Radboud University Nijmegen

Faculty of Science

Ensemble Learning with small machine learning algorithms for Network Intrusion Detection

Thesis MSc Computer Science

Author:
Laurent Floor

Supervisor:
Dr. ir. Lejla Batina

Second reader:
Dr. Martha Larson

July 2020

Acknowledgements

First of all, I would like to thank everyone at Accenture who has been in contact with me. The willingness to exchange information, to help and advise me was always there. In particular, I would like to thank Dimitri Vedeneev for his supervision and the interesting conversations we had every week about the various aspects of digital security.

Secondly, I would like to thank professors Lejla Batina and Martha Larson of the Radboud University for their supervision from university. Their willingness to give advice and assess the result is indispensable.

Abstract

Networks are always vulnerable to unwanted access. Network intrusion detection aims to detect these intruders. Machine learning is a technique that can provide scalable tools to recognise and classify attack patterns, eventually to help deter or defeat certain types of network intrusions. In this thesis, I investigate the possibilities to combine different machine learning techniques. In this way, I want to compare the accuracy and improve the impact on the detection rate for the individual attack classes.
In the research, I use network data from the KDD99 data set and the CICIDS2017 data set. From these raw data I select features belonging to a certain attack class and train individual models with Naive Bayes learning and Decision Trees.
I compare four different techniques to combine the individual models. As ensemble methods, I investigate soft voting, hard voting, ANN and Decision Trees. As a result, I found that with soft voting it is possible to increase both the accuracy and the F_1 score for the attack classes with a certain combination of individual models.

Contents

1 Introduction

Just as the well-known sentence 'the only safe computer is a *dead* computer' is true [21], also the phrase 'only a *dead* network is secure' is true. As long as a network exist, it is vulnerable to intrusion. Therefore, the challenge this paper propose is to explore how to detect such intrusions faster.

Reality shows that truth behind the previous statement. A research survey of Accenture illustrates the impact of cybercrime on society [32]. The general message of the survey is that the number of attacks is increasing as well as the economic costs. This evolution of cybercrime is changing all the time, but the recent shift is in particular visible in three areas.
In the past four years, there is an evolution of the cybercrime targets. Not only stealing of hostage data is the goal of an attack, but also disrupting and destroying becomes more relevant for cybercrime. If data is the target, a new range of attacks focus not only on copying data, but also on trying to change data to undermine the trust and integrity of the data process. Thirdly, there is an evolution going on in the techniques used. Not only intrusions by using the weakest layer (humans) have become more advanced. Also, tactics, techniques and proceedures that normally used by nation-state become also available to "business" users.

Network intrusions are typically understood in the context of anomoly detection. 'An anomaly is an observation which deviates so much from others as to arouse suspicion that is was generated by a different mechanism.' [3] The detection of anomalies has numerous practical used in varies fields. Machine Learning (ML) algorithms are suitable techniques to distinguish between anomaly and regular behaviour. In previous works, machine learning has been used in different ways to detect network intrusion. A good survey that covers most of the algorithms is [3].

In this paper, I will contribute to the improvement of the network intrusion detection by machine learning. In recent papers, approaches are proposed that use ensemble learning: combining different ML classifiers to give an anomaly detection.
An analogy may be to compare to the field of camera photo quality. You can enhance the quality of the photograph by improve the quality of the lens or sensor. But, similar to my approach, it is also possible several less quality cameras and combine the image. This approach it done by some smartphone cameras.
I set out with three goals within my thesis research. I will focus on an overall higher accuracy, reducing the number of false positives and false negatives, and preserve explainability.

The research question for this thesis can be summarized by:

> Is it possible to create a detection model for network intrusion that consists of multiple small independent machine learning models, to obtain better accuracies and F_1-score?

I try to answer this research question into the following chapters. In chapter 2 I give an overview of the framework of a good network intrusion detection system. I will cover in that chapter the several parts of an intrusion detection system (IDS), such as data processing and applying the models. This chapter also contains an overview of the previous work on this topic. In chapter 3 I give my approach to tackle the research question. The choices for ML models and the data set that are made for this answer are

discussed in chapter 4 and 5 . The conducted research and the observation made, are given in chapter 6. Finally, the conclusions of our research are given in chapter 7

2 Literature study

Machine Learning techniques have been under developments for more than 20 years to help identify network activity at scale that might indicate an intrusion. In this chapter I give the information structured by a general framework for data analyses.
The basic steps to analyze data to obtain a prediction are the following:

1. Acquire a good data set, see chapter 5;
2. Process the data;
3. Apply detection model(s);
4. Give a prediction.

The last three steps will be discussed in the following three sections.

2.1 Data processing

Raw network capture data is not very useful for pure analysis. Steps need to be taken to prepare the data in order for machine learning models to work most efficiently. Two steps can be taken before the data can be used for the detection models.
Data transformation. The main step is to transform the values of the raw network data to normalized numerical values. The categorical data, for example protocol types as `HTTP` and `SMTP`, are not suitable. They must be encoded in a one-hot-encoded form. One-hot-encoding is preferred because simple numerical representation can give one class more impact than others due to a higher numerical value.
Data reduction. Another step that is possible, but not necessary, is the reduction of the size of the normalized data. This is not required, because you can to feed all data in the detection models. But for speed performance and improvement of the quality of the detection model, we can reduce the number of features of the data.
The easiest way to achieve this is by selecting only the features you think are relevant to the problem. For example, you feed the detection models with the important features of a particular class
Another way to create data reduction is to apply the principal component analysis (PCA) method[35]. This encoding method vectorizes all features to a lower number of dimensions.
Another approach is given in [25], where the authors use correlation analyses feature selection to reduce the dimensions of the normalized data.

2.2 Detection models

After the data processing stage, I can apply the detection models. In general, there are two approaches to detect deviant behaviour: by asking what is normal behaviour, or by asking what is anomalous behaviour.
Anomaly detection. The first view is the anomaly detection approach. In this case, the detection algorithm learns specific behaviour of malicious internet connections. For example, an internet connection outside a specific country is seen as malicious because you don't expect access from abroad.
Misuse detection. The second approach learns what is normal behaviour. Another class of anomalies can be detected from matching regular working patterns for users, and spotting any irregularities such as logins outside of working hours.

