Habibu Shoamari Mukhandi **Developing Machine Learning Methods for Network Anomaly Detection** AGU 2018

# Developing Machine Learning Methods for Network Anomaly Detection

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF NATURAL SCIENCES OF ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER

By
Habibu Shomari Mukhandi
July 2018

# Developing Machine Learning Methods for Network Anomaly Detection

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF NATURAL SCIENCES OF ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER

By
Habibu Shomari Mukhandi
July 2018

## SCIENTIFIC ETHICS COMPLIANCE

I hereby declare that all information in this document has been obtained in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Habibu Shomari Mukhandi

## REGULATORY COMPLIANCE

**M.Sc. thesis titled "Developing Machine Learning Methods for Network Anomaly Detection"** has been prepared in accordance with the Thesis Writing Guidelines of the Abdullah Gül University, Graduate School of Engineering & Science.

| | |
|---|---|
| Prepared By | Advisor |
| Habibu Shomari Mukhandi | Dr. Zafer Aydın |

Head of the Electrical and Computer Engineering Program

Assoc. Prof. Vehbi Çağrı GÜNGÖR

## ACCEPTANCE AND APPROVAL

M.Sc. thesis titled Developing Machine Learning Methods for Network Anomaly Detection and prepared by Habibu Shomari Mukhandi has been accepted by the jury in the Electrical and Computer Engineering Graduate Program at Abdullah Gül University, Graduate School of Engineering & Science.

……….. /……….. / ………..

(Thesis Defense Exam Date)

**JURY:**

| | | |
|---|---|---|
| Dr. Zafer Aydın | :……………….……….. | signature |
| Assoc. Prof. Çagrı Güngör | :………………….............. | signature |
| Dr. Mete Çelik | :…………………........................ | signature |
| Member | :……………………………………. | signature |
| Member | :……………………………….… | signature |

**APPROVAL:**

The acceptance of this M.Sc. thesis has been approved by the decision of the Abdullah Gül University, Graduate School of Engineering & Science, Executive Board dated ….. /….. / ……….. and numbered ……………..……. .

……….. /……….. / ………..

**(Date)**

Graduate School Dean
Name-Surname, Signature

## ABSTRACT

# Developing Machine Learning Methods for Network Anomaly Detection

Habibu Shomari MUKHANDI

M.Sc. in Electrical and Computer Engineering Department

**Supervisor:** Dr. Zafer Aydın

July-2018

Machine learning refers to training of a computer (machine) to be able to acquire knowledge from data (i.e. experience) and improve itself on a given task. The field of machine learning has become a mainstream, improving hundreds of millions of lives. Fraudulent actions in computer networks, credit card transactions and website advertisement traffic might devastate large businesses and cause anually fiscal loss of billions of dollars around the globe. In this thesis, we propose various machine learning methods for three fraud detection problems: network anomaly detection, credit card fraud detection and detection of fraudulent clicks to advertisements on the internet. We design various classifiers such as logistic regression, k-nearest neighbors, decision tree, support vector machine, and ensemble classifiers such as random forest, bagging, stacking and AdaBoost. The hyper-parameters of the classifiers are optimized by performing cross-validation experiments on train sets. In the next step, the models are trained using the optimum hyper-parameter configurations and predictions are computed on test sets. Among the various methods compared the highest accuracy is obtained by ensemble learners.

*Keywords: Anomaly Detection, Fraud Detection, Network Anomaly Detection, Credit Card Fraud Detection, Fraud Detection for Advertisement Clicks, Machine Learning, Ensemble Classifiers*

## ÖZET

# BİLGİSAYAR AĞLARINDA ANORMAL DURUM TESPİTİ YAPAN ÖĞRENME YÖNTEMLERİNİN GELİŞTİRİLMESİ

Habibu Shomari Mukhandi

Elektrik ve Bilgisayar Mühendisliği Bölümü Yüksek Lisans

**Tez Yöneticisi:** Dr. Zafer Aydın

Temmuz-2018

Makine öğrenmesi, verilerdeki bilginin bir bilgisayar ya da makina tarafından otomatik olarak öğrenilmesi ve karşılaşılan yeni durumlarda anlamlı bilgi ya da davranışların üretilmesini amaçlar. Bir çok uygulama alanı bulunan makine öğrenmesi daha önce hiç karşılaşılmamış olan sıradışı durumların tespit edilmesi için de kullanılmaktadır. Bilgisayar ağlarındaki siber saldırılar, kredi kartı dolandırıcılığı ve internet sitelerinin linklerine yapılan çok sayıda sahte tıklamalar dünya genelinde ekonomileri ciddi oranda zarara uğratabilecek niteliktedir. Bu tezde üç farklı anormal durum tespiti problemi üzerinde çalışılmıştır: bilgisayar ağlarında saldırı tespiti, kredi kartı dolandırıcılığı tespiti ve internet sitelerdeki linklere sahte tıklama tespiti. Anormal durum tespiti için geliştirilen ve optimize edilen modeller arasında rastgele orman, en yakın komşu, destek vektör makinası, logistic regresyon, karar ağacı, AdaBoost, çantalama ve yığınlama gibi sınıflandırma yöntemleri bulunmaktadır. Yöntemlerin hiper-parametreleri eğitim kümelerinde yapılan çapraz doğrulama deneyleri ile optimize edilmiştir. Bir sonraki aşamada optimum hiper-parametre konfigürasyonları kullanılarak eğitilen modeler ile test verilerinde tahmin sonuçları hesaplanmıştır. Bu deneyler neticesinde genel doğruluk oranı ve F-measure skorlarında yüksek başarı elde edilmiştir. Geliştirilen yöntemler arasında en başarılı sonuçlar topluluk modelleri ile elde edilmiştir.

*Keywords: Anormal Durum Tespiti, Dolandırıcılık Tespiti, Bilgisayar Ağlarında Anormal Durum Tespiti, Kredi Kartı Dolandırıcılığı Tespiti, İnternet Sitelerinde Sahte Tıklama Tespiti*

# Acknowledgements

I want to thank Dr. Zafer Aydin for believing in me and decided to be my supervisor through out my master's degree. His guidance made it possible for me to conduct tireless and fruitful research on different Machine learning algorithms on various areas. I would like to thank you for your support, dedication and availability were extraordinary. Definately a fantastic person.

To Assoc. Prof V. Cagri Gungor who help with providing data for anomalies detection from Labris Networks which helped immeasurabely for training and testing different algorithms.

Finally, a special thanks to my parents who have always supported me in all my decisions and always encouraged me to be the best version of myself. For being models of commitment and courage I dedicate this work to them. Also, to my two sisters and my two brothers, I am so lucky to have you all.

# Table of Contents

# List of Figures

# List of Tables

*This thesis is dedicated to my family*

# Chapter 1

# Introduction

As human population continues to grow and technology gets ubiquitous and more affordable, so does data about them. Ranging from business to many other fields including entertainment and medicine there is a need to analyze this large amounts of data to understand the needs and demands of people and improve the services provided. Furthermore, the huge growth of computer networks and their accessibility via variety of devices including PCs and mobile devices have increased the number of applications running on computer network platforms especially the internet. As a result of this the security of computer networks has gained considerable importance. According to Landwehr et al. [1], all computer systems are victims of security vulnerabilites which are both economically costly and technically difficult to resolve by the companies that produce computer systems. source of vulnerability is the fraudulent actions such as fraudulent ad clicks that Another might reduce profits and cause unnecessary load on the network. To address these issues, in this thesis, three anomaly detection problems are studied: network anomaly detection, credit card fraud detection and ad tracking fraud detection.

## 1.1 Problems

### 1.1.1 Network Anomaly Detection

The first problem we studied is network anomaly detection for which we developed an Intrusion Detection System (IDS) that detects anomalies and attacks in a computer network. Computer intrusion is a collection of activities or actions that violate the security of a computer system [2]. The IDS typically contains a predictive model (i.e. a classifier) that has the capability of differentiating attacks (i.e. anomalies oroutliers) from normal samples. It can be limited to learn a two-class problem or a multi-class problem as in KDDCup99 dataset. The two-class problem is based on discriminating attacks from normal patterns while the multi-class problem deals with the classification of different types of attacks that occur in a network and those that do not pose any threat to the network (i.e. normal packets). In all cases, the goal is to develop a classifier that will make accurate predictions on unseen instances (e.g. test data).

Anomaly detection is an essential problem in intrusion detection systems. An anomaly is an object (an instance or an example) that deviates significantly from the rest of the objects, as if it were generated by a different distribution [3]. In network anomaly detection we try to find an object(s) (network connection(s)) that deviates from normal connection behaviour. These anomalies on a network are also called as attacks [4], which is defined as a series of operations that endagers and risks the security of a system and an anomaly is an event or an occurance that raises suspicion from the perspective of security. A network anomaly detection system can be classified as anomaly-based or signature-based. Based on the way these two categories view attack and anomaly, each has its own advantages and disadvantages. According to Jyothsna et al. [5], a signature-based system looks for signatures or patterns of provided data and an anomaly-based system tries to estimate or learn the normal characteristics of the system and generates an anomaly warning sign whenever the deviation between a new observed instance and the normal behavior surpasses a chosen threshold. An anomaly-based system may also raise a warning when the difference between the new observed instance and the expected one falls below a given limit. A signature-based system can detect specified and well-known attacks well but does not have the ability to detect new and unfamiliar intrusions. On the other hand, an anomaly-based detection system can detect previous unseen attacks and intrusions but such systems the observed number of false positives (i.e events classified as attacks while they are not) is higher than in signature-based systems. In the litereature, the more explored category is the anomaly-based detection system.

Different mechanisms and approaches are developed for the problem of network anomaly detection including: stastical approaches, knowledge-based systems whereby a human expert on the field analizes each individual connection to determine if it is an attack or not, and machine learning approaches, which include Bayesian networks [6], genetic algorithms [7], neural networks [8], Markov models, decision trees [9], Adaboost [10], Multiboost [11], Bagging [3], support vector machines (SVM) [8] and other supervised and unsupervised algorithms. According to Chandola et al. [12], many previous research does not employ similarity and distance meausures to determine the difference between a target (a newly observed instance) and the known type. According to the authors, more future research will explore this area even further to improve anomaly detection systems. The distance measure suggested are such as Kullback-Leibler distance to rank features, the entropy measure, and measures for similarity such as the cosine similarity.

#### 1.1.1.1 Labris Dataset

The first network anomaly detection data set we used in this thesis contains binary classes, in which 90% of the data set is labeled as normal and the remaining 10% as attack. There is also a second class attribute called Attack Type, which is used for multi-class classification. In total, there are 42 attributes and 9 attack types (i.e. classes including the normal). The types of classes are: (1) Syn ack ddos, a distributed Denial of Service attack that exploits normal TCP in the form of a three way handshake, (2) icmp ddos, also known as ping flood, which tries to put servers out of service by request for its response more than the ability of the server, (3) rst ack ddos, also known as three-way handshake reset attack, which is used for sending a reseting bit ag to stop a TCP connection, (4) rst ddos, which is also a reset attack but with no handshake, (5) fin ddos, which sends a massive amount of TCP packets with fin bit enabled which makes the server busy dropping the incoming packets, (6) ack ddos, which is a distributed Denial of Service with three way handshake, (7) http get, which is introduced for attacking web servers, (8) syn ddos, and (9) normal, which is not considered an attack.

| Class | No.of Instances | Instances in Percentage |
|---|---|---|
| Normal | 62124 | 91% |
| Syn_ack_ddos | 562 | 0.80% |
| Icmp_ddos | 17 | 0.024% |
| Rst_ack_ddos | 1384 | 2% |
| Rst_ddos | 920 | 1.46% |

| | | |
|---|---|---|
| fin_ddos | 9 | 0.131% |
| ack_ddos | 404 | 0.641% |
| http_get | 1477 | 2.16% |
| Syn_ddos | 1525 | 2.17% |

Table 1.1.1.1.1: Labris network anomaly detection dataset

#### 1.1.1.2 KDDCup99 Dataset (Intrusion Detection Learning)

Intrusion is defined as an intentional illegal trial to access data, manipulate it, or compromise a computer system's veracity and or make its data unusable [13]. An intruder may be from outside (i.e. a hacker) and an insider who knows the how the system is designed (e.g. IT manager and system administrator). Sometimes, a company may hire a penetration testing expert to behave like an intruder so that he/she can point out and fix the holes (bugs and errors in the system) before a person with malicious intent attempts to compromise it. Computer intrusion can be categorized into two types: misuse intrusions and anomaly intrusions. Misuse intrusions are attacks on known weak points of a computer system. Anomaly intrusions are based on observations of instances that deviate from normal usage patterns. These include trying to break-in, hidden attacks disguised as normal usages, denial of service, and malicious use of the system.

In KDD99 cup competition, the challenge was to develop a network intrusion detection system (IDS), which is a classifier that can differentiate malicious connections from the normal legitimate ones. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. The KDDCup99 dataset is the benchmark that contains a wide range of attacks that mimic real world intrusions in an imitated real military network environment [14]. In 1998, MIT Lincoln Lab prepared and managed the experiment to be able to simulate a wide range of intrusion types and study them and build machine learning models that could learn and protect the systems and environments from real world threats. Lincoln Labs set up an environment for nine weeks to gather raw TCP connections for a local area network (LAN). LAN was set to mimic a real world US Air Force LAN. The labs treated the set up LAN as if it were a true Air Force environment but they perpertrated it with different attacks imaginable that a hacker may conduct. The labs gathered about four gigabytes of data for model training from the first seven weeks. The training set was later processed into about five million connection records. The remaining two weeks were taken as test data which produced about two million connection records.

