along with the evaluation metrics.

3.4.1 Output Information

When an experiment is done in the output is provided important information about it. This information is called Run information in WEKA.

The first thing observed is the classifier that is used to train the model. After that, the name of the dataset used is stated along with the number of instances and attributes in it. Also, the type

of the test mode used is also shown. Last but not least, the output also provides the time needed in order the model to be built.

3.4.2 Testing Method

The test method that is used in the experiments for training and testing the classifiers is the k-fold cross validation and the number of folds used is ten. With this method, the dataset is divided in ten parts and in every round, one of the parts takes the role of the test set and the other nine parts are combined and take the role of the training set. This procedure is repeated as many times as the existing folds, so in this case, ten times. In every lap the prediction error is calculated and at the end of all laps the mean prediction error is calculated and displayed as the percentage of error of the classifier.

3.4.3 Evaluation Methods

For evaluating the results of the experiments the following methods are applied.

In WEKA:

- **Correctly Classified Instances:** to determine the percentage of accuracy of the classifier.
- **TP Rate:** to determine the instances that are classified correctly (true positive).
- **FP Rate:** to determine the instances that are classified falsely (false negative).
- **Precision (aka Positive Predictive Value):** to determine the actual number of instances that belong to a class (true positives) among all the classified instances that belong to the same class.
- **ROC Area:** general performance of the classifier.
- **Confusion Matrix:** representing the correctly and incorrectly classified instances of the dataset assigned to each class.

In the following figure, a random example is displayed so that the above metrics can be visualized.

```

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

Correctly Classified Instances      22570           89.5919 %
Incorrectly Classified Instances    2622            10.4081 %
Kappa statistic                     0.7906
Mean absolute error                  0.1034
Root mean squared error              0.3152
Relative absolute error              20.7817 %
Root relative squared error          63.1897 %
Total Number of Instances           25192

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,912   0,123   0,895     0,912   0,903     0,791   0,968    0,969    normal
                0,877   0,088   0,897     0,877   0,887     0,791   0,963    0,946    anomaly
Weighted Avg.   0,896   0,106   0,896     0,896   0,896     0,791   0,966    0,958

=== Confusion Matrix ===
      a    b  <-- classified as
12272 1177 |    a = normal
 1445 10298 |    b = anomaly

```

Figure 3. WEKA evaluation metrics

In MOA:

- **Input:** Measures of the resources that are put into process in order to achieve an output.
- **RAM – Hours:** counts cost per hour and memory used at the same time. Every GB of RAM deployed for 1 hour equals to 1 RAM-Hour.
- **Time:** time needed to build the model.
- **Memory:** the memory required to store the building model and its parameters.

In the following figure, a random example is displayed again so that the above metrics can be visualized.

Evaluation				
Values				
Measure	Current		Mean	
<input checked="" type="radio"/> Input	2,95	-	2,99	-
<input type="radio"/> Ram-Hours	0,00	-	0,00	-
<input type="radio"/> Time	0,28	-	0,17	-
<input type="radio"/> Memory	0,00	-	0,00	-

Figure 4. MOA evaluation metrics

CHAPTER 4

RESULTS AND ANALYSIS

In this chapter, all the results from the method implemented are presented and analyzed based on the ways explained in Data Analysis from the third chapter.

All the outcomes from the previous implementations of the algorithms (including the new method) are presented.

4.1 DATASET PRESENTATION

To start the experiments, the original dataset was loaded in WEKA as it is presented in the figure below. It is clearly seen that the dataset has 42 attributes and that there is not a filter applied on it, due to the fact that in the filter section 'None' is selected.

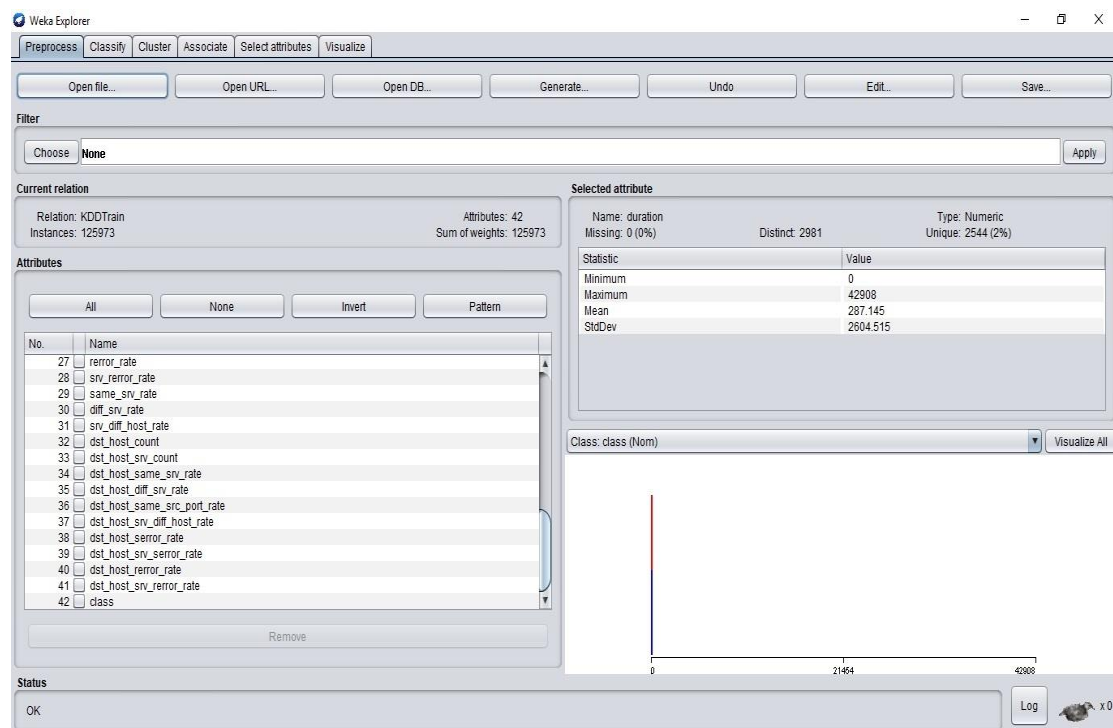


Figure 5. Original KDD-99 dataset

In the second figure below the filter is applied and the new filtered dataset is ready to use. In the filter section the option 'InterquartileRange' is selected in its default settings. The new dataset contains 44 attributes due to the fact that the attributes 'extreme values' and 'outliers' were added.