A general list of detection models is as follows:

- **Rule based models.** These kinds of models defines conditions that satisfies benign or malicious network traffic. According these rules the traffic is allowed or blocked.
- **Statistical models.** These kinds of models find outliers with mathematical statistical strategies. According the distribution of benign data, the traffic of outliers is blocked.
- **Supervised Machine Learning.** Supervised machine learning models learn to classify the data to a certain class. This is done in the training phase where the training data is given, as well the belonging classes.
- **Unsupervised Machine Learning.** Unsupervised machine learning models cluster the data. Its categories data, without knowing any information about the class labels of the data.

2.3 Prediction

There are two ways to express the outcome of a detection model [3]. One possibility is the class label that indicates if the data instance is normal or the belonging attack type. The other option give the prediction with probabilities for each class.
These two ways relate to each other by:

$$\text{Class label} = \arg\max_i(p_1, p_2, ..., p_i, ...p_n) \tag{1}$$

2.4 Multiple models

In this writing above, I focus on an individual model. But it is also possible to use multiple models. Each model has its own data pipeline and an own detection algorithm and data processing, and output. To combine these outputs in an final output, we need an *ensemble learning method*. There are several methods to combine the outputs[10]. The most important ones are:

1. **Hard voting.** For each instance, the ensemble method determines which class label is most common by majority voting.
2. **Soft Voting.** For each instance, the prediction weights are added together and with equation 1 the class label is determined.
3. **Weighted Voting.** This is an extension of Soft Voting, where the addition is weighted.
4. **Machine Learning.** Classification algorithms like Artificial Neural Networks and Decision tree can be used to map the output of the models to a final prediction by classification.

2.5 Previous work

The authors of [12] give an overview of of the different techniques of ensemble-based learning. They discussed several examples from literature that uses a supervised learning, unsupervised learning or has a hybrid approach. A hybrid approach means that both supervised and unsupervised techniques are combined. Besides the 14 data mining techniques, 5 papers are discussed that made use of statistical approaches such as the Hidden Markov Model.
To give more of a picture of what has been done and achieved in previous work, I will discuss four papers in detail.

In his paper [6], Borji proposed a combining classification approach with four base classifiers. The used base ML models are the Artificial Neural Network (ANN), Support Vector Machine (SVM), k-Nearest Neighbors (k-NN) and Decision Trees. The outputs of these classifiers are combined on three different ways: hard voting, by calculating the Bayesian average and by computing a belief function. Using the KDD99 data set they found that the detection rate of the combined classifiers is higher than the four individual classifiers. In this paper is not analysed the effect on the different attack types of the KDD99 data set.
The authors of [13] choose as base classifiers: Decision Tree, Random Forest, k-NN and a Deep Neural Network (DNN). Tested on the NSL-KDD data set (a successor of KDD99) they found that weighted voting is effectively improving the detection rate in comparison of the individual attacks. The final results show that the detection rate of the low present attack is still low.
In [24] the three base classifiers are ANN, Decision Trees and Logistic Regression. By manual setting weights to an ensemble learning algorithm, it is determined which algorithm benefits the attack/non-attack detection and the overall accuracy.
In [30] the authors create a sequential model consisting of five individual models. These models are in order: Random Tree, Decision Tree, k-NN, ANN, and Naive Bayes. The main goal of this paper is to reduce the false-negative rate. Using the KDD99 data set they showed that the sequential model performs better than individual models on accuracy and false negative rate. An extensive overview of the impact for the individual attack classes is missing in the paper.

3 Approach of the research

Given the general framework for a network intrusion detector as presented in the previous chapter, I will give and explain the choices I made for each step. This section describes my approach to solving the research question. I discuss the solution on how I will create a model that consist of different independent detection model.

My choice for the data set is discussed in chapter 5. The data sets I used were the KDD99 and the CICIDS2017 data sets. These labelled data sets have besides benign data traffic, respectively 4 and 7 attack types.
In the data processing step I will reduce the size of the input data. This is to done to create small data sets. I choose to select features that are relevant for each attack type. This selection is done following the literature. We will not use the PCA method because this give a reduction the explainability of the model. The disadvantage of the PCA method that it is not traceable which features are important for a decision.
In section 4, I will deep in the several machine learning models that are used in literature. There is also a reasoned choice is given for the two methods I use in the research: Naive Bayes learning and Decision Trees.

As output for the final model, for each network instance a classification whether the connection is normal or to which attack type it belongs will given. The key reason we choose for this kind of output is that it is explainable. From a raised alarm, the actions that are taken can be quite far going like in the end interrupting the network.
As method to ensemble the individual models, I will try the different approaches as described in the previous chapter.
I will generate a model according the steps described above, a scheme that represents the final model is given in figure 1. To obtain that model I will take the following steps:

1. I choose two or more different machine learning model, see chapter 4;
2. For each attack type, we select the important features;
3. For each ML model, we create for each attack type a detection model;
4. I reduce the number of independent models by looking at the accuracy and F_1 scores of the ensemble models.

In this research there a new additions. The most important ones in comparison with existing litereture are:

- Combine feature selection for certain attacks and multiple machine learning algorithms;
- Compare different ensemble methods;
- This approach is verified on the common and old data set KDD99 and the state-of-the art data set CICIDS2017.