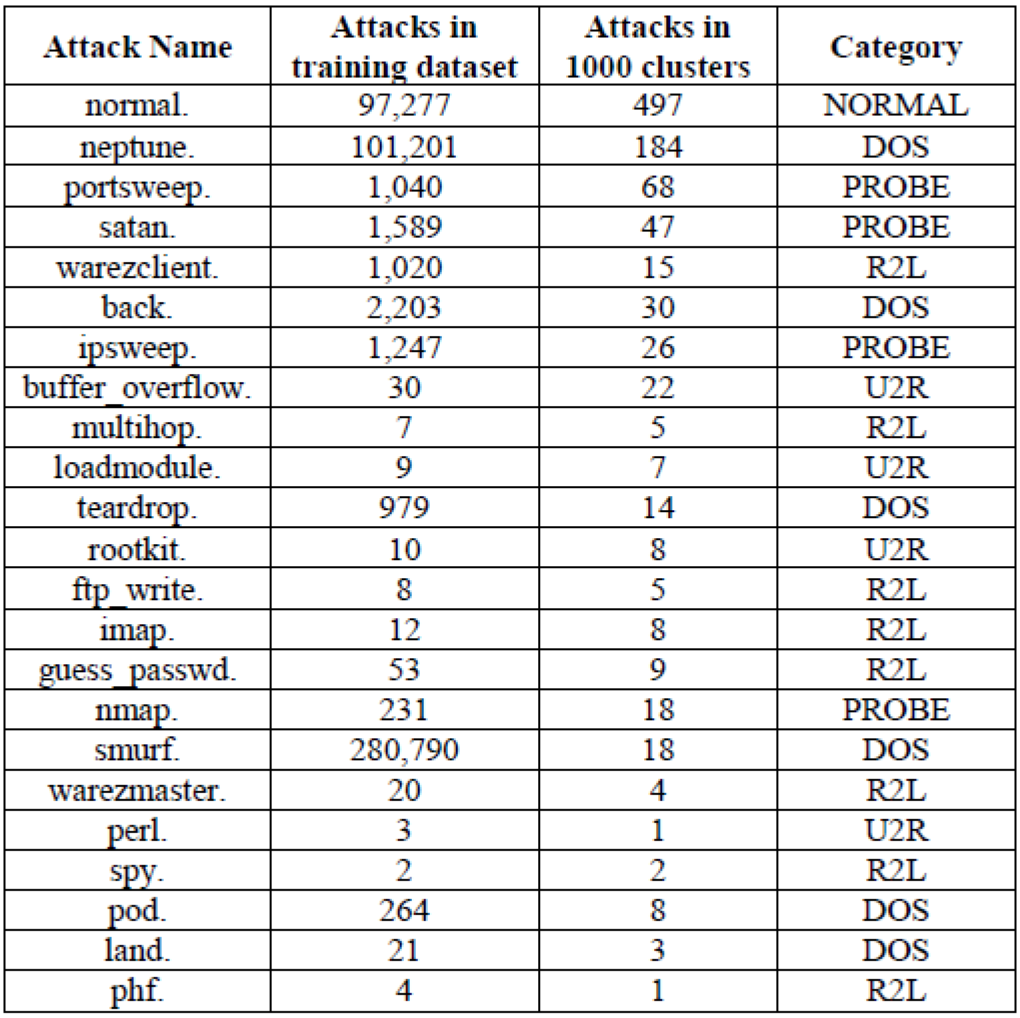
The KDD training dataset used by Hormozi et al. [15] contains 10% of the whole dataset which is 494,020 single connection instances each having 41 features labelled as normal or attack and if it is an attack what kind of attack it is. A connection that deviates from the normal pattern is considered as an attack or a malicious connection [15]. A 10% portion is chosen due to memory sensitive algorithms, approximately 80% of which contains attacks and the remaining 20% are normal connections.

Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. In this experiment every connection record was of size 100 bytes. Furthermore, the recorded attacks were of 18 types which were categorized into four categories, namely: DOS: denial-of-service, e.g. syn ood; R2L: an illegal attempt of accessing from a remote computer, e.g. password; U2R: an illegal access to a systems administator (or root) privileges, e.g., different types of "buffer overflow" attacks; probing: surveillance and other probing, e.g., penetration testing through port scanning. For a test data to be effective for testing it is essential that it comes from or generated by a different probability distribution. There may be some attacks not in the train set. This way the experiment tries to implement a real world scenario. Several intrusion experts claim that most not yet seen attacks are forms and modifications of known and already seen attacks and the known patterns of known attacks can be sufficient to detect novel variants. The datasets features 24 attack types in the training set, and there is an additional 14 new unseen attacks in the test set alone not included in the training set. The attack types of the dataset are listed in Table 2.1.1.2.1.

## Previous works

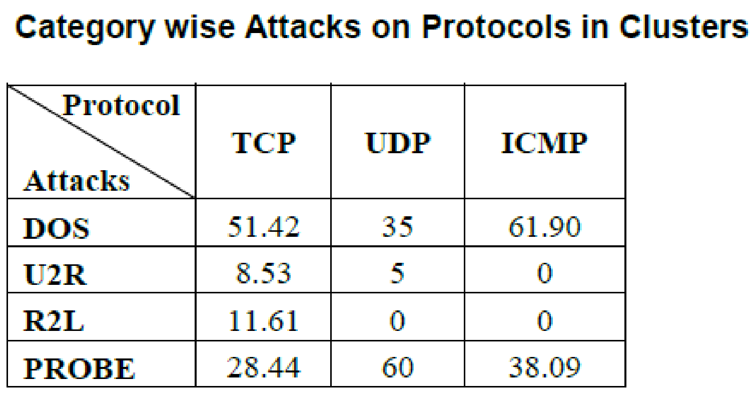
The authors, Salazar et al. [16], have conducted analysis and experiments of attacks on the 10% portion of KDDCup 99 training set using Kmeans clustering technique. They clustered the training set, which contains of 494,019 instances and prepared 1000 clusters [16]. The clustering is used to determine and rank protocols according to number of attacks.

The results are as follows:



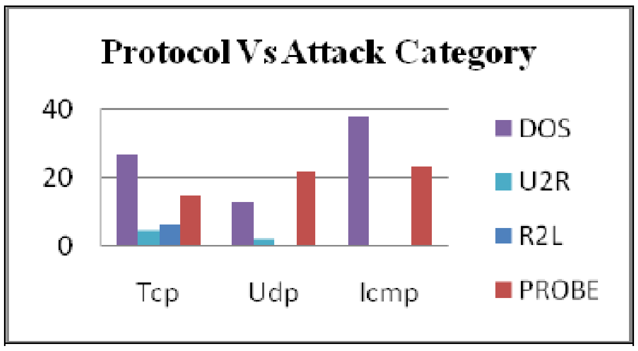
| Attack Name | Attacks in training dataset | Attacks in 1000 clusters | Category |
|---|---|---|---|
| normal. | 97,277 | 497 | NORMAL |
| neptune. | 101,201 | 184 | DOS |
| portsweep. | 1,040 | 68 | PROBE |
| satan. | 1,589 | 47 | PROBE |
| warezclient. | 1,020 | 15 | R2L |
| back. | 2,203 | 30 | DOS |
| ipsweep. | 1,247 | 26 | PROBE |
| buffer_overflow. | 30 | 22 | U2R |
| multihop. | 7 | 5 | R2L |
| loadmodule. | 9 | 7 | U2R |
| teardrop. | 979 | 14 | DOS |
| rootkit. | 10 | 8 | U2R |
| ftp_write. | 8 | 5 | R2L |
| imap. | 12 | 8 | R2L |
| guess_passwd. | 53 | 9 | R2L |
| nmap. | 231 | 18 | PROBE |
| smurf. | 280,790 | 18 | DOS |
| warezmaster. | 20 | 4 | R2L |
| perl. | 3 | 1 | U2R |
| spy. | 2 | 2 | R2L |
| pod. | 264 | 8 | DOS |
| land. | 21 | 3 | DOS |
| phf. | 4 | 1 | R2L |

Table 1.1.1.2.1: Number of Attacks Before and After Clustering [16]



Category wise Attacks on Protocols in Clusters

| Protocol / Attacks | TCP | UDP | ICMP |
|---|---|---|---|
| DOS | 51.42 | 35 | 61.90 |
| U2R | 8.53 | 5 | 0 |
| R2L | 11.61 | 0 | 0 |
| PROBE | 28.44 | 60 | 38.09 |



Statistical View of Category Wise Attack on Protocol Types

Figure 1.1.1.2.1: Results on KDD99 using 100 clusters [16].

In the paper by Pang-Ning Tan et al. [17] first redudant records are removed which may cause the learning algorithms to be inclined and favour the more frequent records. Their solution was to first remove the redudant records and their results are shown in Figure 1.1.2.2.1. J. Kevric et al. [18] achieved an accuracy of 89.24% with hybrid of Random Tree and NB tree [19] on KDD99 cup test set. In another study by Jaswal et al. [20], the authors uses a hybrid technique by using K means algorithm to remove noise in the data set and remove duplicates and form an input for SVM. The authors claim this hybrid approach was able to detect all the attacks in their database and perfomed better in KDDCup 99 in which 22 attacks were detected compared to the benchmark of 11 attacks from previous research works. KDDTest is the original KDDCup test set. After removing redundancy, Tavallaee et al [21] generated KDDTest+ and KDDTrain+ which include 22,544 and 125,973 instances, respectively.

**Recent works with deep learning methods**

Some of the previous works used deep learning techniques on the KDDCup99 data sets. The following paragraph summarize the methods used by different authors and their reported accuracies. Aygun et al. [22] used two hybrid approaches. One was to use an autoencoder in which it has one input layer and one or more hidden layers (the authors used one hidden layer for this experiment) and one output layer. The number of input units and output units are the same but the number of hidden units is less than the number of units in input and output layers. The second hybrid approach used is called denoising autoencoder where the input is more distorted than in the normal autoencoder hybrid technique. The authors started with data preprocessing by taking 38 numeric features of the data set and converting them to binary values by using 1-N encoding approach. The autoencoders obtained an accuracy of 88.28% on KDDTest+ and for the denoising autoencoder the accuracy was 88.65%. According to Vinayakumar et al. [23], the performance of SVM on the binary classification (i.e a connection is normal or an attack) was better than their multi layer perceptron (MLP) and deep belief network (DBN). However, in multi-class classification problem their DBN setting with 4 layers perfomed the best. Their DBN included 350 neurons in each layer and the training was run till 1000 epochs with a fixed learning rate of 0.1. When the authors used 500 epochs, most of the connections were still classified as normal even though some were attacks. The accuracy of DBN4 was 71.5%, with a true positive rate of 99.99% and a false positive rate of 0.3% for normal connections and a true positive rate 92.8% and false positive rate of 5.9% on DOS connections. When they increased the number of layers to 8, the accuracy and the true positive rate increased but false positive rate increased too. Furthemore, when the number of layers was increased to 12 layers the accuracy decreased to 38.1%. Shone et al. [24] used a technique called non-symmetric deep autoenconder (NDAE), which is a novel approach proposed by the authors. NDAE differs from the conventional auto encoders by not having an encoder-decoder paradigm only multi hidden layers of encoders (i.e. non symmetric) to facilitate unsupervised feature selection and by removing a human expert control which can be prone to errors. The authors claim NDAE reduces both computational and time overheads with little cost on accuracy. The authors went a step further by stacking many NDAEs which allows a machine to learn complex relationships among features. The NDAE is also a powerful feature extractor algorithm. However, when the stacking of NDAEs was trained with a softmax layer as in other deep learning algorithms, the results were worse than the results of shallow classifiers such as SVM, RF, and k-NN. Therefore, the authors replaced the softmax layer with RF because RF tend to do well with intrusion detection problems. The stacking NDAEs included two layers of NDAEs with 3 hidden layers on each NDAE. The accuracy was obtained as 99.5999%. Nguyen Thanh et al [25] implemented DBN with stacked AEs and stacked restricted Boltzmann machines (RBM) [26] during the pre-training phase of deep learning. Their approach has been able to detect intrusions with a low error rate of less than 2%. The architecture of stacked AEs contained one hidden layer with an output layer but each hidden layer is connected to the input of the next AE and also to its own output and one of the hidden layers of an MLP. For stacked RBMs, there are no output layers but each hidden layer of an RBM is connected to the input layer of the next RBM and to a hidden layer of an MLP.

### 1.1.2 Credit Card Fraud Detection

Credit card fraud is a misuse of a credit card that includes making a transaction by swapping a stolen credit card, by fake cards, by copying and then faking of the card information, by collection of personal information, by phishing (malicious websites) or by employees who work in credit card companies [15]. An essential problem in credit card transactions is the detection of fraud transactions. Fraud detection in credit card financial activities is a huge problem impacting large financial companies and causing cuts of profit margins in billions of dollars yearly [27]. In European Union alone, according to European Central Bank (ECB), during 2010 the cost and losses caused by fraud reached a monentary value of 1.26 billion in Euro Payment Area alone [28]. Fraud detection is defined as the detection of illegal activites that occur in commercial and financial organizations such as banks, credit card providing companies such American Experess, insurance agencies such as SGK, stores, and cell phone companies, etc [12].