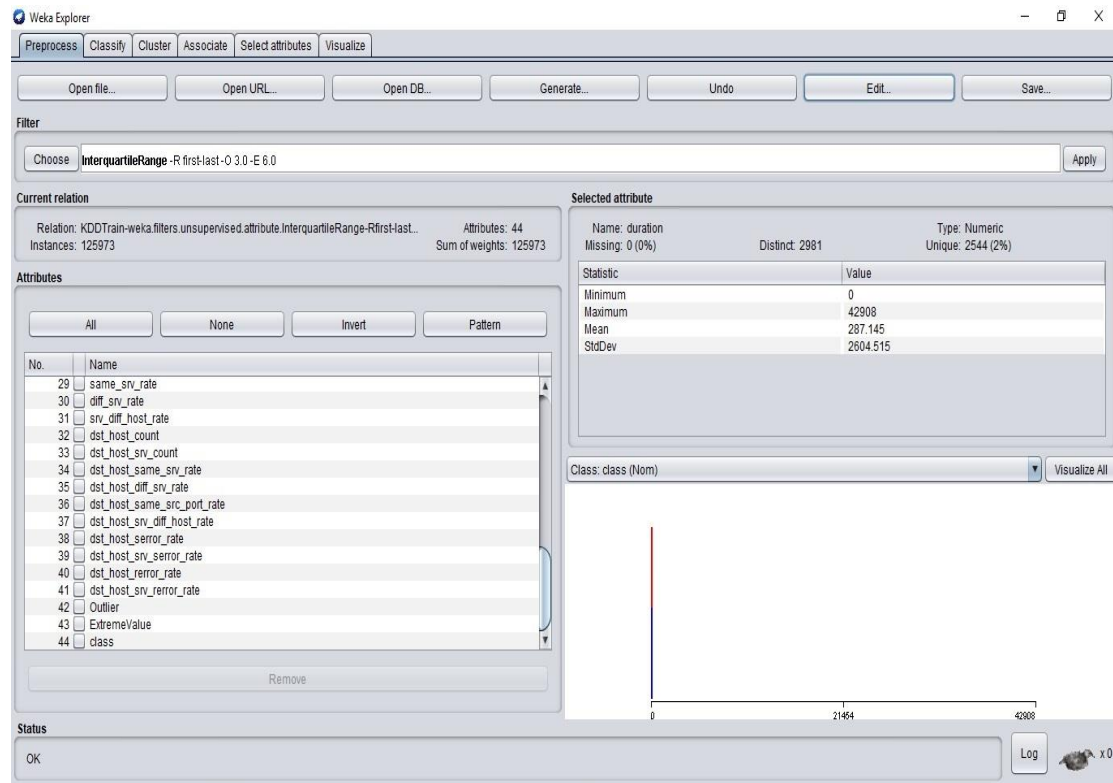


Figure 6. Filtered KDD-99 dataset

4.2 RESULTS IN WEKA ENVIRONMENT

To present the results in a way that can be easy to interpret, screenshots before and after the new method are provided and compared to each other.

The original KDD-99 dataset that is used contains 125.973 instances and 42 attributes and the altered, from the filter, KDD-99 contains 125.973 instances and 44 attributes. The four classifiers used for the experiments are Hoeffding Tree, Naïve Bayes, J48 and lazy IBk.

4.1.1 Hoeffding Tree

In the figure n. the results from the classification of the original dataset are shown.