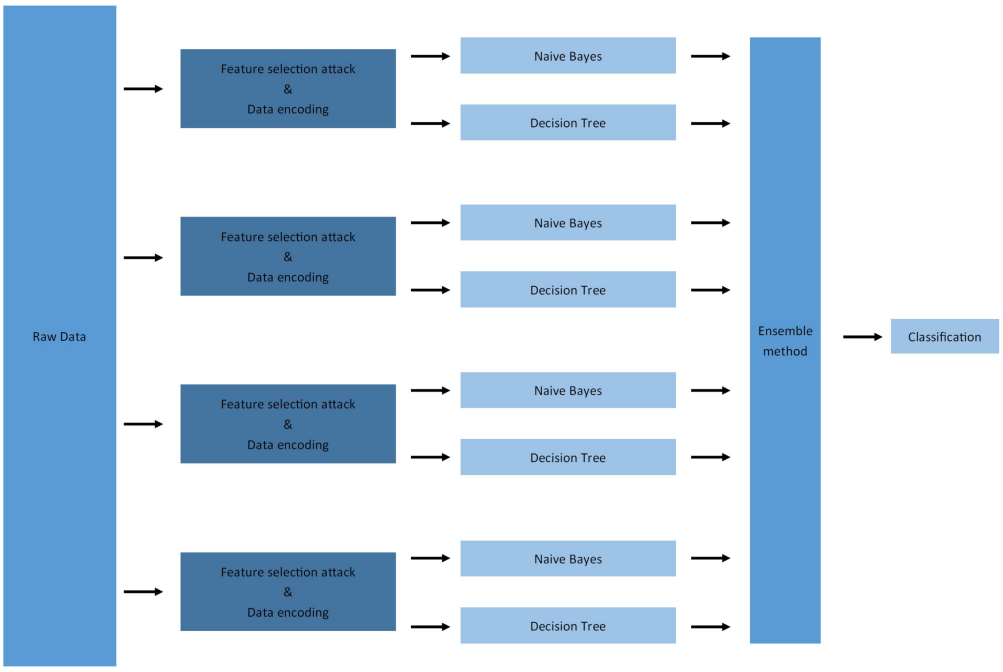


Figure 1: Schematic overview of our ensemble model.

4 Machine Learning algorithms

I consider in this chapter several supervised and unsupervised machine learning models. I explain the principles behind the most well-known and most used algorithms.
For this thesis, I combine a set of ML algorithms $\{A_1, A_2, ..., A_n\}$. A machine learning algorithm is an function $A : (X, W) \rightarrow (p_1, ..., p_m)$. Where X is the input data, in our case the network data, and the probabilities of the output are given with p_i, for m classes. During the training or learning phase, the set of weights W optimized. Using equation 1, the predicted class label c can calculated from the output of A.
This learning phase can be done in two ways, supervised or unsupervised. Supervised learning means that the input for the training phase is data with labels; for unsupervised are only data without labels is used.
The following paragraphs describe several models and can be seen as an extensive summary of different papers. The main papers I use for this overview are [3], [8] and [16]

4.1 Supervised methods

For the supervised learning methods, the algorithms used during the training phase are not only the network data instances $X = \{x_1, ..., x_m\}$, but also the the true class c.
Supervised Learning is especially good at recognizing patterns that have occurred before. Because it contains data that is labelled, it is also able to classify these patterns. This means that it is less suitable for recognizing new attacks.

4.1.1 Naive Bayes

The Naive Bayes method is a probabilistic classifier [28, 34]. It is derived from Bayes' theorem that express the probability of an event based on prior knowledge.
This famous probability equation is given by

$$P(c|X) = \frac{P(X|c)P(c)}{P(X)} \tag{2}$$

where $P(c|X)$ is the posterior probability, $P(c)$ is the probability of class c, $P(X|c)$ is the probability of the event given a class and $P(X)$ is the prior probability of the instance. The 'naive' step of this algorithm is to assume independent for each element in the data set X. In this way the equation becomes:

$$P(c|X) = \frac{P(x_1|c) \cdot (...) \cdot P(x_m|c)P(c)}{P(X)} \tag{3}$$

To apply Bayes theorem, take the following steps.
We need to calculate the different elements of the right hand side of equation 3. For the Naive Bayes Algorithm, we choose to use a Gaussian approach, because it deals with multiple categories.
This means that we calculate the likelihood of the features with

$$P(x_i|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp - \frac{(x_i - \mu_c)^2}{2\sigma_c^2} \tag{4}$$

When we have calculated all parts of equation 4, we are able to map with equation 3 x_i to the class with the highest probability.
One of the advantages of the Naive Bayes Algorithm is that it execute very fast and is able to handle big data sets.

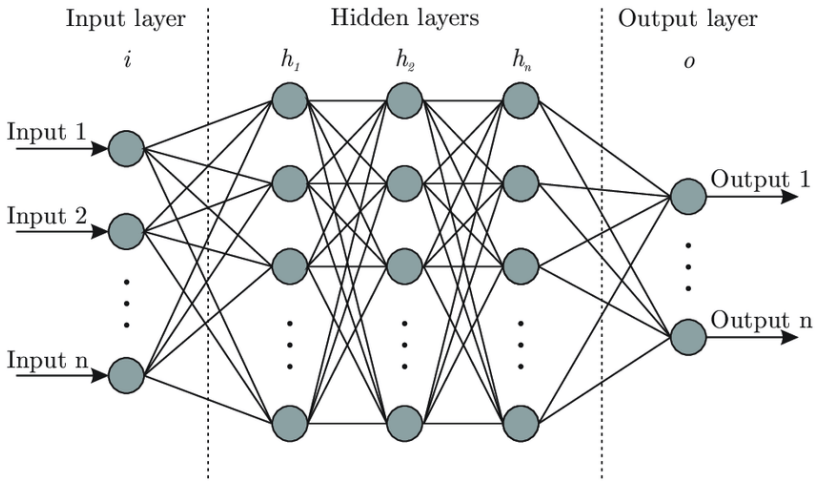


Figure 2: Schematic representation of an Artificial Neural Network [7].