According to Yufeng Kou et al. [13], credit card fraud can be divided into two main categories: online fraud and offline fraud. Offline fraud is defined as the fraud committed by using a stolen or lost card at a store or a call center. This kind of fraud can be dealt by the institution issuing the card to revoke the card before it is used by a fraudster. Online fraud is committed through transactions done on the web (e.g. online shopping). In this kind of fraud only the card's information is demanded and client's signature is not needed or required.

A fraudster may operate through a website which looks like a legitimate website, on which he promotes products and services and sells goods at lower prices than the actual market prices. The uninformed purchaser provides his/her credit card information and makes a purchase. The fraudster then uses the obtained information to conduct his personal transactions [29]. Credit cards are among the best targets of frauds, since it cost little to steal a huge amount of money and in a little amount of time without taking too much risk. This is because most of the time it takes days or even weeks to detect credit card fraud crimes if they can be detected at all [15]. In real world applications, transactions are scanned by automatic tools, which may approve financial transactions as legitimate, send the most suspicious financial transactions to experts or leaves them unclassified until a client comes to complain in given time frame, in which case those transactions are labeled frauds otherwise they are labeled as legitimate [30]. In the past, financial institutes used to send SMS to users for every transaction held. However, this method increased operational costs due to the fact that each SMS might cost up to two dollar cents [29]. To reduce the unnecessary operational expenditures, the financial institutes decided to start reporting transactions of 50 dollars and above. As a result, many clients lost money and started migrating to competitors, which caused significant revenue loss for the institutes.

The aim of credit card fraud detection is to automatically and efficiently detect malicious use of a credit card. One approach is to use machine learning classifiers [28]. The challenge in this problem is the unbalanced data (genuine transactions by far outnumber fraud transactions) [31]. In this thesis, we worked on the credit card fraud detection dataset available in Kaggle [32]. The data set contains 30 attributes, including the time (in seconds) when the transaction took place, the amount of the transactions (in euros) and 28 attributes labeled as v1 to v28, which were obtained by applying PCA transformation to the original set of features [33] [34]. The PCA is performed in order to hide the sensitive information of the card holders due to legal and moral reasons because credit card detection is considered highly confidential and most of the time data related to card holders is not disclosed to public [35]. In our experiments, we removed the time attribute because we realized that it does not contribute anything to the classification. The credit card fraud detection data set is highly unbalanced: out of

284,807 transactions the positive class (frauds) account for 492 transactions, which constitutes approximately 0.172% of the transactions (Table 1.1.2.1).

| Class | No. of instances | Percentage |
|---|---|---|
| Normal | 284315 | 99.828% |
| Fraud | 492 | 0.172% |

Table 2.1.2.1: Credit card fraud detection dataset

### 1.1.3 TalkingData AdTracking Fraud Detection

TalkingData is one of the largest companies in China that provides independent big data services. The company features over 70% of nation's mobile devices. The company receives upto three billion clicks per day, of which 90% can turnout to be fraudulent. The company's method to combat fraud is to check users' profiles and flag the IP addresses that have many clicks per day without any apps installed as frauds. Using the acquired information, the company aims to generate an IP blacklist and device blacklist. To achieve this goal, TalkingData has launched a competition recently in Kaggle [4]. In this competition, the objective was to predict whether a user will download an app after clicking a mobile app advertisement. The attributes of the dataset are shown in the table below:

| Attribute name | Description | Type |
|---|---|---|
| Ip | IP address of device. | Categorical |
| App | application identifier for marketing. | Categorical |
| Device | device type id of client's mobile device (e.g., iphone 7, iphone 8, huawei mate 7, Samsung galaxy s8 etc.) | Categorical |
| Os | Operating system version id of the mobile device used | Categorical |
| Channel | channel id of mobile ad publisher | Categorical |

| | | |
|---|---|---|
| click_time | timestamp of click (in UTC) | Time |
| attributed_time | if user download the app for after clicking an ad, this is the time of the app download | Time |
| is_attributed | The class attribute to be predicted if the user has downloaded the app or not | Class |

Table 1.1.3.1: Different attributes of the dataset

## Previous works

When we were running the experiments on our workstation, it took more than three weeks to run and the memory consumption was increasing. Therefore we had to stop the experiments. However, we have been able to get results from the winner of the competition who claims after undersampling he still had to use 100 GB of memory. The following is what was done by the winning team.

Their solution depends on negative under-sampling, which means they use all positive examples (i.e. is attributed == 1) and down-sampled negative examples on model training. They down-sampled negative examples such that their size became equal to the number of positive ones. It discarded about 99.8% of negative examples, but they didn't observe deterioration in performance when they tested with their initial features. Moreover, they stated that they could get better performance when creating a submission by bagging five predictors trained on five sampled datasets created from different random seeds. This technique enabled them to use hundreds of features while keeping LGB training time less than 30 minutes. To extract features, first, they started with features made available in Kaggle's Kernels section. They did feature engineerings using all the data examples instead of the down-sampled ones. The Five raw categorical features (ip, os, app, channel, device), time categorical features (day, hour) and some count features. Then, they created a bunch of features in a brute-force way. For each combination of five raw categorical features (ip, os, app, channel, and device), they created the following click series-based feature sets (i.e., each feature set consists of 31 ($2^5$ - 1 features): click count within next one/six hours, forward/backward click time delta, and average attributed ratio of past clicks they didn't do feature selection. They just added all of them to the model. At that point, the LGB model's score was 0.9808.

Next, they tried categorical feature embedding by using LDA/NMF/LSA. Here is the pseudo code to compute latent Dirichlet allocation (LDA) topics of IPs related to app [23]. This ended up with 100 new features. They also computed similar features using NMF and PCA, in total 300 new features. 0.9821 with a single LGB. After that, they removed all raw categorical features except app since they supposed embedding features cover information available from them. Surprisingly, this minor change made the public LB score jump up from 0.9821 to 0.9828. Besides features mentioned here, they created higher dimensional LDA features and features that try to address the duplicate sample problem. Those features somewhat improve their public LB score.

For the models, they used day 7 and 8 for training and day 9 for validation, and chose the best number of iterations of LGB. Then, they trained a model on day 7, 8, and 9 with the obtained number of iterations for creating submission. After they finished feature engineering, owlight's five-bagged LGB model reached 0.98333 on public LB (and 0.98420 on private LB), which was trained on 646 features. The memory usage for training this model was around 100 GB. They implemented a simple three layer NN model as some kernels do. It scored worse than LGB models by 0.0013 points with 0.005 downsampling rate at first. The final three-bagged NN model scored 0.98258 on public LB. They made their final submission with a rank-based weighted averaging. It is composed of seven bagged LGB models and a single bagged NN. It scored 0.98343 on public LB [36].

## 1.2 Objectives

The objective of this thesis is to increase prediction accuracy on the anomaly detection problems including network anomaly detection, credit card fraud detection, and ad tracking fraud detection. All of the above mentioned problems contain a highly unbalanced classes, in which the number of samples belonging normal class is comparatively much larger than the number of samples having abnormal class. We use the F-measure (or F1-score) to optimize hyper-parameters of the models and to compare the performance of different algorithms on test sets rather than looking at the overall accuracy, which could be misleading. However, for some competitions such as Talking-Data Adtracking Dataset in kaggle.com area under the ROC curve (AUROC or AUC) is considered as the metric to score and rank the submissions made by the participants.

Therefore we also provide results on a range of accuracy metrics including AUC, whenever possible.

## 1.3 Main Contributions

All above mentioned problems can be viewed as machine learning problems because they have a data set, which can be used to train a learning model. Also, they all fall under the classification category for instance, a transaction can be classified as normal or fraud transaction. In the case of network anomaly detection, a connection can be classified as a normal connection or an attack.

Most of the previous works in the literature on anomalies and outlier detection apply feature selection methods on datasets and then use individual classifiers or a clustering algorithm such as K-means [37] and its variations to predict a correct class of outliers and anomalies. Others have also tried to undersample the number of examples from the normal class so that they can be equal to the number of instances in the outlier class. There are no papers which have explored stacking ensemble of classifiers technique. Most have just used individual classifiers to detect outliers on the datasets studied in this thesis.

As the main contribution of this thesis, various stacking ensembles are developed and optimized to obtain the best F-measure for the aforementioned anomaly detection problems. For example, the best stacking ensemble on credit card dataset contained Naïve Bayes, IBK, Random Forest, and J48 as the base learners and a logistic regression as the meta-learner with an L2-norm regularization employing Newton-CG solver. We also applied the undersampling technique to equate the proportion of negative and positive examples, feature selection, feature column normalization and implemented different classification algorithms for comparison. In KDDCup99 dataset, we improved the overall F-measure by more than 30% as compared to the literature. In Labris dataset, the improvement in F-measure was 0.5-0.6% both on binary and multi-class classificiation problems as compared to the literature. In credit card fraud detection, we improved the F-measure by 5% with respect to the results posted on Kaggle's Kernel section. The best accuracy is obtained using the under-sampled dataset.

Other classifiers which are considered as ensembles of weak classifiers such as Adaboost and Random Forest are also optimized in this thesis by performing 10-cross validation experiments on train set. For Adaboost, we have used both decision trees and decision stump. We have also included changing number of random seeds to obtain higher metrics measures.

In addition to developing and optimizing ensembles we also implemented statistical techniques proposed in the literature for outlier detection such as Mahalanobis distance and $\chi^2$-statistic measures.

## 1.4 Structure

The rest of the thesis is organized as follows:

**Chapter 2**

Chapter two describes the algorithms and methods developed for credit card fraud detection and hyper-parameter optimization.

**Chapter 3**

Chapter three explains the results from the experiements conducted and comparisons to the previous works in the literature.

**Chapter 4**

Chapter four concentrates on conclusion and future works by providing what can be improved in terms of feature engineering and possibilities of employing deep learning methods as more data is available.

# Chapter 2

## Methods

## 2.1 Stastical Methods

### 2.1.1 Mahalanobis distance

Mahalanobis distance is the generalization of how many standard deviations away is a point from the mean of a certain distribution [38]. The distance is small if the point is within the mean. The Mahalanobis distance can be computed as

$$\boldsymbol{M} = (\boldsymbol{O} - \bar{o})^{\boldsymbol{T}}\boldsymbol{S}^{-\boldsymbol{1}}(\boldsymbol{O} - \bar{o}) \qquad \textbf{(2.1.1.1)}$$

where o is the feature vector (i.e. a data sample) in test (or validation) data, ō is the mean of the feature vectors in train data labeled as "normal", $S^{-1}$ is the inverse of the scatter matrix (i.e. covariance matrix) for samples labeled as "normal" in train data. In the formula below, $O$ and ō are column vectors. The scatter matrix can be computed as

$$\boldsymbol{S} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{n=1}}^{\boldsymbol{N}} (\boldsymbol{O_n} - \bar{o})^{\boldsymbol{T}}(\boldsymbol{O_n} - \bar{o}) \qquad \textbf{(2.1.1.2)}$$

where $O_n$ is the $n^{th}$ training data, ō the mean of the feature vectors in “normal" labeled train data, and $N$ is the number of samples in train data [39]. The Mahalanobis distance can be used to find outliers by measuring the distance between a data sample and a distribution (or average of data samples) in such a way that when the distance is greater than a threshold the data sample can be categorized as an outlier. Therefore one can classify a test (or validation) data object $o$ as an outlier if

$$\boldsymbol{M > \tau_M} \qquad \textbf{(2.1.13)}$$

where $\tau_M$ is a threshold parameter [40]. Mahalanobis distances are computed between data samples in validation set and samples labeled as "normal" in train set of Labris data for different values of O (at least 10 different values). Then we used this threshold to detect outliers using Mahalanobis distance on the test set and computed the F-measure, overall accuracy, sensitivity, specificity and precision [3].

### 2.1.2 $\chi^2$ -statistic

The $\chi^2$-statistic is used to measure whether there is a significant difference between measured frequency and the expected frequency. It aims to measure if a certain point falls within the expected distance and be classified as normal or otherwise be classified as an outlier and it is computed as

$$\chi^2 = \sum_{i=1}^{D} \frac{(O_i - E_i)^2}{E_i} \qquad \textbf{(2.1.2.1)}$$

where $O_i$ is the $i^{th}$ feature value of the data object $O$ (i.e. sample), $E_i$ is the mean of the $i^{th}$ dimension among all samples labeled as "normal" in train set, $D$ is the number of dimensions in each feature vector [3]. A data sample is classified as an outlier if its $\chi^2$-statistic is greater than a threshold. $\chi^2$ -statistic is computed for validation set of Labris data taking different values for X (at least 10 values). We found the optimum T value that gives the best F-measure on this set. Then use the optimum threshold to detect outliers using $\chi^2$-statistic on test set and compute the F-measure, overall accuracy, sensitivity, specificity and precision [3].

## 2.2 Classification Based Methods

### 2.2.1 k-NN classification algorithm

The k-nearest neighbor method computes the distance between the feature vector of the test sample (whose class is unknown) and the feature vectors of the train set samples [41]. It then makes a decision by combining votes from the k samples of the train set that are closest to the test sample. For distance functions, Euclidean, Manhattan or Minkowski measures can be used [3]. An example that shows how k-nearest neighbor operates is given in Figure 2.2.1.1.