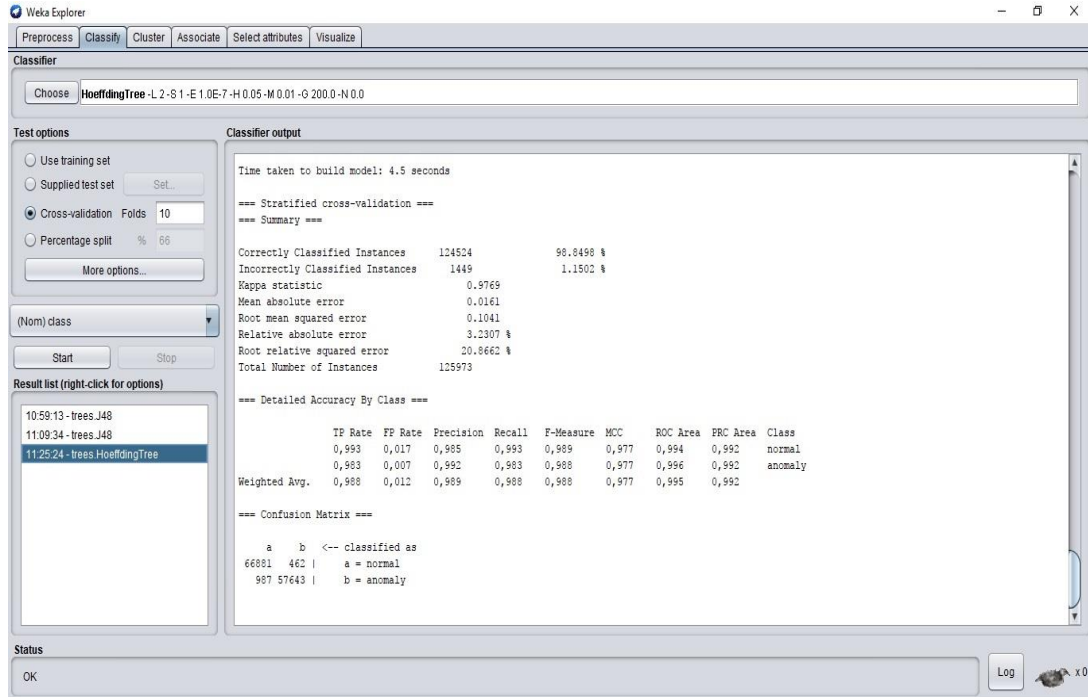


Figure 7. WEKA classification: Hoeffding Tree in original dataset

Following, the results from the classification of the altered dataset are shown.

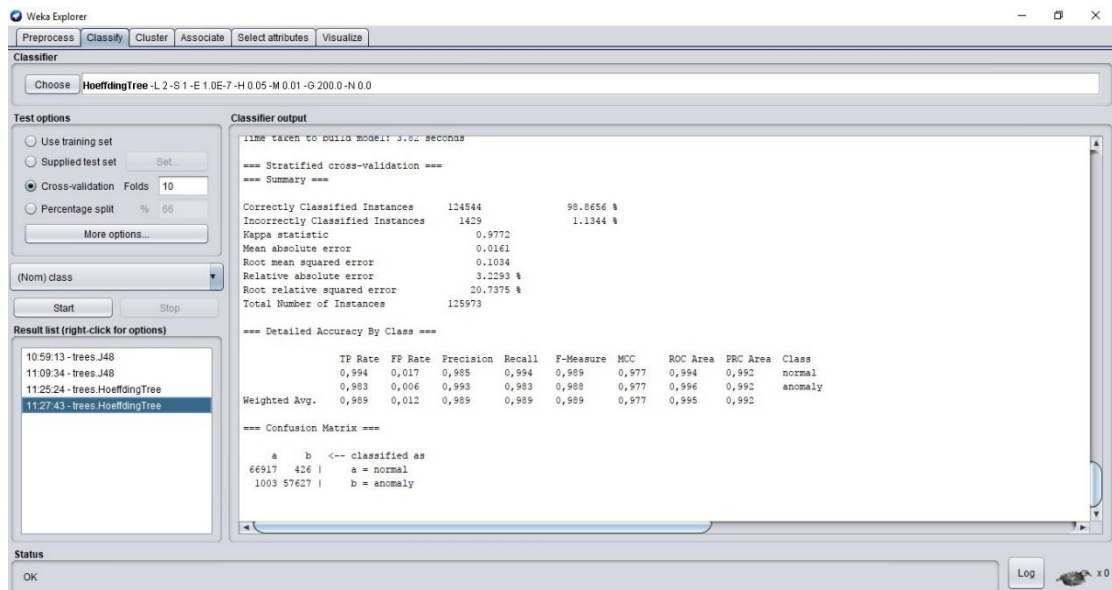


Figure 8. WEKA classification: Hoeffding Tree in dataset after the use of the IQR filter

To summarize the results, in order to compare them the table below is provided. It presents the general characteristics and the results in different metrics displayed in the figures. After the table the two confusion matrices of the results are also provided.

	Original Dataset	Filtered Dataset
Classifier	Hoeffding Tree	Hoeffding Tree
Test Mode Type	10 fold Cross-Validation	10 fold Cross-Validation
Time Needed to Build	4.5 seconds	3.82 seconds
Model		
Correctly Classified Instances	98.8498%	98.8656%
TP Rate (normal)	0.993	0.994
FP Rate (normal)	0.017	0.017
Precision	0.985	0.985
ROC Area	0.994	0.994

Table 3. Hoeffding Tree results comparison

```

=== Confusion Matrix ===
      a    b  <-- classified as
66881  462 |    a = normal
  987 57643 |    b = anomaly
  
```

Figure 9. Confusion Matrix of the original dataset

```

=== Confusion Matrix ===
      a    b  <-- classified as
66881  462 |    a = normal
  987 57643 |    b = anomaly
  
```

Figure 10. Confusion Matrix of the filtered dataset

From the above results it is clear that the accuracy of the classifier in the filtered dataset is better than the accuracy of the original dataset. Also there is a slight improvement in the true positive rate of the filtered dataset. One important detail is that the time taken for the first classification to be done in the original dataset lasted 4.5 seconds but in the filtered dataset the system adapted faster and built the model in 3.82 seconds. The rest of the results remain the same due to the

small size of the datasets. Last, as it is displayed, there is no difference in the confusion matrices of the two datasets.

4.1.2 Naïve Bayes

In the figure n. the results from the classification of the original dataset are shown.