4.1.2 Artificial Neural Networks

An Artificial Neural Network (ANN) consist of a web of connected nodes[26]. Inspired by the human brain, this method is classifying an instance using an input layer, hidden layers and an output layer. Each layer has several nodes, and these nodes are connected with edges to the nodes in the layer before and after. Each edge has a weight and during the training process the weights are updated.
Due to the number of different nodes, its creates a model that has the ability to fit very good. The down side is that train such a model takes a lot of time.
Given the n layers, where layer i has j nodes, and the networks has weights W. The algorithm to train a Neural Networks is as follows:
The algorithm starts by initializing the weights to small values. For instance x_i, the values of the features are the input for the nodes of the input layer. For each instance we calculate the output $o = (p_1, ..., p_m)$. Then for every training example (x_i, c) the distance between the actual class c and the output o is minimized by back propagation. The algorithm ends when the loss function is minimized to certain degree or a time limit is exceeded. For a schematic overview see figure 2.
More elaborated version of an ANN is covered with deep learning, see [11].

4.1.3 K-Nearest Neighbours

This classifier, shortly k-NN [18], is based on a function that measures the distance between two samples. An example of this function is the Euclidean distance. Given an test instance x_i, the algorithm calculate the k neighbours in the train set that has the shortest distance. By majority voting, the most apparent class of the neighbors, the attack class is chosen.
A good example is given in figure 3 The disadvantage of k-NN is that the storage requirements are large, because every training instance has to be stored. Also the time for prediction is high, because it is necessary to determine all distances to the training set before knowing which instances are the neighbors.

4.1.4 Decision trees

Decision trees consist out of three basic elements, a schematic version is visible in figure 4. First, we have the start of the tree: the root node. Secondly, there are the internal

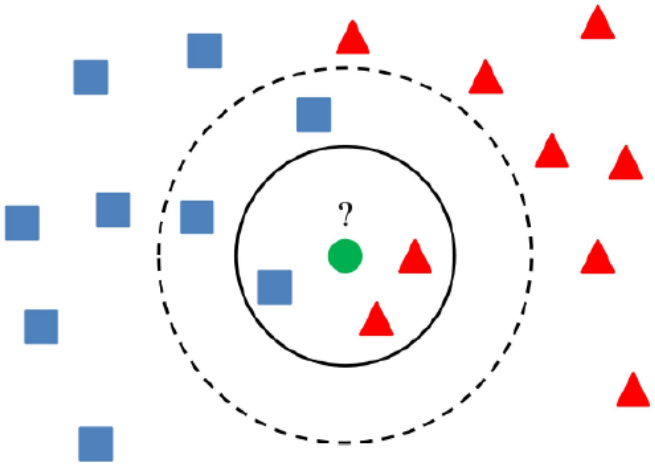


Figure 3: Schematic explanation of the k-NN classifier. To classify the green dot, you need to calculate the nearest neighbors. For $k = 3$ the majority of the neighbors is blue; for $k = 5$ the majority of the neighbors is red. Depending on the choice of k, is classified as blue or red.[4]

nodes. Both at the root node and internal nodes, a decision is made which edge will taken. This decision depends on the values or features of the given instance. Each edge leads to another decision node, or finally to an end node. These end nodes is also called leafs.
Given the training set the decision tree is build. There are different methods for construction. The most famous ones are $CART$ and $IDS3$ and it successors $C4.5$ and $C5$. After building the tree, classification is possible by putting a test instance in the top of the tree.
The big advantage of this type of classification is the possibility to give a visual representation of the tree.

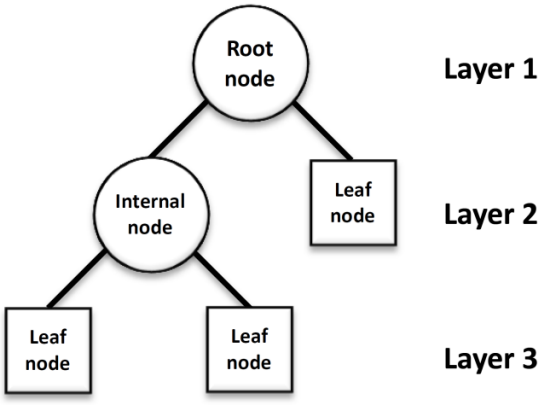


Figure 4: Schematic overview of a decision tree [9]

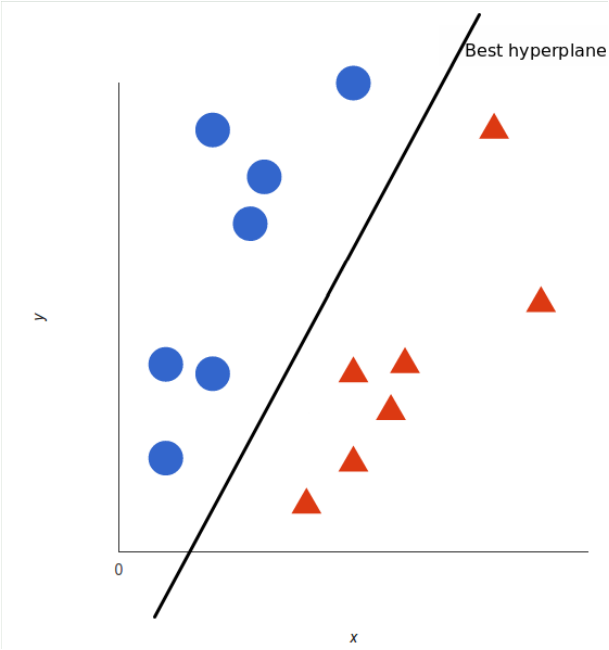


Figure 5: A simple example of the a SVM in two dimensional space. The plane distinguish between the blue and red instances [17].

4.1.5 SVM

For classification with the Support Vector Machine, hyper planes are calculated that separates between different classes[20]. The n features of an instance is mapped into a $\mathbb{R}^n$ space. The hyper plane that separates the hyper space the can be calculated by an linear or an non-linear way. The hyper plane is calculated based on the training set. A prediction of an instance of the test set is done by mapping it between the hyper planes.

4.2 Unsupervised methods

For the unsupervised methods, we can use only the network data during the training phase. The algorithm determines by itself clusters.
The disadvantage of supervised learning has disappeared with unsupervised learning. The latter method is capable of detecting anomalies that have not occurred before. However, with unsupervised learning it is difficult to find out what kind of attack it is, because it is only clustering and not classifying.