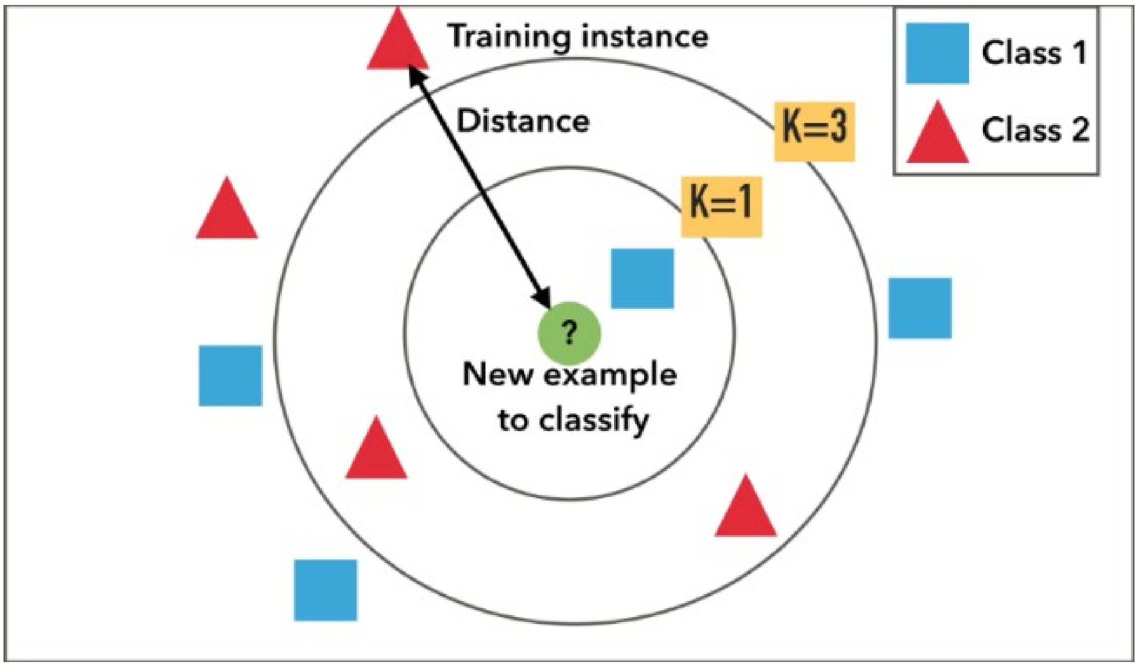


Figure 2.2.1.1: k-Nearest Neighbor algorithm [42].

We optimized the number of nearest neighbors (i.e. the k parameter) by performing 10-fold cross-validation experiments on train sets, in which we maximized the F-measure. Then we trained the models using these optimums and computed predictions on test data.

## 2.2.2 Logistic Regression

Logistic regression is a linear classifier, in which the decision boundary is a hyperplane. It may be attractive due to its short training times for problems that contain many numeric features and when the samples that belong to different classes can be separated by a hyperplane with high accuracy. Logistic regression can be applied both to binary and multi-class classification problems [8]. An example showing how logistiic regression works is provided in the figure below.

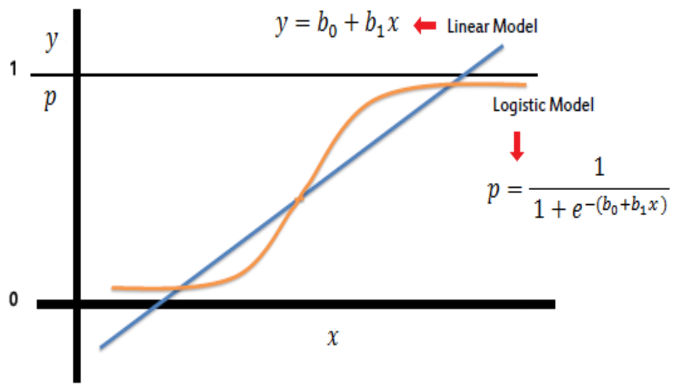


Figure 2.2.2.1 Logistic Regression [43]

In our experiments with logistic regression we optimized the ridge coefficient in the log-likelihood by performing a 10-fold cross-validation experiments on train sets maximizing the F-measure. Then we trained the models using these optimum values and computed predictions on test data.

## 2.2.3 Decision Tree (i.e. J48 in Weka)

A decision tree is a supervised learning method, which starts from the root node, performing a test on an attribute at each node and makes a classification decision when it reaches to a leaf node [9]. An example of a decision tree is given in Figure 2.2.3.1. In this thesis, we implemented the decision tree model J48 in WEKA and optimized the number of seeds by performing 10-fold cross-validation experiments on train sets and maximizing the F-measure. Then using the optimum number of seeds, we trained the model on train set and computed predictions test data.

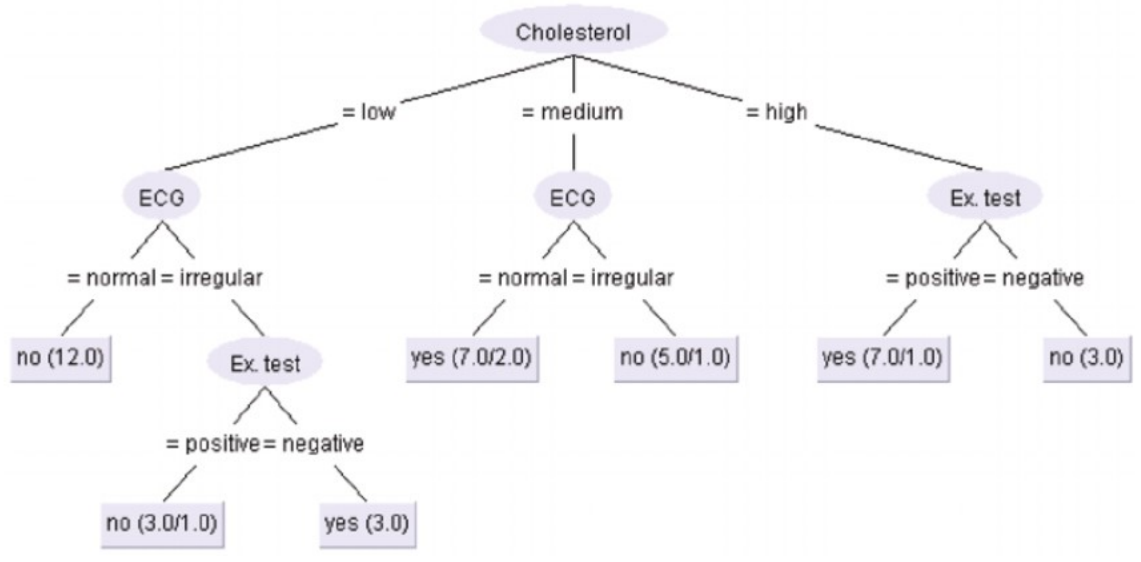


Figure 2.2.3.1 Decision tree [35]

### 2.2.4 Naive Bayes

The Naive Bayesian classifier is based on Bayes' theorem and it is called naïve because of its assumptions that all features in a feature vector are independent from each other given the class which is known as class conditional independence [6]. The algorithm is simple and works well with high dimensional input data. To explain naïve Bayes classifier well we can imagine a two class problem with one class having more examples than the other. Naïve Bayes believes that the new unseen example will most likely belong to the abundant class, this belief is modeled by the a priori probably distribution $P(c)$. The classifier uses the Bayes formula to calculate the a posteriori distribution $P(c|x)$ given the likelihood $P(x|c)$, the a priori distribution $P(c)$, and the evidence distribution $P(x)$. Naive Bayes [44] is summarized in equations 2.2.4.1 and 2.2.4.2 below. We used naïve Bayes to understand how a simple classifier performs on the anomaly detection problems studied in this thesis. After estimating the likelihood terms and the a priori distribution, predictions can be computed on a test example as the particular class that maximize the posterior distribution $P(c|x)$.

$$\boldsymbol{P(c|x) = \frac{P(x|c)P(c)}{P(x)}} \qquad \textbf{(2.2.4.1)}$$

$$\boldsymbol{P(c|x) = P(x_1|c) \times P(x_2|c) \times \ldots. \times P(x_n|c) \times P(c)} \qquad \textbf{(2.2.4.2)}$$

### 2.2.5 SVM

A support vector machine classifier separates the classes by a hyperplane after transforming the data to a higher dimensional space. It is among the max-margin classifiers, which find the optimum hyperplane that maximize the margin distance [8, see chapt 7]. Figure 2.2.5.1, below shows an example in which two classes are separated by a hyperplane. In this thesis, we employed the SVM with and RBF kernel and optimized the C (i.e. cost) and gamma parameters. This requires finding the best C, gamma pair that gives the highest F-measure on train set. For this purpose, we chose a grid of C, gamma values, and performed a 10-fold cross-validation experiment for each pair. Then we selected the particular pair that gave the best F-measure on the train set. The following values are considered for the C parameter: $2^{-5}$, $2^{-3}$, $2^{-1}$, $2^{1}$,$2^{3}$, …, $2^{13}$, $2^{15}$ and the following values for the gamma parameter: $2^{-15}$, $2^{-13}$, ..., $2^{-1}$, $2^{1}$, $2^{3}$, $2^{5}$. After obtaining the best hyper-parameter combination we trained the SVM using the optimums and computed predictions on test data.

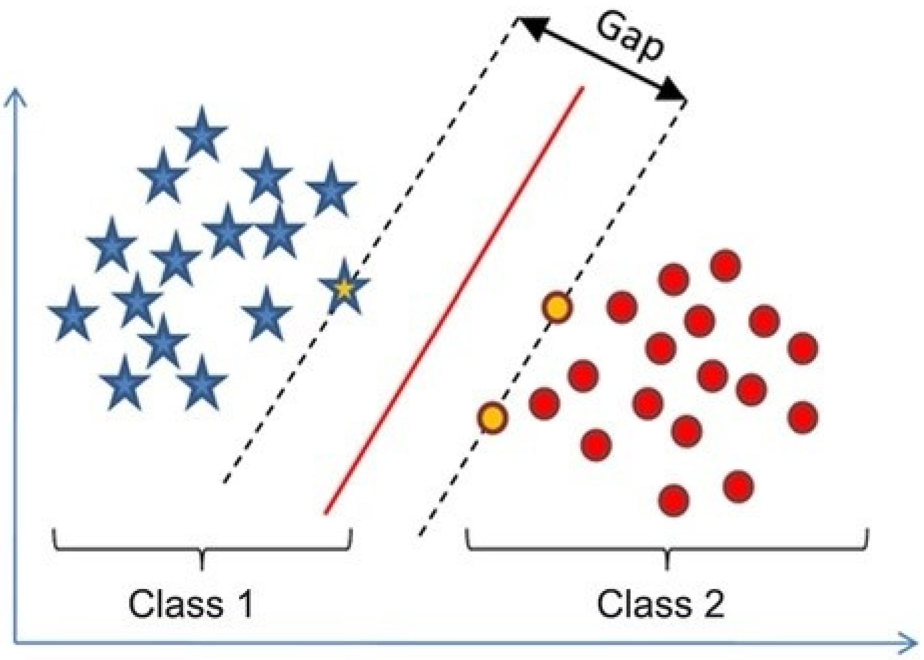


Figure 2.2.4.1 Support Vector Machine classifier [45]

### 2.2.6 Bagging

Bagging, which is the short form of bootstrap aggregating, is a meta-algorithm which has the potential to increase the stability and accuracy of machine learning algorithms. Bagging can be implemented both for regression and classification problems. In addition, it minimizes inter class variance ameliorating the overfitting problem. The algorithm generates different train sets from the data set with replacement trains a different model for each of them and computes predictions on test data. Ultimately the outputs from each learner is voted and a single output is obtained which makes it an ensemble of classifiers [3]. The Figure 2.2.6.1 represents the bagging algorithm.