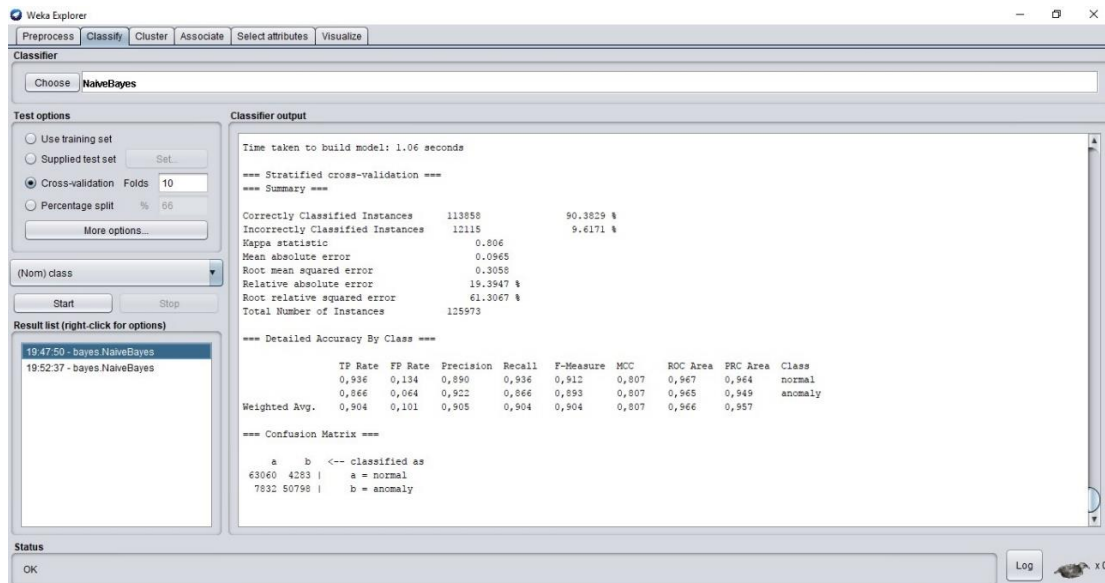


Figure 11. WEKA classification: Naïve Bayes in original dataset

Following, the results from the classification of the altered dataset are shown.

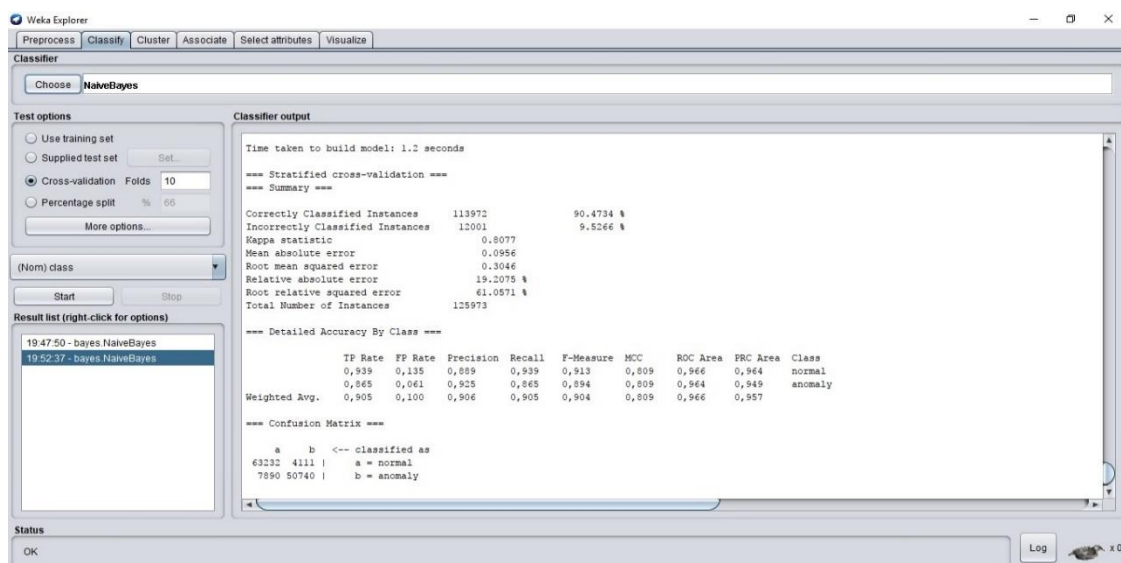


Figure 12. WEKA classification: Naïve Bayes in dataset after the use of the IQR filter

The following table summarizes the results above.

	Original Dataset	Filtered Dataset
Classifier	Naïve Bayes	Naïve Bayes
Test Mode Type	10 fold Cross-Validation	10 fold Cross-Validation
Time Needed to Build Model	1.6 seconds	1.2 seconds
Correctly Classified Instances	90.3829%	90.4734%
TP Rate (normal)	0.936	0.939
FP Rate (normal)	0.134	0.135
Precision	0.890	0.889
ROC Area	0.967	0.966

Table 4. Naïve Bayes results comparison

```

=== Confusion Matrix ===
      a    b  <-- classified as
63060  4283 |   a = normal
 7832 50798 |   b = anomaly
  
```

Figure 13. Confusion Matrix of the original dataset

```

=== Confusion Matrix ===
      a    b  <-- classified as
63232  4111 |   a = normal
 7890 50740 |   b = anomaly
  
```

Figure 14. Confusion Matrix of the filtered dataset

By observing the above results, there are several conclusions conducted. First, the time taken to build the filtered model has improved from 1.6 seconds needed for the original model, to 1.2 seconds. Second, the accuracy of the classifier has improved 0.0905 seconds. Third, the true positive rate, the false positive rate and the precision have also improved. The ROC Area rate though is not improving from the original to the filtered model. Last, the confusion matrix of the filtered model provides better results that the confusion matrix of the original model.

4.1.3 J48

In the figure n. the results from the classification of the original dataset are shown.