4.2.1 One-Class SVM

One-class SVM is an unsupervised version of the SVM algorithm. As the name suggested, this algorithm only distinguish between one class. whether the data is benign, or the network data is an attack. Researchers has already achieved good results with the One-Class SVM. But for data analysing with our big data set the method is too slow.

4.2.2 K-Means

K-means is an unsupervised clustering method that is proposed in the seventies and an old and well known algorithm[31, 38]. The principles behind k-means are easy to understand. For the given set of instances, the number of clusters k must choose in advance. This choice is important and can highly influence the quality of the algorithm. The algorithm start by choosing randomly k random centroids and all instances are assigned to the closest centroid. This initial choice of centroids is random and can results in non representative clusters.
After the initial choice an iterative process starts where every time new centroids are chosen for the clusters. For this centroids, new clusters are calculated by assigning the nearest instances. The algorithm ends when no objects moves to another cluster.

4.2.3 Density based

In article [40], the authors proposed DBSCAN, an unsupervised clustering method. As the acronym suggest, it stands for Density-Based Spatial Clustering of application with Noise, the clustering algorithm is density based.
This algorithm decides for every instance if it is an core-point or a border-point. An core point is when the number of instances in an neighbourhood of ϵ is higher than a certain level. Below that level the instance becomes an border-point.
For each core-point and the points within ϵ becomes in an clusters. When some clusters overlap, the clusters merges. For each point that is outside an cluster is seen as noise.

4.3 Selection of models

The list of above is a overview of the possible choices can made. In literature there exist for each type a more advanced or customized algorithm that is fine tuned for a particular problem. My choice is based on the principles behind the algorithms, and not on an comparison in depth. That is because I try to find out if it is possible to generate a model that exist out of not perfect classifiers, but performs even better with weak classifiers.
The comparison of the models is neccesary and possible, because every model has it own disadvantages and advantages. But literature is not extensive or unanimous about the these.
For our choice we state three demands:

1. **Accuracy.** For the sub model I demand that it works with enough precision.
2. **Speed.** Because of several models are combined, I want to have not a too slow model that impacted the final model.
3. **Explainability.** One big deal in machine learning and the big challenge is the complexity of the models. Because decisions are made with great impact, it is important that you know the reasons behind is reasonable.

Besides the pros and cons described in the sections above we use [5] and [19] to give an indication for our demands.
The indicators and the demands for a classification model are summarized in the table 1.
From these list we choose the Naive Bayes approach and the Decision Tree Algorithm. We found after several attempts that the unsupervised models are not suitable for our solution. For example the One-Class SVM is more an outlier methods that that is not able give explainability about the attack class an so not usable for this research. The K-means algorithm gives an low accuracy percentage. Also it is difficult to map the clusters of the unsupervised ML methods to the classes of data set. Therefor we only use the two supervised models.

Algorithm	**Complexity**	**Time**	**Note**
SVM	Medium	High	Difficult to perform on multiple classes
NB	Low	Low	
DT	Low	Medium	Perform well with large data in a short time
ANN	Medium	Medium	
K-NN	Low	High	Depends on choice of K, large storage requirements

Table 1: Summary of the demands and the indicators for five supervised learning algorithms. The algorithms are Suport Vector Machine (SVM), Naive Bayes (NB), Decision Trees (DT), k-Nearest Neighbors (k-NN) and Artificial Neural Networks (ANN)

5 Data set exploration

For this research I will use data sets that contains network traffic data. Three important requirements for a good data set:

- **Labels.** The data is labeled; such that we can verify our results;
- **Comparable.** The data set is used in other researches, so there is an possibility to compare our results.
- **State-of-the-art.** The data set is state of the art; so it represents relevant and real network traffic and attacks.

The last two criteria are conflicting. The most-wide used data set is KDD99 but more that 20 years old. And more recent data sets are little used. So we decide to use the most used data set in network anomaly detection, and take a data set that is state of the art. Both data sets will be clarified in the next subsections.

5.1 KDD99

In 1998, Defense Advanced Research Projects Agency (DARPA) and the MIT Lincoln Labs organised a contest to propose a data set that can be used for evaluation of network intrusion detection. The KDD Cup 1999 (KDD99) is a subversion of the final data set of this contest.[41]
The Lincols Labs collected nine weeks of raw TCP data from a local-area network. This network simulated an existing network where some attacks are added. The split of this data was five seven weeks training data and the remaining weeks test data.
The KDD-99 is a widely used data set. The vast majority uses it, more than 95%. For the papers I read relating to ensemble learning I found no example that used another data set. This is remarkable, because the data set is criticised on several points. This comes up in a later paragraph. First, I will discuss the attacks and the features in the data set.

5.1.1 Attacks

The attacks in the KDD-99 data set can be categorize in four types. These categories are based on the work of Kendall in [15] for the DARPA data set.
The four categories are given by:

- **Dos, Denial of service** is a attack to disrupt resources of an network by overloading the computing or memory power the resource. This is done a overload of queries that handles legitimate requests, and/or by denies access requests.
- **U2R, User to root attacks** contains attacks in which a user gains root access without the legitimate privileges.
- **R2U, Remote to user attacks** is a class of exploits where a remote user, the attacker, is sending packets to a machine in a network to gain local access.
- **Probe attacks** are scanning attacks to gather information about the network and the recourse's in it. This attack type is useful for planning future attacks.

In figure 6 is the distribution of the instances on the different classes. The main part of the data is normal data.