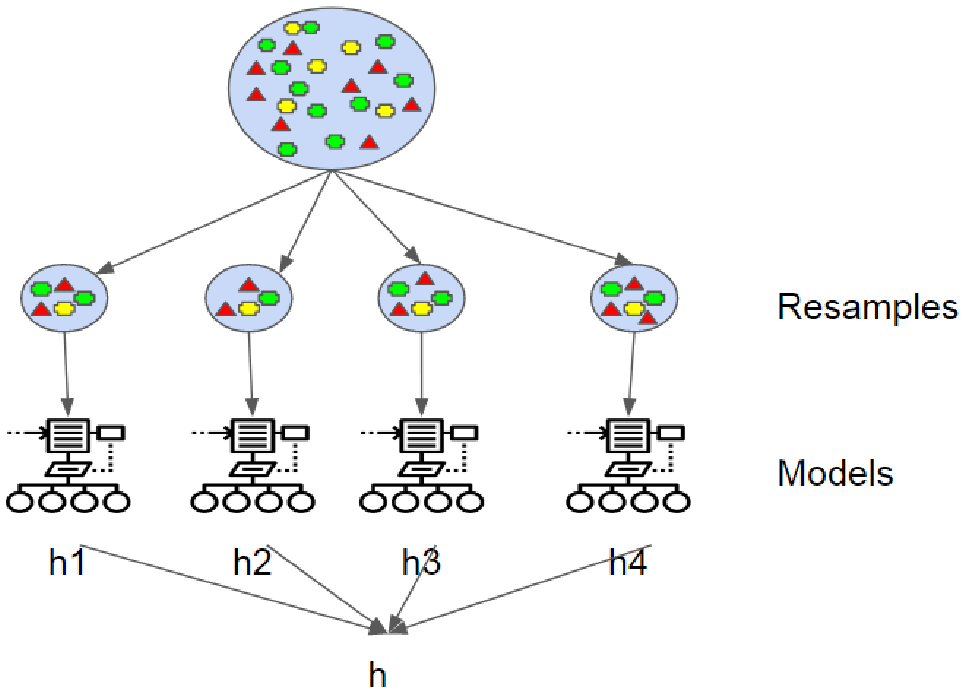


Figure 2.2.5.1 Bagging algorithm [46]

### 2.2.7 Random Forest

Random forest is a bagging variant, in which the base learners are decision trees each trained utilizing a randomly selected subset of variables or features [47]. Random forests are robust against overfitting and outliers. They can be used both in regression and classification problems. Similar to bagging, the outputs of the base learners can be combined by a voting procedure [3]. Figure 2.2.7.1 depicts how a random forest works.

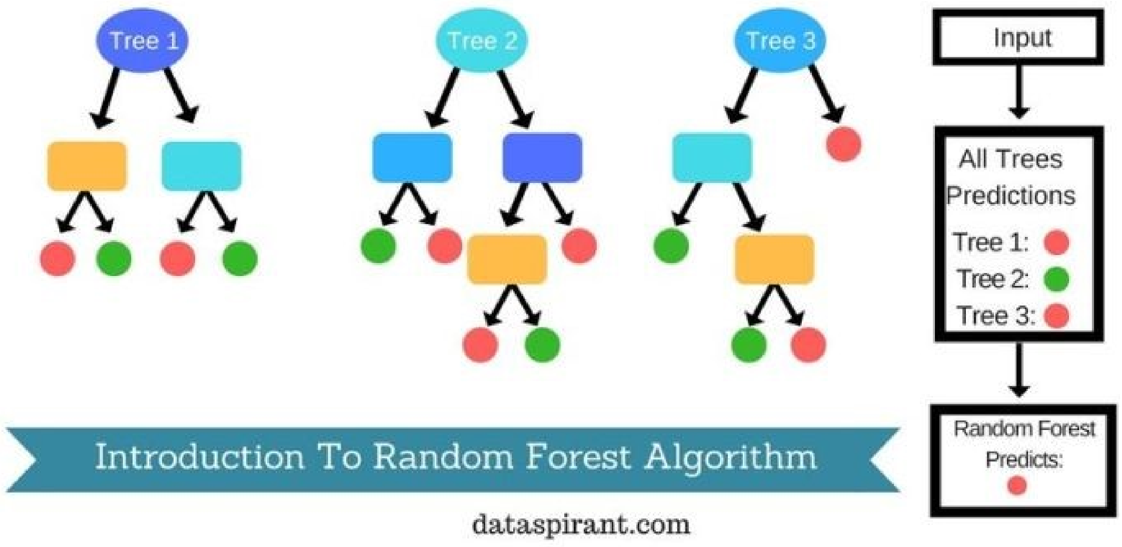


Figure 2.2.6.1 Random Forest [49]

In this thesis, we optimized the number of trees by performing 10-fold cross-validation experiments on train sets maximizing the F-measure. We considered the following values for the number of trees parameter: 10, 50, 100, 150, 200, 250, 500. After finding the optimum number of trees we trained the random forest classifier using the this optimum and computed predictions on test data.

### 2.2.8 AdaboostM1

AdaBoost is the abbreviation for "Adaptive Boosting" [10]. It is a meta algorithm that can improve the performance by combining several weak learners. Typically one-level desicion trees such as decision stumps are used as the base learners, which are added to the ensemble one at a time in each iteration. In boosting, each train sample has a weight, which represents how likely the sample will be selected in the next iteration. Initially the weights of all the train samples are equal. After the first base learner is trained by bootstrap sampling, predictions are computed for the train samples. The weights of the samples that are misclassified are increased and the weights of those that are classified correctly are decreased. The updated weights are used to form a new bootstrap sample to train a new base learner in the next iteration. The outputs of multiple-base learners are combined by a weighted voting approach, which uses higher weights for accurate classifiers and lower weights for less accurate ones [3].

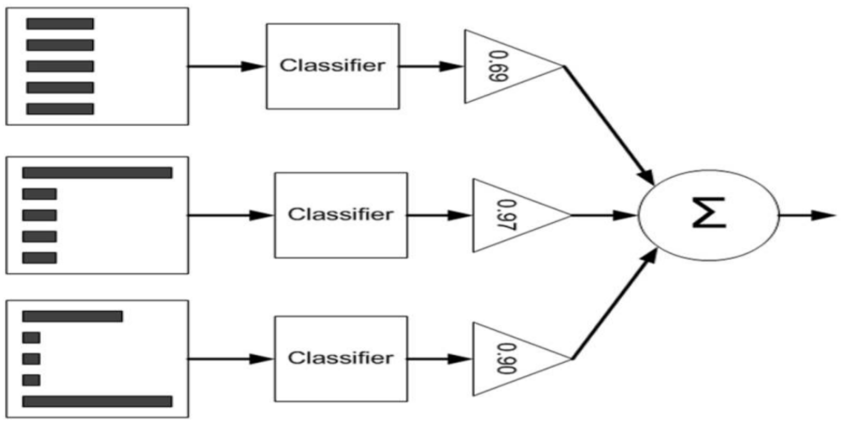


Figure 2.2.7.1: Adaboost [50]

In this thesis, employed decision stump and decision tree as the base learners of Adaboost. We optimized the number of seeds parameter by performing a 10-fold cross-validation experiment on train sets maximizing the F-measure with different number of seeds. We considered different number of seeds included 1, 2, 10, 15, 20, 30 seeds and different. Then using this optimum, we generated predictions on test data.

## 2.2.9 Stacking Ensemble of Classifiers on the whole data

After finding the optimum hyper-parameters for the individual classifiers on the train sets we combined them as base classifiers of the stacking ensemble and chose one of them as the meta learner.

### 2.2.9.1 Stacking Ensemble 1

This method employs k-NN, J48 which is the Java version of C4.5 [50], and naïve Bayes as the base learners and logistic regression as the meta learner.

### 2.2.9.2 Stacking Ensemble 2

This version employs the naive Bayes and k-NN (i.e. IBK in WEKA) as the base learners and logistic regression as the meta learner.

## 2.2.10 Stacking Ensemble of Classifiers on the under-sampled data

Due to high imbalance nature of our datasets (i.e our minority class is less than 1%) we applied under-sampling to balance the proportion of positive and negative examples.

### 2.2.10.1 Stacking Ensemble 3

This method employs naive Bayes, k-NN, J48, and random forest as the base learners and logistic regression as the meta learner.

### 2.2.10.2 Stacking Ensemble 4

This ensemble uses naive Bayes, k-NN, J48, and logistic regression as the base learners and random forest as the meta learner.

### 2.2.10.3 Stacking Ensemble 5

The fifth stacking ensemble combines naive Bayes [29], k-NN, random forest, and logistic regression using J48 as the meta classifier.

### 2.2.10.4 Stacking Ensemble 6

The last stacking ensemble uses naive Bayes [29], k-NN, random forest, and J48 as the base learners and logistic regression as the meta learner. This ensemble is implemented by scikit-learn employing various regularization techniques for logistic regression including L1 norm, L2 norm with Newton-CG solver, L2 norm with SAG, L2 norm with SAGA, L2 norm with LBFGS. The regularization coefficient C of logistic regression is optimized by performing 10-fold cross-validation experiments on train sets. The following values are considered for the C parameter: 0.000001, 0.00001, 0.0001, 0.001, 0.01,0.1, 1, 10, 100, 1000, 10000,100000.

## 2.3 Feature selection

Reducing the dimensionality of the data by removing unwanted features has the potential of improving the accuracy of machine learning algorithms. It also makes them run faster. Furthermore, dimensionality reduction produces a more compact and more understandable representation of the most relevant features.

There are two techniques used in feature selection, namely: the filter technique and the wrapper technique. Filter technique uses a metric to rank the features, which can be combined with a search algorithm to find the best feature subset. Wrapper technique uses a machine learning algorithm to produce a subset of desired features which will ultimately be used for learning. Different search techniques can be used to find the optimum feature subset such as forward selection, backward selection, and bidirectional selection (i.e. a combination of forward and backward selection). We used both filter and wrapper approaches to select the top five features and checked if the results improve compared to the case that uses all the features. In WEKA, we use WrapperSubsetEval attribute evaluator which employs by default a 5-fold cross validation. Furthermore, WEKA has OneRAttributeEval which is based on OneR classifier, InfoGainAttributeEval which is based on C4.5 classifier and information gain, GainRatioAttributeEval which uses gain ratio, SymmetricalUncertaintyAttributeEval, ChiSquaredAttributeEval which computes the $\chi 2$ statistic of each feature with respect to the class, ReliefFAttributeEval which is example-based feature evaluator, SVMAttributeval which uses SVM to decide the value of features, PrincipalComponents which uses principal components transform and choose the largest eigenvectors, and LatentSemanticAnalysis that performs latent semantic analysis and transformation. For CfsSubsetEval, an entropy metric is used called symmetric uncertainty. CfsSbusetEval employs te correlation metric to assess the inter-redudancy of the features.

## 2.4 Feature Normalization

We normalized features to interval [0,1]. We used weka.filters.unsupervised.attribute.Discretize -B 10 -M -1.0 -R first-last –precision filter in WEKA for this purpose. Normalization can use equal bining approach or equal frequency approach to normalize features. Also, entropy based normalization is widely used in the literature [51, see chapt. 7].

## 2.5 Software

### 2.5.1 Matlab

MATLAB is a non-open source programming language developed by MathWorks. The name MATLAB is short form for matrix laboratory, which is a multi-model environment that can solve a wide range of numerical problems that include matrix computations. MATLAB contains libraries for machine learning, image processing, computer vision, and data visualization as well as interfaces for programs coded in other languages (e.g. C++). MATLAB also enables defining new user interfaces [53].

### 2.5.2 WEKA

Weka stands for Waikato Environment for Knowledge Analysis developed at the University of Waikato in New Zealand. It is an open source software written in Java and licensed under the GNU General Public License. Weka contains hundreds of machine learning algorithms for a wide range of tasks including pre-processing, regression, classification, clustering, association rule mining and visualization [21]. It also allows developers around the world to be able to develop and share new machine learning algorithms.

### 2.5.3 Scikit Learn

Scikit-learn (formerly known as scikits.learn) [53] is an open source software machine learning library for the Python programming language [54]. It contains a variety of regression, clustering, and classification algorithms including ensemble methods such as random forest, individual classifiers such as support vector machines, gradient boosting, unsupervised algorithms such as DBSCAN and k-means. Scikit-learn is made to work together with Python’s numerical and scientific libraries such as Scipy and NumPy.

# Chapter 3

# Results

To assess the accuracy, different datasets have been prepared. The next section explains the accuracy measures used in this thesis.

## 3.1 Accuracy Metrics

To evaluate the accuracy of classifiers, the following metrics have been computed: confusion matrix, overall accuracy, precision, recall, specificity, F-measure, AUC, and Matthew's correlation coefficient (MCC). Due to high imbalance of the datasets used, F-measure is used to find the optimum hyper-parameters of the models. The definitons of each metric are given below.

### 3.1.1 Confusion Matrix

Confusion matrix, which is also known as an error matrix, is a table used to compute the accuracy of a classification algorithm. Figure below summarizes the contents of a confusion matrix. The definitions of the terms on the figure are as follows:
TP = Number of samples predicted as positive and are labelled (i.e. actually) positive
TN = Number of samples predicted as negative and are labelled (i.e. actually) negative
FP = Number of samples predicted as positive but are labelled (i.e. actually) negative

FN = Number of samples predicted as negative but are labelled (i.e. actually) positive

### 3.1.2 Overall Accuracy

To calculate the overall classification accuracy, the predictions on the test set are compared with the correct labels. The equation below formulates how this metric is computed in percentage.

$$\boldsymbol{Accuracy} = \mathbf{100} \times \frac{(\boldsymbol{TP} + \boldsymbol{TN})}{(\boldsymbol{TP} + \boldsymbol{FN} + \boldsymbol{FP} + \boldsymbol{TN})} \qquad (3.1.2.1)$$

## 3.1.3 Precision

Precision is a measure, which computes how many of the positive predictions are correctly predicted as positive. It is formulate as follows

$$\boldsymbol{Precision} = \mathbf{100} \times \frac{(\boldsymbol{TP})}{(\boldsymbol{TP} + \boldsymbol{FP})} \qquad (3.1.3.1)$$

## 3.1.4 Recall

Recall computes how many of the instances whose true labels are positive are correctly predicted as positive. It is also known as the sensitivitiy or true positive rate. It is formulated as

$$\boldsymbol{Sensitivity} = \mathbf{100} \times \frac{(\boldsymbol{TP})}{(\boldsymbol{TP} + \boldsymbol{FN})} \qquad (3.1.4.1)$$

## 3.1.5 Specificity

Specificity is a measure that computes how many of the examples for which the true label is negative are correctly predicted as negative. Equation below formulates the the specificity measure

$$\boldsymbol{Specificity} = \mathbf{100} \times \frac{(\boldsymbol{TN})}{(\boldsymbol{TN} + \boldsymbol{FP})} \qquad (3.1.5.1)$$

## 3.1.6 F-measure

Also known as the F-Score, F-measure is an accuracy metric that computes the weighted harmonic mean of the recall and precision. It is formulated as

$$\boldsymbol{F - measure} = \mathbf{2} \times \frac{(\boldsymbol{Precision} \times \boldsymbol{Recall})}{(\boldsymbol{Precision} + \boldsymbol{Recall})} \qquad (3.1.6.1)$$

## 3.1.7 AUC

Area under the receiver operating characteristic (ROC) curve, which is obtained as a plot of true positive rate versus false positive rate for different decision thresholds.