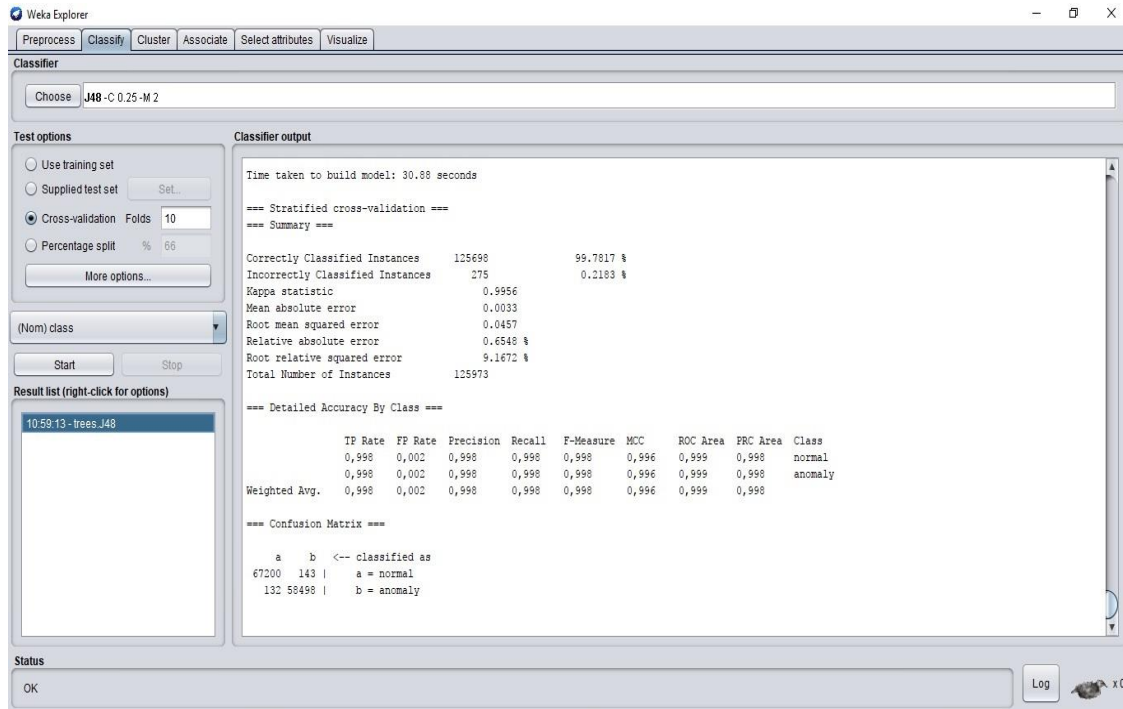


Figure 15. WEKA classification: J48 in original dataset

Following, the results from the classification of the filtered dataset are shown.

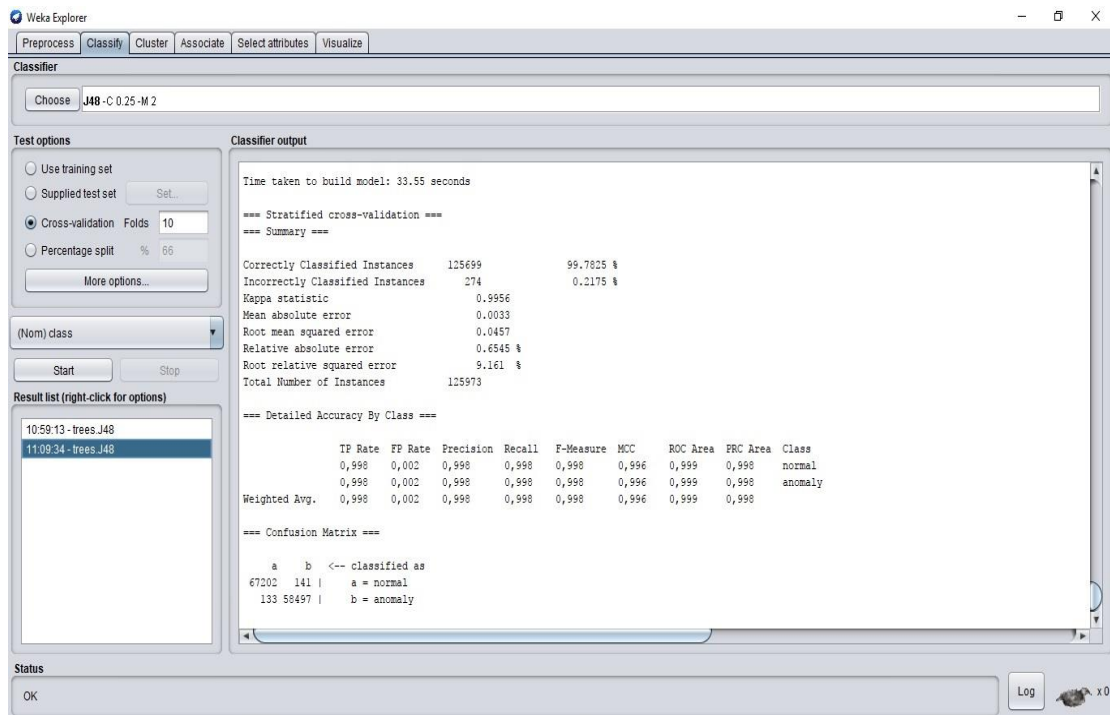


Figure 16. WEKA classification: J48 in dataset after the use of the IQR filter

The following table summarizes the results above.

	Original Dataset	Filtered Dataset
Classifier	J48	J48
Test Mode Type	10 fold Cross-Validation	10 fold Cross-Validation
Time Needed to Build Model	30.88 seconds	33.55 seconds
Correctly Classified Instances	99.7817%	99.7825%
TP Rate (normal)	0.998	0.998
FP Rate (normal)	0.002	0.002
Precision	0.998	0.998
ROC Area	0.999	0.999

Table 5. J48 results comparison

```

=== Confusion Matrix ===
      a    b  <-- classified as
67200  143 |   a = normal
 132 58498 |   b = anomaly
  
```

Figure 17. Confusion Matrix of the original dataset

```

=== Confusion Matrix ===
      a    b  <-- classified as
67202  141 |   a = normal
 133 58497 |   b = anomaly
  
```

Figure 18. Confusion Matrix of the filtered dataset

In the case of J48 the results in the two models do not differentiate in matters of true positive rate, false positive rate, precision and ROC Area. That is because the accuracy of the J48 classifier is so high that the difference is not visible in such a small dataset. Although, by looking at the percentage of accuracy, again the filtered dataset has higher accuracy than the original one. The odd thing in this case is that the classifier needed more time to build the filtered model compared to the original one. In the confusion matrices though, in the second model the results are better than the original one.