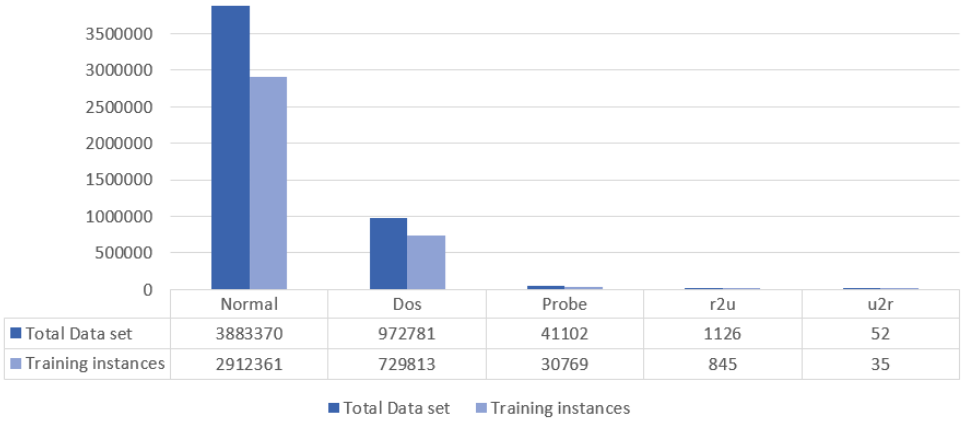


Figure 6: The distribution of the different class of the KDD99 data set.

5.1.2 Features

A network connection record contain information spread over 41 features. These features are subdivided in four categories.
The first category we discuss are the basic features of a connection:

1. **duration:** give in seconds the length of the connection;
2. **protocol type:** has the possible values of 'tcp', 'udp' and 'icmp';
3. **service:** give the network service on the destination. possible values are for example 'https', 'ftp' and 'smtp';
4. **source bytes:** the number of data bytes from source to destination;
5. **destination bytes:** the number of data bytes from destination to source;
6. **flag:** discrete value whether the connection is normal or an error status;
7. **land:** discrete value whether the connection is from the same host;
8. **wrong fragment:** give the number of wrong fragments
9. **urgent:** number of urgent packets in the connection

Now we list the content features:

10. **hot:** gives the number of hot indicators
11. **num failesd logins:** give the number of failed login attempts
12. **logged in:** discrete value whether connection has an successful logged in
13. **num compromised:** give the number of compromised conditions
14. **root_shell:** discrete value whether an root shell is obtained
15. **su attemped:** discrete value whether a sudo root command is attempted
16. **num root:** gives the number of root accesses
17. **num file creations:** give number of file creation operations
18. **num shells:** give the number of shell prompts
19. **num access files:** give the number of operations on access control files
20. **num outbound:** give the number of outbound commands in an ftp-session

21. **is hot login:** discrete value whether the login belongs to the hot list
22. **is guest login:** discrete value whether the login is an guest login.

The third category of features are about the traffic. It is computing for a time interval of two seconds.

23. **count:** number of connections to the same host as the connection in the previous time interval
24. **serror rate:** percentage of the same-host connection that return an SYN error
25. **rerror rate:** percentage of the same-host connection that return an REJ error
26. **same server rate:** percentage of the same-host connection with the same service
27. **different server rate:** percentage of the same-host connection with an different service
28. **server count:** number of connection to the same service as the connection in the previous time interval
29. **service serror rate:** percentage of the same-service connection that returns a SYN error
30. **service rerror rate:** percentage of the same-service connection that returns a REJ error
31. **service different host rate** percentage of the same-service connection that has a different host

The fourth category contains features about the host. These are given for time intervals bigger than two seconds.

32. **destination host count** give the number of connections having the same destination host
33. **destination host service count:** give the number of connection that have the same destination host and use the same service
34. **destination host same server rate** percentage of the connection with the same destination port and use the same service
35. **destination host different service rate:** percentage of the connection with the same destination port and use a different service
36. **destination host same source port rate:** percentage of the current host that have the same source port
37. **destination host service different host rate:** percentage of connection to the same services coming from different hosts.
38. **destination host serror rate:** percentage of the connection that return as S0 error
39. **destination host server serror rate** percentage of the connection that return as S0 error for a specific service
40. **destionation host rerror rate:** percentage of the connection to the current host that have an rst error
41. **destination host service rerror rate** percentage of the connection with the came host and specific service that has an RST error
42. **destination host server rerror rate:** percentage of the connection to the same host and specified servic that has an rst error

Attack Class	Important Features
Dos attacks	protocol type, service, flag, source bytes, destination bytes, land, wrong fragment, urgent, number compromised, count, srv count, serror rate, srv serror rate, rerror rate, server rerror rate, server rate different server rate, dst host host srt count, dst host same srv rate, dst host same src port rate, dst host srv serror rate, dst host rerror rate, dst host srv rerror rate.
Probe attacks	Flag, rerror rate, dst host srv diff host rate
r2l attacks	service, Source bytes, Destination bytes, urgent, failed logins, is guest login, dst host srv serror rate.
u2r attacks	flag, source bytes, destination bytes, number root.

Table 2: Important features of the KDD99 data set according [14]

Attack Class	Important Features
DOS attacks	service, flag, source bytes, same server rate, dst host serv serror rate
probe attacks	protocol type, source bytes, same server rate, destination host server count, destination host same server rate, destination host different server rate
r2l attacks	duration, service, source bytes, hot, server count, different server rate.
u2r attacks	source bytes, destination bytes, hot, number file creations, destination host server count

Table 3: Important features of the KDD99 data set according [37]

5.2 Relevant features for each attack

In literature, research is done which features are relevant for the particular attacks in the data set. In [2] the authors list mention the relevant feature classes as mentioned in the above paragraph: Basic, Content, Traffic and Host.
In [14] the author look to more specific attribute for each attack class. This is summarized in table 2. In [37] is done a similar approach, where the maximal features are reduced to 6. These features are given in table 5

5.2.1 Criticism

The first author that criticized the KDD99 dataset was John McHugh in [23]. In his article he focus on the DARPA data set, that is the basis for the KDD99 data set. His critique mainly focus on the lack of statistical evidence that the dataset contains representative data. For example, questions are raised about the distribution of the attacks and the simplicity of the simulated network.
Subsequently, [22] contains also a depth research about the difference between the simulated and real data. For example, the KDD data set missed unusual traffic as garbage data or malformed commands that are no attacks. This is relevant, because otherwise benign data that has normal faults we be detected as malicious. The authors created an advanced data set by adding some real traffic to the simulated data to solve this mentioned problem.
A less criticized data set is proposed by [39], the NSL-KDD data set that consist of selected instances of the complete KDD99 data set [27].This new data set is a reaction on two important shortcomings of the KDD99 data set. This data set still suffers from

some of the observations made by John McHugh, but is still useful for benchmark research according the authors. The first shortcoming is that is tackled three-quarters of the KDD99 data set contains redundant data that biased the outcome of the Machine Learning algorithm. This redundant data are mainly repeated instances. An second problem of the KDD99 data set is that many researchers used to take a selection of the train set instead of the test set to validate the results. This results in incomparable results between different learners and research papers.