### 3.1.8 MCC

Matthews Correlation coefficient (MCC) is another accuracy metric used to check the quality of a classifier. It uses the true positive, true negative, false positive and false negative values to calculate the MCC. The equation below shows how this measure is computed .

$$\boldsymbol{MCC} = \frac{\boldsymbol{TP \times TN - FP \times FN}}{\sqrt{\boldsymbol{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}} \qquad \textbf{(3.1.8.1)}$$

## 3.2 Results on Credit Card Fraud Dataset

For AdaboostM1 [10], we used decision stump [56] as the base learner. For the randomly generated test set (using stratified remove folds in WEKA) the overall accuracy on the fraud class was 99.9136% and the overall cross-validation accuracy on the fraud class was 99.92%. The other measures can be seen on the Tables 3.2.1 and 3.2.2. We also used J48 as the base classifiers and the results were the same.

Due to highly imbalance in class labels we also decided to use under-sampling by choosing equal number of normal transactions as the number of fraud transactions randomly. The reduced data set had 328 samples of the normal class and 328 samples of the fraud class. The best performance according to F-measure was by logistic regression with L2-norm regularizer and Newton-CG as the solver [56]. This method obtained an F-measure of 94.86% and a recall of 98.06% as it can be seen on Table 3.2.2 followed by the stacking ensemble 3 with an F-measure of 94.3% and recall of 92.7%.

| **Classifier** | **Accuracy** | **Recall** | **F-measure** | **FP Rate** | **Precision** | **ROC Area** | **MCC** |
|---|---|---|---|---|---|---|---|
| Stacking Ensemble 3 | 94.36% | 92.7% | 94.3% | 4.0% | 95.9% | 97.9% | 88.8% |
| AdaboostM1 | 94.05% | 92.1% | 93.9% | 0.4% | 95.9% | 97.4% | 88.2% |
| Stacking Ensemble 5 | 93.90% | 90.2% | 93.7% | 2.4% | 97.4% | 94.4% | 88.0% |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Logistic Regression | 93.29% | 91.5% | 93.2% | 4.9% | 94.9% | 97.4% | 86.6% |
| Stacking Ensemble 4 | 93.14% | 88.7% | 92.8% | 2.4% | 97.3% | 97.0% | 86.6% |
| Random Forest | 94.82% | 91.2% | 94.6% | 1.5% | 98.4% | 97.9% | 89.9% |
| Stacking Ensemble 1 | 99.97% | 89.9% | 93.2% | 3.0% | 96.7% | 95.7% | 87.1% |
| KNN | 92.84% | 88.1% | 92.5% | 2.4% | 97.3% | 95% | 86.1% |
| Naïve Bayes | 91.76% | 86.3% | 91.3% | 2.7% | 96.9% | 95.6% | 84.0% |

Table 3.2.1: The accuracy measures of classifiers for credit card fraud detection. A 10-fold cross-validation is performed on the train set.

| **Classifier** | **Accuracy** | **Recall** | **F-measure** | **FP Rate** | **Precision** | **ROC Area** | **MCC** |
|---|---|---|---|---|---|---|---|
| Stacking Ensemble 6 | 93.03% | 98.06% | 94.86% | 49.42% | 98.73% | 100% | 87.95% |
| Stacking Ensemble 1 | 93.00% | 88.03% | 93.15% | 47.83% | 96.48% | 100% | 84.98% |

Table 3.2.2: The accuracy measures of stacking ensembles 1 and 6 for credit card fraud detection. Logistic regression is employed as the meta learner with an L2 norm regularizer and a Newton-CG solver. A 10-fold cross-validation is performed on the train set.

| Classifier | Accuracy | Recall | F-measure | FP Rate | Precision | ROC Area | MCC |
|---|---|---|---|---|---|---|---|
| Stacking Ensemble 1 | 99.97% | 83% | 90.4% | 0.0% | 100% | 100% | 90.8% |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Stacking Ensemble 2 | 99.97% | 80.4% | 89.1% | 0.0% | 100% | 100% | 89.7% |
| Random Forest | 99.90% | 70% | 72% | 70% | 0.0% | 74% | 96% |
| Adaboost | 99.91% | 74% | 75% | 0.0% | 75.3% | 98.2% | 75% |
| KNN | 99.92% | 70% | 74.4% | 70% | 0.0% | 80% | 97% |
| Logistic Regression | 99.93% | 67.1% | 75.9% | 0.0% | 87.3% | 98.9% | 76.5% |

Table 3.2.3: The accuracy measures of classifiers for credit card fraud detection on test set

| **Classifier** | **Recall** |
|---|---|
| Logistic Regression c = 0.01 | 94.92% |
| Logistic Regression c = 0.1 | 89.39% |
| Logistic Regression c = 1 | 91.28% |
| Logistic Regression c = 10 | 91.59% |
| Logistic Regression c = 100 | 91.59% |

Table 3.2.4: The recall of logistic regression for credit card fraud detection. A 10-fold cross-validation is performed on the under-sampled train set [32]

| **Classifier** | **Recall** |
|---|---|
| Logistic Regression c = 0.01 | 55.96% |
| Logistic Regression c = 0.1 | 59.93% |
| Logistic Regression c =1 | 61.26% |
| Logistic Regression c = 10 | 61.85% |

| | |
|---|---|
| Logistic Regression c =100 | 61.85% |

Table 3.2.5: The recall of logistic regression for credit card fraud detection. A 10-fold cross-validation is performed on the train set [32]

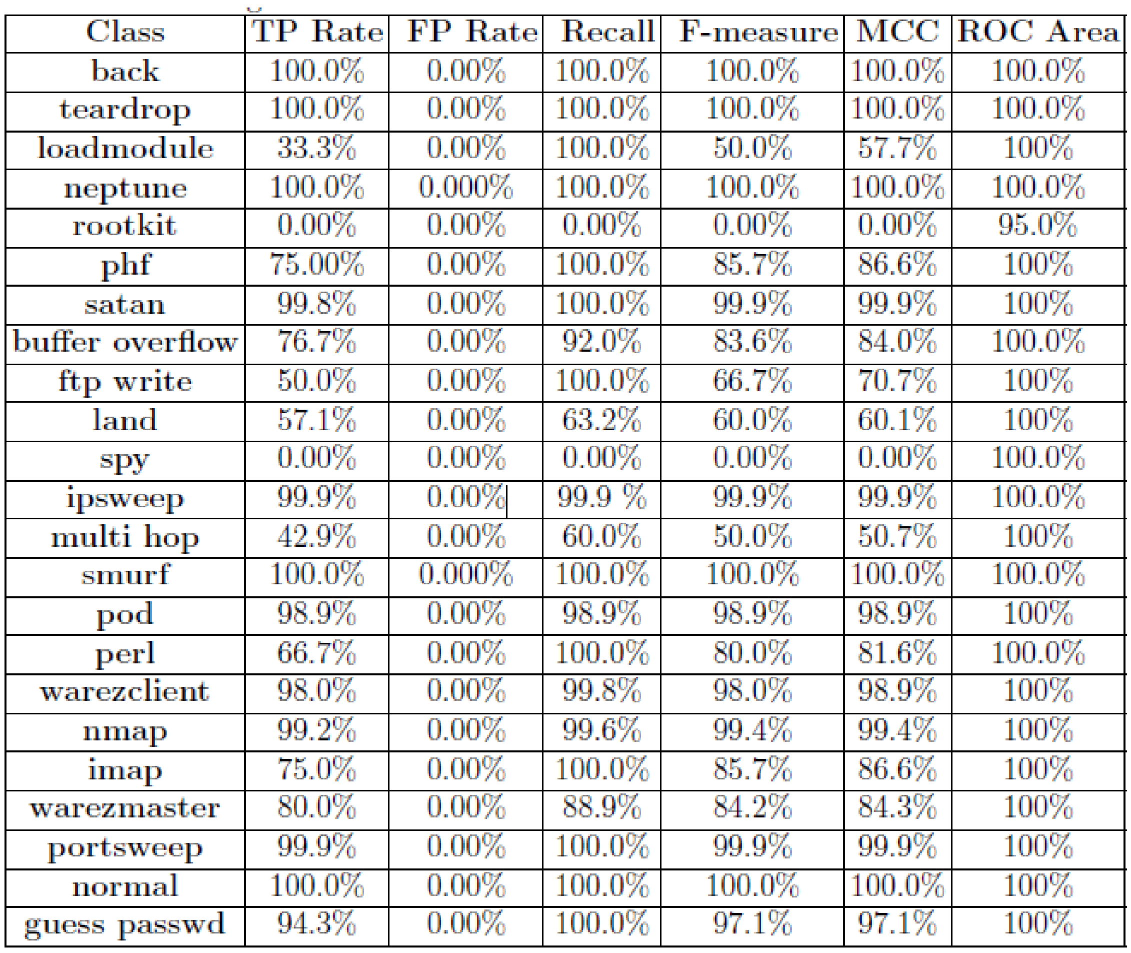
## 3.3 Results on KDD cup99 Dataset

### 3.3.1 Postprocessing and Validation

As mentioned earlier, a stratified 10-fold cross-validation is performed on train set provided to optimize the hyper-parameters of the models. Then the models are trained using the optimum hyper-parameter configurations and predictions are computed on test data. For the KDD cup99 Dataset, the performance of the different classifiers including simple classifiers gave astonishing results. Below are the results of the random forest classifier with the number of trees set to 10.

| Correctly classified examples | Incorreclty classified examples | K coefficient | Mean absolute error | RMS Error | RAE | RRS Errror |
|---|---|---|---|---|---|---|
| 99.9964% | 0.0036% | 0.9999 | 0 | 0.0018 | 0.022% | 1.126% |

Table 3.3.1.1: Binary classification accuracy measures of random forest for network intrusion detection on KDD Cup99 dataset. A 10-fold cross-validation is performed on train set



| Class | TP Rate | FP Rate | Recall | F-measure | MCC | ROC Area |
|---|---|---|---|---|---|---|
| back | 100.0% | 0.00% | 100.0% | 100.0% | 100.0% | 100.0% |
| teardrop | 100.0% | 0.00% | 100.0% | 100.0% | 100.0% | 100.0% |
| loadmodule | 33.3% | 0.00% | 100.0% | 50.0% | 57.7% | 100% |
| neptune | 100.0% | 0.000% | 100.0% | 100.0% | 100.0% | 100.0% |
| rootkit | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 95.0% |
| phf | 75.00% | 0.00% | 100.0% | 85.7% | 86.6% | 100% |
| satan | 99.8% | 0.00% | 100.0% | 99.9% | 99.9% | 100% |
| buffer overflow | 76.7% | 0.00% | 92.0% | 83.6% | 84.0% | 100.0% |
| ftp write | 50.0% | 0.00% | 100.0% | 66.7% | 70.7% | 100% |
| land | 57.1% | 0.00% | 63.2% | 60.0% | 60.1% | 100% |
| spy | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 100.0% |
| ipsweep | 99.9% | 0.00% | 99.9 % | 99.9% | 99.9% | 100.0% |
| multi hop | 42.9% | 0.00% | 60.0% | 50.0% | 50.7% | 100% |
| smurf | 100.0% | 0.000% | 100.0% | 100.0% | 100.0% | 100.0% |
| pod | 98.9% | 0.00% | 98.9% | 98.9% | 98.9% | 100% |
| perl | 66.7% | 0.00% | 100.0% | 80.0% | 81.6% | 100.0% |
| warezclient | 98.0% | 0.00% | 99.8% | 98.0% | 98.9% | 100% |
| nmap | 99.2% | 0.00% | 99.6% | 99.4% | 99.4% | 100% |
| imap | 75.0% | 0.00% | 100.0% | 85.7% | 86.6% | 100% |
| warezmaster | 80.0% | 0.00% | 88.9% | 84.2% | 84.3% | 100% |
| portsweep | 99.9% | 0.00% | 100.0% | 99.9% | 99.9% | 100% |
| normal | 100.0% | 0.00% | 100.0% | 100.0% | 100.0% | 100% |
| guess passwd | 94.3% | 0.00% | 100.0% | 97.1% | 97.1% | 100% |

Table 3.3.1.2: Multi-class classification accuracy measures of random forest for network intrusion detection on KDD Cup99 train set. A 10-fold cross-validation is performed on train set

| Classifier | Accuracy |
|---|---|
| J48 | 99.96% |
| Random Tree | 99.95% |
| Adaboost | 97.86% |