4.1.3 lazy IBk

In the figure n. the results from the classification of the original dataset are shown.

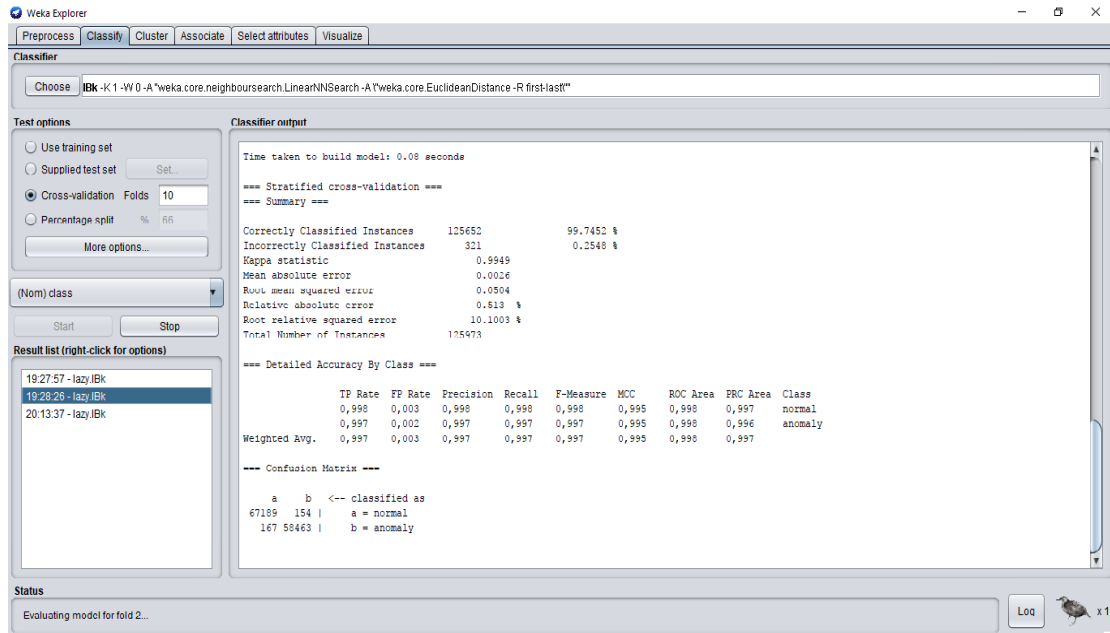


Figure 19. WEKA classification: lazy IBk in original dataset

Following, the results from the classification of the filtered dataset are shown.

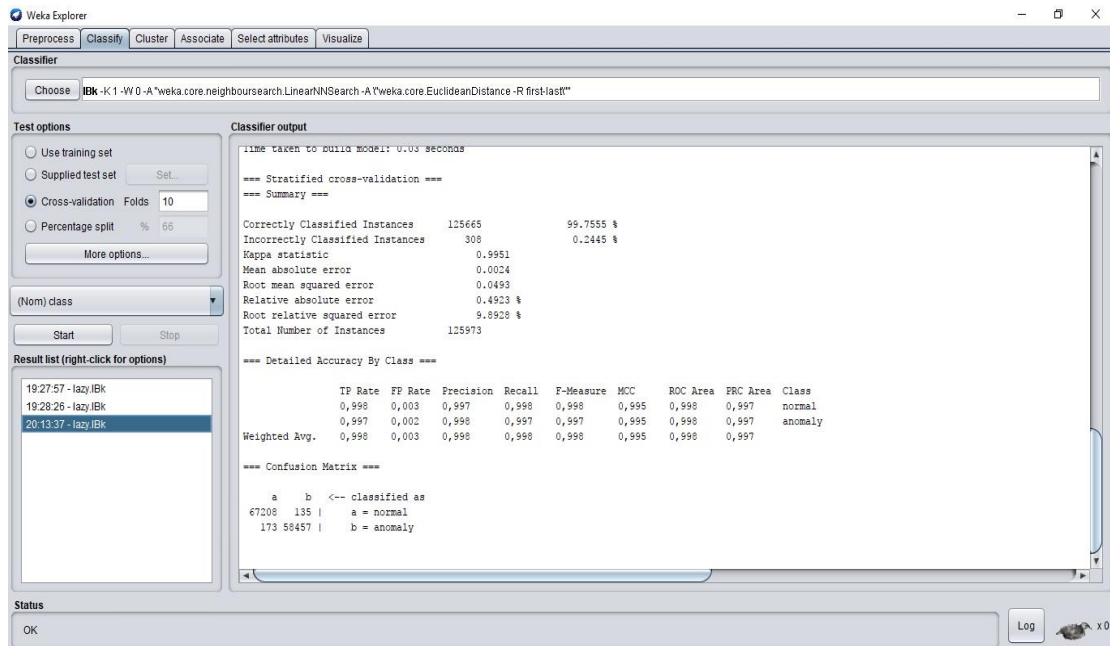


Figure 20. WEKA classification: lazy IBk in dataset after the use of the IQR filter

The following table summarizes the results above.