5.3 CICDS

Since the introduction of the KDD99 data set and it criticism, there is still looked for a new state of the art data set. Several data sets are generated and used in literature. Nevertheless, KDD99 is still widely used. In 2017 researchers of the Canadian Institute for Cyber Security and the University of New Brunswick published a new intrusion detection data set [33].
The data capture the network traffic of five days: Monday till Friday. A top priority by generating this data set was the human-like behaviour of the data set. The virtual network consists out of a firewall, a web server, an Ubuntu server, several Windows and Linux systems. The attacks are initiate from a Windows and an Kali environment. On every day, except Monday, multiple attacks are launched.
In figure 7 is the distribution visible of the total data set.

5.4 Attack types

The data set contains seven main categories of attacks. In the following I will discuss them:

1. **Brute Force Attack.** This kind of attacks can be used for password cracking and discover hidden content in a web application. It is done by the principle by trying till you find useful information.
2. **Heartbleed attack.** The Heartbleed attacks used an fault in the SSL and TLS protocol. It send a (SQL) request to the server of the vulnerable party to obtain more information than necessary.
3. **Web Attack** Examples of this attacks are SQL injections and Cross-Site Scripting, where the weaknesses of a web application are misused.
4. **Infiltration Attack.** This attack is done from the inside, using flaws in vulnerable software. After exploit the attacker is for example able to do full port scan.
5. **Botnet Attack.** A botnet is a collection of (ro)bots that act independent. The owner of the botnet can use the bots for stealing data, sending spam etc.
6. **Dos Scan Attack** This attacks try to make a network or an resources unavailable by sending a lot of request to overload the system.
7. **DDos Attack.** A DDos attack is a special variant of the DOS attack, where the attacks are executed using multiple systems.

5.4.1 Relevant features

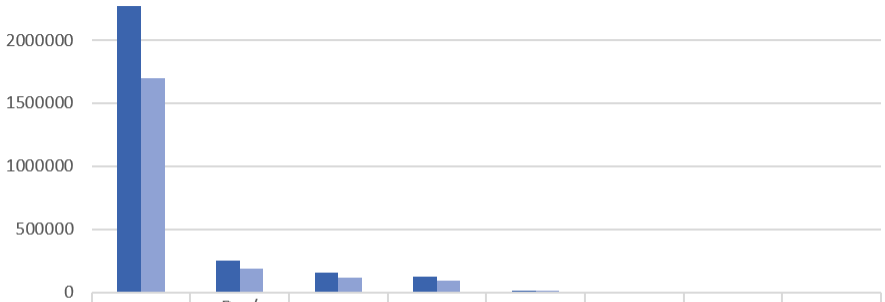
The list of features of the CICIDS2017 dataset is longer than the KDD99 data set. Therefor the overview of the features are given in the appendix.
The authors of the data set also researched which data features are the most important to each attack class. This is given in the following table 4.
Also in [36] the authors look for the important attack types.

Attack Class	Important Features
Brute Force Attack	Init_Win_bytes_forward, Fwd PSH Flags, SYN Flag Count, Flow Packets/s, Subflow Fwd Bytes, Total Length of Fwd Packets, ACK Flag Count
DoS/Heartbleed	Bwd Packet Length Std, Subflow Fwd Bytes, Total Length of Fwd Packets, Flow Duration, Fwd IAT Mean, Bwd IAT Mean, Flow IAT Mean, Active Min, Active Mean, Flow IAT Std, Flow IAT Min, FWD IAT Min
Web Attack	Init Win F. Bytes, Subflow F. Bytes, Init Win B Bytes, Total Len F. Packets
Infiltration Attack	Subflow Fwd Bytes, Total Length of Fwd Packets, Flow Duration, Active Mean
Botnet Attack	Subflow Fwd Bytes, Total Length of Fwd Packets, Fwd Packet Length Min, Bwd Packets/s
Port Scan Attack	Init_Win_bytes_forward, Bwd Packets/s, PSH Flag Count
DDoS Attack	Bwd Packet Length Std', Average Packet Size', Flow Duration, Flow IAT Std

Table 4: Important features of the CICIDS2017 data set according the authors of the data set [33]

Attack Class	Important Features
Brute Force Attack	Destination Port, Fwd Packet Length Std, min_seg_size_forward
DoS/Heartbleed	Destination Port, Total Length of Bwd Packets, Init_Win_bytes_forward, Idle Std
Web Attack	Fwd Packet Length Mean, Fwd IAT Mean, Init_Win_bytes_backward
Infiltration Attack	Total Length of Fwd Packets, Active Std, Active Min, Idle Std
Botnet Attack	Destination Port, Bwd Packet Length Mean,Bwd Packet Length Std, min_seg_size_forward
Port Scan Attack	Bwd Packet Length Mean, PSH Flag Count, Init_Win_bytes_backward, act_data_pkt_fwd, min_seg_size_forward
DDoS Attack	Destination Port, Fwd Packet Length Max, RST Flag Count, Active Std

Table 5: Important features of the CICIDS2017 data set according [36]



	Normal	Dos/ Heartbleed	Botnet	DOSS	Web	Port	Infiltration	Bruteforce
Total Data set	2272688	251723	158930	128027	13835	2180	1966	36
Training instances	1704300	188986	119408	95960	10292	1625	1446	21

Figure 7: The distribution of the different class of the CICIDS2017 data set.