Table 3.3.1.3: Binary classification accuracy of J48, random tree and Adaboost for network intrusion detection on KDD Cup99 train set. A 10-fold cross-validation is performed on train set

The results on Table 3.3.1.3 did not surprise us because most researchers claim decision trees algorithms perform well on the KDDCup99 and other intrusion detection problems. The following table summarizes results from other researchers.

| Classifier | J48 | NB | NB Tree | RF | Random Tree | MLP | SVM |
|---|---|---|---|---|---|---|---|
| Accuracy | 93.82% | 82.66% | 93.51% | 92.79% | 92.53% | 92.26% | 65.01% |

Table 3.3.1.4: Binary classification accuracy measures of various methods on KDDTest [17]

| Classifier | J48 | NB | NB Tree | RF | Random Tree | MLP | SVM |
|---|---|---|---|---|---|---|---|
| Accuracy | 81.05% | 76.56% | 82.02% | 80.67% | 81.59% | 77.41% | 69.52% |

Table 3.3.1.5: Binary classification accuracy measures of various methods on KDDTest+ [17]

| Classifier | J48 | NB | NB Tree | RF | Random Tree | MLP | SVM |
|---|---|---|---|---|---|---|---|
| Accuracy | 63.97% | 55.77% | 66.16% | 63.26% | 58.51% | 57.34% | 42.29% |

Table 3.3.1.6: Binary classification accuracy measures of various methods on KDDTest-21 [17]

In addition, the Tavallaee et al. [21] ran experiments on an unseen test set called KDDTest21, which consisted of 11,850 instances. Bolon-Canedo et al. [56] who use feature selection algorithms on the data set states that the filter technique was selected because of the size of the KDD Cup 99 dataset was large. The following is the comparison of the results (in percentage) attained in binary classification problem on the test set.

| Method | Error | True Positives | False Positives | No. of Features |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| EMD + Cons + One layer | 7.78 | 90.52 | 0.77 | 6 |
| EMD + Cons + PSVM | 7.78 | 90.53 | 0.78 | 6 |
| EMD + Cons + FNN | 8.01 | 90.18 | 0.54 | 6 |
| EMD + INT + C4.5 | 6.69 | 91.81 | 0.49 | 7 |
| PKID + Cons + NB | 7.99 | 90.18 | 0.42 | 6 |

Table 3.3.1.7: Binary classification accuracy measures of naïve Bayes and C4.5 using a subset of selected features on test set [56]

| Method | Error | True positives | False Positives | No. of Features |
|---|---|---|---|---|
| PKID + Cons + C4.5(0.25) | 5.15% | 94.07% | 1.90% | 6 |
| PKID + Cons + C4.5(0.50) | 5.14% | 94.08% | 1.92% | 6 |
| EMD + INT + C4.5(0.25) | 6.74% | 91.73% | 0.44% | 7 |
| EMD + INT + C4.5(0.50) | 6.69% | 91.81% | 0.46% | 7 |
| PKID + Cons + NB | 7.99% | 90.18% | 0.42% | 6 |

Table 3.3.1.8: Binary classification accuracy measures of various methods on KDD Test [57].

| Combination | Score according to cost matrix [58] | No. of features |
|---|---|---|
| KDD winner | 0.2331 | 41 |
| EMD + INT + C4.5 | 0.2344 | 11 |
| EMD + INT + C4.5 | 0.2324 | 15 |

Table 3.3.1.9: Binary classification accuracy of Decision tree (C4.5) with Entropy minimization discretization (EMD) with Interact algorithm (INT) on KDDTest [57].

In the paper by Bolon-Canedo et al. [2] a hybrid approach that combines different feature selection methods and discretization are employed to boost two class classification performance. The best results are realized when seven attributes are employed. This indicates that only those attributes, rather than of the whole set of 41 attributes, are relevant when the classification is implemented in a real world system. These results are shown in the table 3.3.1.10 with Proportional k-Interval Discretization (PKID) [56]

| Method | Error | TP | FP |
|---|---|---|---|
| PKID+Cons+C4.5(0.25) | 5.15 | 94.07 | 1.90 |
| PKID+Cons+C4.5(0.5) | 5.14 | 94.08 | 1.92 |
| EMD+INT+C4.5(0.25) | 6.74 | 91.73 | 0.44 |
| EMD+INT+C4.5(0.50) | 6.69 | 91.81 | 0.49 |
| KDD Winner | 6.70 | 91.80 | 0.55 |
| 5FNs_poly | 6.48 | 92.45 | 0.86 |
| 5FNs_fourier | 6.69 | 92.72 | 0.75 |
| 5FNs_exp | 6.70 | 92.75 | 0.75 |
| SVM Linear | 6.89 | 91.83 | 1.62 |
| SVM 2poly | 6.95 | 91.79 | 1.74 |
| SVM 3poly | 7.10 | 91.67 | 1.94 |
| SVM RBF | 6.86 | 91.83 | 1.43 |
| ANOVA ens. | 6.88 | 91.67 | 0.90 |
| Pocket 2cl. | 6.90 | 91.80 | 1.52 |
| Pocket mcl. | 6.93 | 91.86 | 1.96 |

Table 3.3.1.10: Binary classification accuracy measures of various methods on KDDTest [2].

## 3.4 Results on Labris Dataset

### 3.4.1 Mahalanobis distance

Mahalanobis distance is good at detecting outliers, which are usually far away from the mean of the normal examples. In this thesis, Mahalanobis distance based classifier is implemented for binary classification problem. The optimum threshold for classifying a data sample as an outlier or not is chosen at if its Mahalanobis distance is 75 percentile or below of all the samples distances and every instance's Mahalanobis distance beyond the threshold is classified as an attack. To assess prediction accuracy, We randomly selected a subset of data samples from the training set (such as 20%) and saved them into separate files. These were used as validation data to optimize model configurations or hyper-parameters when necessary. We saved the remaining samples as the new train data (excluding the validation data) in WEKA and then a script that computes Mahalanobis distance is run in Matlab. Then the accuracies given in Table 3.4.1.1 are computed using Java. Based on these results, Mahalanobis distance based classifier did not perform well in network anomaly detection on Labris data.

| Classifier | Accuracy | F-measure | Sensitivity | Precision | Specificity |
|---|---|---|---|---|---|
| Mahalanobis | 70.39% | 13.49% | 25.05% | 0.092% | 74.99% |

Table 3.4.1.1: Binary classification accuracy of Mahalanobis distance based outlier detection method on Labris test set

### 3.4.2 Chi-square

**Preprocessing**

Similar to Mahalanobis distance, $\chi^2$-statistic is used to discriminate attacks from normal examples. We randomly selected a subset of data samples from the training set (such as 20%) and saved them into separate files. These were used as validation data to optimize model configurations or hyper-parameters when necessary. We saved the remaining samples as the new train data (excluding the validation data) in WEKA and then a Matlab script was executed that computes the chi-square measure. The optimum threshold to classify a data sample as attack is set to 75 percentile. Then, prediction accuracies given in Table 3.4.2.1 are computed using Java. These results show that the $\chi^2$-statistic does not perform well in network anomaly detection on Labris data.

| Classifier | Accuracy | F-measure | Sensitivity | Precision | Specificity |
|---|---|---|---|---|---|
| $\chi^2$-statistic | 70.24% | 13.078% | 24.29% | 0.089% | 74.91% |

Table 3.4.2.1: Binary classification accuracy measures of chi-square statistic based outlier detection method on Labris test set

### 3.4.3 Simple Classifiers

We implemented simple classifiers for network anomaly detection on Labris dataset including OneR, naïve Bayes and decision tree (J48). All three gave amazingly high accuracies around 99% because are few attributes can be sufficient for achieving high accuracy. Using attributeSelector in WEKA to learn these attributes and one by one attributes were removed by single removal of one attribute at a time and differenct combination of attributes removal but still the classifiers found high accuracy models from the remaining set of attributes.

| Classifier | Accuracy |
|---|---|
| OneR | 99.00% |
| Naïve Bayes | 99.00% |
| Decision Tree (J48) | 99.00% |

Table 3.3.3.1: Binary classification accuracy of simple classifiers. A 10-fold cross-validation is performed on Labris train set.

**K-Nearest Neighbors**

We optimized the number of nearest neighbors parameter (i.e. k) of the k-NN method. We considered values from 1 to 10 with increments of 1 and performed a 10-fold cross-validation on train set. The optimum k value is found as 1. The optimization step took about 40 hours on the a computer with 4 core processor and 8 GB of memory. All the values of k gave almost similar F-measure but WEKA suggested k = 1 as the best hyper-parameter. The command for optimizing k is "*weka.classifiers.meta.CVParameterSelection -P "K 1.0 10.0 10.0" -X 10 -S 1-weka.classifiers.lazy.IBk -K 1 -W 0 -A weka.core.neighboursearch.LinearNNSearch -A "weka.core.EuclideanDistance -R first-last" 1.000 0.000 1.000 1.000 1.000 1.000 1.000*"

Once the optimum k is found, predictions are computed on test data using this optimum. Table 3.4.3.2 below include the accuracy measures of k-NN method on Labris test data.

| Class | TP Rate | FP Rate | Recall | F-measure | MCC | ROC Area |
|---|---|---|---|---|---|---|
| Normal | 100.0% | 0.00% | 100.0% | 100.0% | 99.9% | 100.0% |
| Syn ack ddos | 64.4% | 0.03% | 67.5% | 65.9% | 65.7% | 98.3% |
| icmp ddos | 100% | 0.00% | 100.0% | 100.0% | 100% | 100% |
| rst ack ddos | 88.2% | 0.04% | 81.1% | 84.5% | 84.2% | 99.8% |
| rst ddos | 69.6% | 0.00% | 79.4% | 74.2% | 74.0% | 99.3% |
| Fin ddos | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 72.9% |
| ack ddos | 81.9% | 0.01% | 77.0% | 79.4% | 79.3% | 98.7% |
| http get | 99.8% | 0.00% | 100.0% | 99.9% | 100% | 100% |
| syn ddos | 87.2% | 0.03% | 87.0% | 87.1% | 86.8% | 99.5% |

**Table 3.4.3.2:** Multi-class classification accuracy of k-NN on Labris test set

**Logistic Regression**

We performed a 10-fold cross-validation experiment on the Labris train set. Table 3.4.3.3 below shows various accuracy measures.

**Support Vector Machines**

The best C, gamma pair is found as: C = $2^1$, and gamma = $2^{15}$ and the results are shown on the Table 3.4.3.3.

**Random Forest**

We considered different values for the number of trees but all of them gave similar F-measure. We selected this parameter as 200 and computed predictions on test data. Table 3.4.3.3 show accuracy measures of random forest.

| Classifier | Accuracy | F-measure | Sensitivity | Specificity | Precision |
|---|---|---|---|---|---|
| Random Forest | 98.7% | 98.7% | 98.7% | 99.99% | 98.7% |
| SVM | 98.02% | 97.8% | 98.6% | 98% | 100% |
| Stacking ensemble | 99.14% | 99.1% | 99.1% | 100% | 99.2% |
| Logistic Regression | 97.77% | 97.5% | 97.8% | 98% | 98.0% |

Table 3.4.3.3: Binary classification accuracy measures of different classifiers. A 10-fold cross-validation is performed on Labris train set

## 3.4.4 Stacking ensemble of classifiers

We employed logistic regression as the meta learner, J48, Naïve Bayes, k-NN as the base learners. We run with J48 as a base classifier, Naive Bayes as a base classifier, and Naive Bayes with KNN as base classifiers. Also, the combination of all of the above, i.e. the combination of logistic regression as meta learner and J48, Naïve

Bayes, K-NN, and SVM as base classifiers gave the best perfomance. The results were as shown in Table 3.4.3.3.