	Original Dataset	Filtered Dataset
Classifier	Lazy IBk	Lazy IBk
Test Mode Type	10 fold Cross-Validation	10 fold Cross-Validation
Time Needed to Build Model	0.08 seconds	0.03 seconds
Correctly Classified Instances	99.7452%	99.7555%
TP Rate (normal)	0.998	0.998
FP Rate (normal)	0.003	0.002
Precision	0.998	0.997
ROC Area	0.998	0.998

Table 5. lazy IBk results comparison

```

--- Confusion Matrix ---
      a      b  <-- classified as
67189  154 |   a = normal
 167 58463 |   b = anomaly
  
```

Figure 21. Confusion Matrix of the original dataset

```

=== Confusion Matrix ===
      a      b  <-- classified as
67208  135 |   a = normal
 173 58457 |   b = anomaly
  
```

Figure 22. Confusion Matrix of the filtered dataset

By observing the above results, it is clearly declared that with the filtered dataset the classifier needs less time to build the new model and the accuracy also improves. The remaining metrics are the same in the two datasets except the precision, which in this case has a minor decrease. The confusion matrix of the filter dataset again provides better results that the confusion matrix of the original dataset.

4.3 RESULTS IN MOA ENVIRONMENT

MOA framework has a special option for concept drift. It allows the evaluation of a dataset that contains drift concepts. In the following two figures, the evaluation of the original and the filtered dataset is made, accordingly.

By observing this first figure, the first thing that is stated are the black lines in the plot. Those black lines are representing the concept drift in the dataset. In this case, the evaluator has found many points of drift among the instances and as it is obvious in the plot the lines exist every time the red line does a gradual or sudden change.

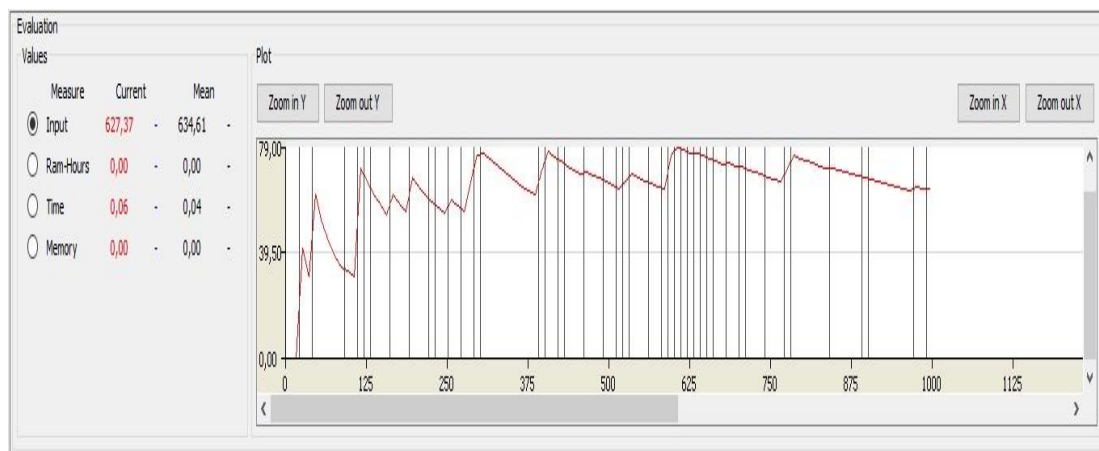


Figure 23. MOA environment: Original Dataset

By observing the second figure, an instant difference from the first one is appeared. After the point of 750 in the x axis the black lines that represent concept drift are becoming dense. This is happening, due to the fact, that with the new filtered dataset the evaluator is more confident compared to the previous case and it detects the points of concept drift easier than before.

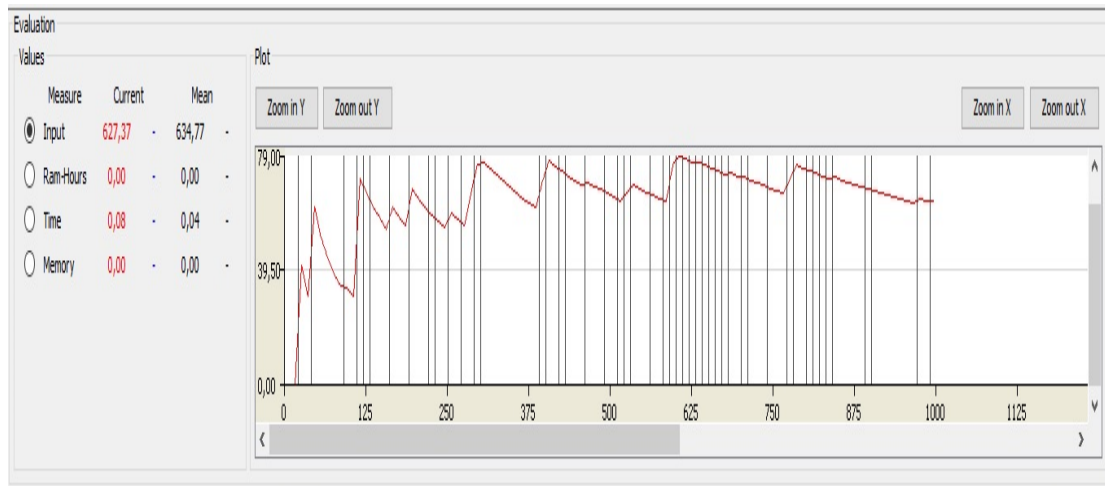


Figure 24. MOA environment: Filtered Dataset

As all the results have been presented there is one conclusion that can be derived. With the use of the Interquartile Range filter on the KDD-99 dataset the classifiers become more accurate and they adapt faster during the classification process. Also, the rate of the true positives increase in general. The time taken to build the model in most cases decreases and that means that the classifier is getting faster. Last but not least, the concept drift evaluator after the application of the filter becomes more confident and find the drifting concepts easier.

CHAPTER 5

CONCLUSIONS

In this dissertation, the world of data mining and machine learning applications combined with intrusion detection systems was explored. Based on this examination, the connection with the phenomenon of concept drift became more than obvious along with the problems that are generated. In pursuance of a solution to this problem, a new method was proposed, tested and evaluated.