### 3.4.5 Feature Normalization

k-NN classifier (i.e. IBK in WEKA) was chosen for the normalized attributes. All of the tested classifiers' performance dropped a little when the features were normalized.

| Accuracy | F-measure | Precision | Sensitivity | Specificity |
|---|---|---|---|---|
| 98.5% | 98.5% | 98.47% | 98.5% | 99.99% |

Table 3.4.5.1: Binary classification accuracy measures of k-NN on non-normalized features. A 10-fold cross-validation is performed on Labris train set

| Accuracy | F-measure | Precision | Sensitivity | Specificity |
|---|---|---|---|---|
| 98.47% | 98.6% | 98.6% | 98.6% | 99.99% |

Table 3.4.5.2: Binary classification accuracy measures of k-NN on normalized features. A 10-fold cross-validation is performed on Labris train set

### 3.4.6 Results and Evaluation

Mahalanobis distance and Chi-Square based classifiers did not perform well on this data set because there were many overlaps of data points of anomaly and normal data. SVM performed well on the anomalies. All classifiers predicted correctly the normal class data point i.e. TP rate was 1.00 due to big difference in ratio of normal and attacks, the overall accuracies of classifiers are obtained as high.

| Classifier | Accuracy | F-measure | Sensitivity | Precision | ROC area |
|---|---|---|---|---|---|
| Logistic Regression | 98.99% | 90.3% | 88.5% | 92.2% | 99.99% |
| k-NN | 98.47% | 98.6% | 98.6% | 98.6% | 99.99% |
| SVM | 98.02% | 97.8% | 98.0% | 98.6% | 100% |
| RF | 98.7% | 98.7% | 98.7% | 98.7% | 99.99% |
| Stacking | 99.13% | 99.1% | 99.1% | 99.2% | 99.99% |

| | | | | | |
|---|---|---|---|---|---|
| Ensemble 1 | | | | | |
| Normalized k-NN | 98.47% | 98.5% | 98.5% | 98.5% | 74.91% |

Table 3.4.6.1**:** Binary classification accuracy measures of various classifiers. A 10-fold cross-validation is performed on Labris train set

### 3.4.7 Feature Selection

There were four main feature selection methods implemented namely: CfsSubsetEval [59], ClassifierAttributeEval, InfoGainAttributeEval, and Classifier subset evaluator. Out of 41 features, 9 features were selected using CfsSubsetEval [60] as the ones which are relevant to classifiers. However, the features selected did not improve the classifiers perfomances. The selected features are: network_service, dst_bytes, tw_shConnectionCount, tw_shSYNErrorRate, cw_shConnectionCount, cw_shResetRate, cw_shSameServiceRate, cw_ssSYNErrorRate, cw_ssResetRate. The results are as shown in the Table 3.4.7.1.

| Classifier | Accuracy | F-measure | Sensitivity | Precision | ROC Area |
|---|---|---|---|---|---|
| Random Forest (Tree = 200) | 98.8074 % | 98.8% | 98.8% | 98.8% | 100.0% |
| KNN (k = 1) | 98.7504% | 98.7% | 98.8% | 98.7% | 99.8% |

Table 3.4.7.1: Binary classification accuracy measures of various classifiers after feature selection is performed. A 10-fold cross-validation is performed on Labris train set

## 3.5 Results on TalkingData Adtrack Fraud Detection Dataset

Due to the bulk of dataset (i.e. about 185 milllion instances with seven attributes). This dataset is recent we are still working on it

**Previous works**

Submissions were evaluated using the area under the ROC curve measure in Kaggle. The highest AUC score was 0.9843223.

# Chapter 4

# Conclusion

In this thesis, different classiffcation algorithms have been implemented including basic classifiers and more complex ones. Moreover, other techniques such as normalization, under-sampling and feature selection were also employed. Techniques such as normalization did not yield better results than the unnormalized data and feature selection for most of the experiments did not improve the results either. Under-sampling has proven to work better than non-undersampled data because of the class imbalance problem present in the data sets we worked on for outlier detection.

Stacking ensemble of classifiers technique has improved the results considerably, although it is computationally more expensive as it includes training several base learners.

For future work, we propose other ways of feature engineering as it has been done by the winning team of TalkingData Adtracking dataset challenge, where other advanced features such as LDA/NMF/LSA, SOM, are employed. We can consider trying different combinations of features to form new feature vectors. In addition, trying different classifiers with different features and combinining the classifiers as in stacking ensemble of classifiers can be another field which could be tested. This technique of training different classifiers using different sets of features has started to emerge in object dectections in images and videos. As another future direction, we can consider applying deep learning techniques to anomaly detection problems studied in this thesis and incorporate them to the ensemble models.

# Bibliography

[1] Carl E Landwehr et al. “A taxonomy of computer program security
aws". In: ACM Computing Surveys (CSUR) 26.3 (1994), pp. 211-
254.
[2] Veronica Bolon-Canedo, Noelia Sanchez-Marono, and Amparo Alonso-
Betanzos. “A combination of discretization and filter methods for improving
classification performance in KDD Cup 99 dataset". In: Neural Networks, 2009. IJCNN
2009. International Joint Conference on. IEEE. 2009, pp. 359-366.
[3] University of Waikato. Weka. 2018. url: https://www.cs.waikato.ac.
nz/ml/weka/.

[4] Kaggle. Adtracking dataset. 2018. url: https://www.kaggle.com/c/
talkingdata-adtracking-fraud-detection/data.

[5] Jyothsna, V. V. R. P. V., Prasad, V. R., & Prasad, K. M. (2011). A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, *28*(7), 26-35.

[6] Richard O Duda, Peter E Hart, and David G Stork. Pattern classi_ca-
tion. John Wiley & Sons, 2012.
[7] Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, *3*(2), 95-99.
[8] Chris Bishop, Christopher M Bishop, et al. Neural networks for pattern
recognition. Oxford university press, 1995.
[9] James H Steiger, Alexander Shapiro, and Michael W Browne. \On
the multivariate asymptotic distribution of sequential chi-square statistics".
In: Psychometrika 50.3 (1985), pp. 253-263.
[10] Andrea Dal Pozzolo et al. “Credit Card Fraud Detection: a Realistic
Modeling and a Novel Learning Strategy". In: IEEE Transactions on
Neural Networks and Learning Systems (Accepted)(2017) (2017).
[11] dataaspirant. rf image. 2018. url: http://dataaspirant.com/2017/
05/22/random-forest-algorithm-machine-learing/.

[12] Roy De Maesschalck, Delphine Jouan-Rimbaud, and D_esir_e L Massart. “The mahalanobis distance". In: Chemometrics and intelligent laboratory systems 50.1 (2000), pp. 1-18.
[13] dni. bagging image. 2018. url: http : / / dni - institute . in / blogs /
bagging-algorithm-concepts-with-example/.
[14] Documentation. Scikit. 2018. url: http://scikit-learn.org/stable/
documentation.html.
[15] Elham Hormozi et al. “Accuracy evaluation of a credit card fraud detection

system on Hadoop MapReduce". In: Information and Knowledge
Technology (IKT), 2013 5th Conference on. IEEE. 2013, pp. 35-39.
[16] Ekrem Duman, Ayse Buyukkaya, and Ilker Elikucuk. “A novel and
successful credit card fraud detection system Implemented in a Turkish
Bank". In: Data Mining Workshops (ICDMW), 2013 IEEE 13th
International Conference on. IEEE. 2013, pp. 162-171.
[17] Pang-Ning Tan et al. Introduction to data mining. Pearson Education
India, 2006.
[18] J. Kevric, S. Jukic, and A. Subasi, An effective combining classifier approach using tree algorithms for network intrusion detection, Neural Computing and Applications, pp. 1-8, 2016.

[19] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, *A detailed analysis of the KDD CUP 99 data set*, in Proc. 2nd IEEE International Conference on Computational Intelligence for Security and Defense Applications, USA: IEEE Press, pp. 53-58, 2009.
[20] Kalpana Jaswal, Praveen Kumar, and Seema Rawat. “Design and development of a prototype application for intrusion detection using data mining". In: Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on. IEEE. 2015, pp. 1-6.
[21] Mahbod Tavallaee et al. ”A detailed analysis of the KDD CUP 99 dataset". In: Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on. IEEE. 2009, pp. 1-6.
[22] R Can Aygun and A Gokhan Yavuz. “Network anomaly detection with stochastically improved autoencoder based models". In: Cyber Security and Cloud

Computing (CSCloud), 2017 IEEE 4th International Conference on. IEEE. 2017, pp. 193-198.

[23] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. “Evaluating effectiveness of shallow and deep networks to intrusion detection system". In: Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on. IEEE. 2017, pp. 1282-1289.

[24] Nathan Shone et al. “A deep learning approach to network intrusion detection". In: IEEE Transactions on Emerging Topics in Computational Intelligence 2.1 (2018), pp. 41-50.

[25] Nguyen Thanh Van, Tran Ngoc Thinh, and Le Thanh Sach. “An anomalybased network intrusion detection system using deep learning". In: System Science and Engineering (ICSSE), 2017 International Conference on. IEEE. 2017, pp. 210-214.

[26] Ilya Sutskever, Geoffrey E Hinton, and Graham W Taylor. “The recurrent temporal restricted boltzmann machine". In: Advances in neural information processing systems. 2009, pp. 1601-1608.

[27] saed. naive image. 2018. url: http : / / www . saedsayad . com / naive _
bayesian.htm.

[28] Alejandro Correa Bahnsen et al. “Cost sensitive credit card fraud detection
using Bayes minimum risk". In: Machine Learning and Applica-
tions (ICMLA), 2013 12th International Conference on. Vol. 1. IEEE.
2013, pp. 333-338.

[29] Addisson Salazar et al. “Automatic credit card fraud detection based on non-linear signal processing". In: Security Technology (ICCST), 2012 IEEE International Carnahan Conference on. IEEE. 2012, pp. 207-212.

[30] Yoav Freund, Robert E Schapire, et al. “Experiments with a new boosting
algorithm". In: Icml. Vol. 96. 1996, pp. 148-156.

[31] Anusorn Charleonnan. “Credit card fraud detection using RUS and MRN algorithms". In: Management and Innovation Technology International Conference (MITicon), 2016. IEEE. 2016, MIT-73.

[32] kaggleuser. Fraud Table. 2018. url: https://www.kaggle.com/mlg-
ulb/creditcardfraud/home.

[33] Kyungnam Kim. “Face recognition using principle component analysis".
In: International Conference on Computer Vision and Pattern

Recognition. Vol. 586. 1996, p. 591.

[34] Ian T Jollie. “Principal Component Analysis and Factor Analysis" In: Principal component analysis. Springer, 1986, pp. 115-128.

[35] N Malini and M Pushpa. “Analysis on credit card fraud identification techniques based on KNN and outlier detection". In: Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2017 Third International Conference on. IEEE. 2017, pp. 255-258.

[36] kaggle. fork notebook kaggle. 2018. url: https://www.kaggle.com/ yuliagm/talkingdata-eda-plus-time-patterns.

[37] Adil M Bagirov, Julien Ugon, and Dean Webb. “Fast modified global k-means algorithm for incremental cluster construction". In: Pattern recognition 44.4 (2011), pp. 866-876.

[38] Shiming Xiang, Feiping Nie, and Changshui Zhang. “Learning a Mahalanobis distance metric for data clustering and classification". In: Pattern Recognition 41.12 (2008), pp. 3600-3612.

[39] Varun Chandola. Anomaly Detection: A Survey Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2007.

[40] Goeffrey J McLachlan. “Mahalanobis distance". In: Resonance 4.6 (1999), pp. 20-26.

[41] David W Aha, Dennis Kibler, and Marc K Albert. “Instance-based learning algorithms". In: Machine learning 6.1 (1991), pp. 37-66.

[42] adatanalyst. knn image. 2018. url: http://adataanalyst.com/machine-learning/knn/.

[43] researchgate. tree image. 2018. url: https://www.researchgate.net/ figure/Cost-sensitive-decision-tree_fig3_289283385.

[44] George H John and Pat Langley. “Estimating continuous distributions in Bayesian classifiers". In: Proceedings of the Eleventh conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc. 1995, pp. 338-345.

[45] J Ross Quinlan. “C4. 5: Programming for machine learning". In: Morgan Kauffmann 38 (1993).

[46] Yufeng Kou et al. “Survey of fraud detection techniques". In: Networking, sensing and control, 2004 IEEE international conference on.Vol. 2. IEEE. 2004, pp. 749-754.

[47] Leo Breiman. “Random forests". In: Machine learning 45.1 (2001),
pp. 5-32.

[48] Webb, G. I. (2000). Multiboosting: A technique for combining boosting and wagging. *Machine learning*, *40*(2), 159-196.

[49] in_nitescript. adaboost image. 2018. url: https://infinitescript.
com/2016/09/adaboost/.

[50] Yinsheng Qu et al. “Boosted decision tree analysis of surface-enhanced
laser desorption/ionization mass spectral serum profiles discriminates
prostate cancer from noncancer patients". In: Clinical chemistry 48.10
(2002), pp. 1835-1843.

[51] Ian H Witten et al. Data Mining: Practical machine learning tools and
techniques. Morgan Kaufmann, 2016.

[52] matlab. matlab documentation. 2018. url: https://es.mathworks.
com/products/matlab.html.

[53] Sergio Moro, Raul Laureano, and Paulo Cortez. “Using data mining for
bank direct marketing: An application of the crisp-dm methodology".
In: Proceedings of European Simulation and Modelling Conference-ESM'2011.
EUROSIS-ETI. 2011, pp. 117-121.

[54] mit. Kddcup99 dataset. 2018. url: http://kdd.ics.uci.edu/databases/
kddcup99/kddcup99.html.

[55] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python". In:
Journal of machine learning research 12.Oct (2011), pp. 2825-2830.

[56] Jun Liu, Jianhui Chen, and Jieping Ye. “Large-scale sparse logistic regression".
In: Proceedings of the 15th ACM SIGKDD international con-
ference on Knowledge discovery and data mining. ACM. 2009, pp. 547-
556.

[57] Veronica Bolon-Canedo, Noelia Sanchez-Marono, and Amparo Alonso-
Betanzos. “Feature selection and classification in multiple class datasets:
An application to KDD Cup 99 dataset". In: Expert Systems with Applications 38.5 (2011), pp. 5947-5957.

[58] Alexey Grigorev. (2018, July 21). *Cost Matrix*. Retrieved from http://mlwiki.org/index.php/Cost_Matrix

[59] Hall, M. A. (1998). Correlation-based feature subset selection for machine learning. *Thesis submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy at the University of Waikato*.

[60] Mohammad Khubeb Siddiqui and Shams Naahid. "Analysis of KDD CUP 99 dataset using clustering based data mining". In: International Journal of Database Theory and Application 6.5 (2013), pp. 23-34.