Based on the new method, the application of filters in the preprocessing part of machine learning before training the new model, proved to be more efficient, effective and helped the system to adapt quicker according to the results acquired.

The new method using the Interquartile Range filter, rearranges the dataset in a way that helps the machine learning algorithms to detect concept drift faster and improve the accuracy of the new model, although it not always eliminate it. Those optimal results were obtained in all the algorithms that were tested.

Naturally, in the development of the new method, limitations existed as well. The fact that WEKA could not support a bigger dataset narrowed down the analysis of the data experiencing concept drift. As a result, in the evaluation of the experiments the deviation before the application of the new method and after the application is not so large but it is, no matter those limitations, improved.

One challenge that will be worth tested in the future is the combination of the new method with the process of feature selection in order to find out if the concept drift phenomenon is related in a higher degree with specific features. This suggestion could lead to even more optimized results if it is confirmed that there are specific features that extend the effect of concept drift in the data explored.

Finally, this dissertation, lead the way to research, both in theory and in practice, a beneficial way to cope with machine learning problems related to time and also gain new insights and skills in a developing, but most important promising field of computer science.

REFERENCES

- [1] Yi Wu. (2014), *Network Big Data: A Literature Survey on Stream Data Mining*. JOURNAL OF SOFTWARE, VOL. 9, NO. 9
- [2] Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2013), *A Survey on Concept Drift Adaptation*. ACM Comput. Surv. 1, 1, Article 1, 35 pages
- [3] Zliobaite I. and Pechenizkiy M., *Handling Concept Drift in Information Systems*
- [4] Zliobaite, I., Pechenizkiy, M., Gama, J., (2015), *An overview of concept drift applications*
- [5] Sumeet Dua and Xian Du, (2011) *Data Mining and Machine Learning in Cybersecurity*
- [6] Jiawei Han, Micheline Kamber, Jian Pei, (2012), *Data Mining Concepts and Techniques*, 3rd edition
- [7] G.V. Nadiammai, Hemalatha, M., (2014), *Effective approach toward Intrusion Detection System using data mining techniques*
- [8] Paul Dokas, Levent Ertoz, Vipin Kumar, Aleksandar Lazarevic, Jaideep Srivastava, Pang-Ning Tan, (2002), *Data Mining for Network Intrusion Detection*, Computer Science Department, University of Minnesota
- [9] Dariusz Brzezinski, (2013), *Reacting to different types of concept drift: The accuracy updated ensemble algorithm*, Poznan University of Technology, Poland
- [10] Brown, M. (2012), <https://www.ibm.com/developerworks/library/ba-data-mining-techniques/index.html>
- [11] Oracle,
https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/classify.htm#DMCON004, Oracle Data Mining Concepts, Classification
- [12] Oracle,
https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/regress.htm#CHDBHBDI, Oracle Data Mining Concepts, Regression
- [13] Oracle,
https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/market_basket.htm#DMCON009, Oracle Data Mining Concepts, Association
- [14] Oracle,
https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/clustering.htm#DMCON008, Oracle Data Mining Concepts, Clustering
- [15] Berkhin, P., (2002), *Survey of Clustering Data Mining techniques*, Accrue Software, Inc.

- [16] Yihua Liao, V. Rao Vemuri, (2002), *Use of K-Nearest Neighbor classifier for intrusion detection*, Department of Computer Science University of California
- [17] Witten, I., Frank, E., Hall, M., (2011), *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd Edition
- [18] Oracle,
https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_nb.htm#BABIIDDE, Oracle Data Mining Concepts, Naïve Bayes
- [19] Rish, I., (2001), *An empirical study of the naive Bayes classifier*, T.J. Watson Research Center
- [20] Lior Rokach, Oded Maimon, (2014), *Data Mining with Decision Trees: Theory and Applications*, 2nd Edition, Department of Industrial Engineering Tel-Aviv University
- [21] Oracle,
https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_decisiontree.htm#DMCON019, Oracle Data Mining Concepts, Decision Tree
- [22] Mohammad M. Masud¹, Jing Gao², Latifur Khan¹, Jiawei Han², and Bhavani Thuraisingham, (2009), *Integrating Novel Class Detection with Classification for Concept-Drifting Data Streams*, University of Texas, Dallas, University of Illinois
- [23] Baena-Garcia, M., del Campo-Avila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno R., (2006), *Early Drift Detection Method*, Universidad de Malaga, Universitat Politecnica de Catalunya
- [24] Kolter, J., Mallof, M., (2007), *Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts*, Department of Computer Science Stanford University, Department of Computer Science Georgetown University
- [25] Ortiz, D.A., (2015), *Fast adapting ensemble: a new algorithm for mining data streams with concept drift*
- [26] WEKA framework, <https://www.cs.waikato.ac.nz/ml/weka/>, University of Waikato
- [27] MOA framework, <https://moa.cms.waikato.ac.nz/>, University of Waikato
- [28] Eclipse, <http://www.eclipse.org/>
- [29] KDD-99, <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

APPENDIX

Source Code for WEKA for saving and loading the dataset.

```
package weka.api;

import weka.core.Instances;
import weka.core.converters.ArffSaver;
import java.io.File;

import weka.core.converters.ConverterUtils.DataSource;
public class KddLoadSave{
    public static void main(String args[]) throws Exception{
        DataSource kddsrc = new DataSource("/Desktop/kdd99.arff");
        Instances kddtst = kddsrc.getDataSet();

        System.out.println(kddtst.toSummaryString());

        ArffSaver kddsvr = new ArffSaver();
        kddsvr.setInstances(kddtst);
        kddsvr.setFile(new File("/Desktop/newkdd99.arff"));
        kddsvr.writeBatch();
    }
}
```