6 Results

6.1 Set up experiments

For this thesis, I created several algorithmic ensemble sets with Python. For every individual model we design a data pipe line. In this pipeline the data is selected on basis of features. There after the data is normalized to numerical values and if necessary to an one-hot-encoded form.
By applying we use the implementation from the sci-learn package [29]. The reason we choose for this type of implementation is that we do not need special fine tuned models, we are satisfied with weak learners. And besides that, the manual implementation of these algorithms can be labour-intensive.

6.2 Representation of results

The basic measurements about the correctness of a classification model focus on the well-predicted elements. These elements are the True Positives (TP). This measurement, the accuracy is calculated by:

$$\text{Accuracy} = \frac{\text{TP}}{\#\text{All elements}}.$$

This score is not sufficient, because the behaviour of misclassified elements impacts the usability of a model. For example, you want that the positive classified elements represents the reality. So we can measure the precision by:

$$\text{precision} = \frac{\text{TP}}{\text{TP} \cup \text{FP}}.$$

where FP are the False Positives.
Another measurements focus on the missing items of the prediction. This is given by:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} \cup \text{FN}}.$$

where FP are the False Negatives.
An schematic overview of the mentioned data selections is given in 8. The overview of the results of the True positives, True Negatives, False Negatives, False Positives, can be given in the confusion matrix. For our research it is not useful, because we compare a lot of models whereby it is impossible to consider every confusion matrix.
Therefor we focus on the measurements of the precision and recall. Both can be combined in the F1-score. This score integrate both the precision and recall by

$$\text{F}_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{5}$$

6.3 Results - KDD

As explained in section 3 we train in total 8 models, four models based on Naive Bayes, for each attack class; and similar four models for the Decision Tree. The accuracy and the individual F_1-score of each model can be found in table 6.
For the individual models we can state the following observations:

1. The Naive Bayes are preferred to distinguish attack and non-attack data. They are perform well on classifying normal data, but lack on distinguish between the attacks.

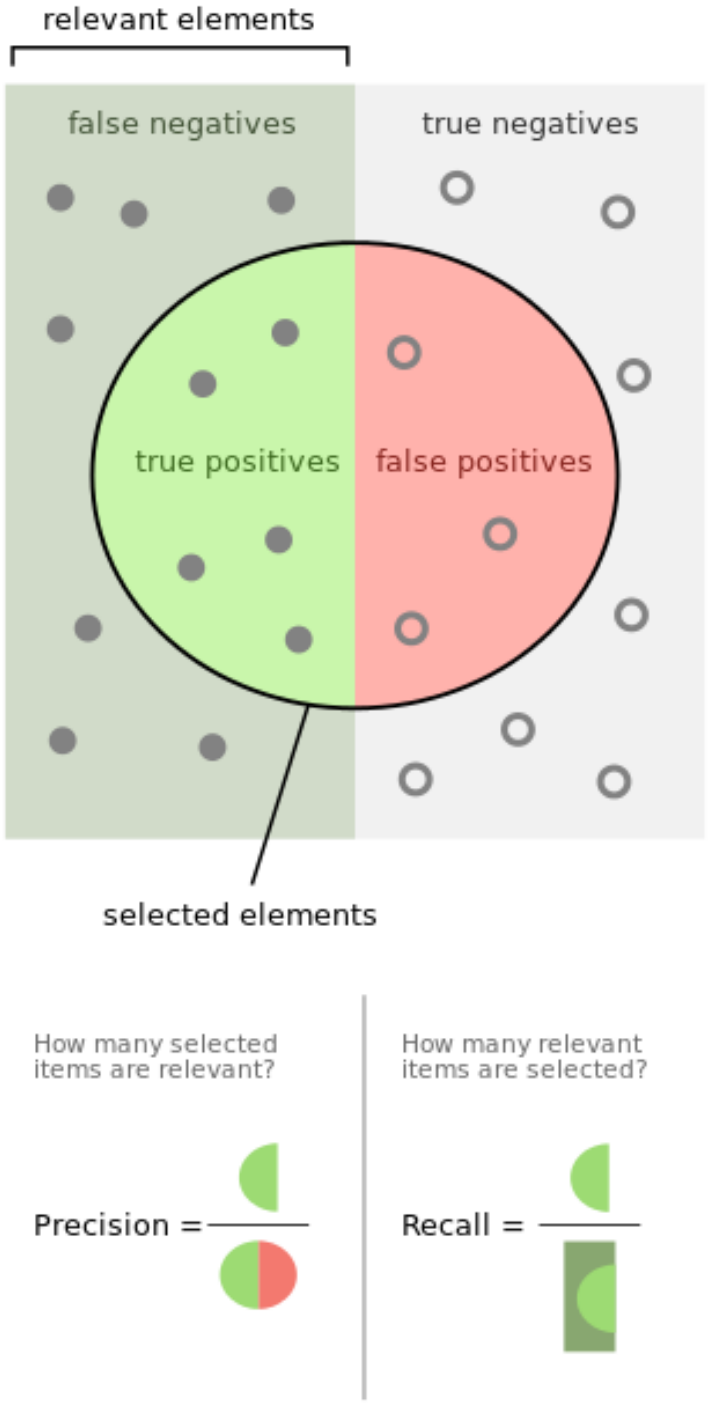


Figure 8: Schematic representation the data set separated in false negatives and positive and true negatives and positives.

#	ML	Features	Acc.	F_1 score Normal	Dos	Probe	U2R	R2U
1	Naive Bayes	Probe	0,80	0,89	0,03	0	0	0
2	Naive Bayes	Dos	0,80	0,89	0,03	0	0	0
3	Naive Bayes	U2R	0,80	0,89	0,03	0	0	0
4	Naive Bayes	R2U	0,87	0,93	0,57	0	0	0
5	Decision Tree	Probe	0,91	0,96	0,89	0,56	0,02	0
6	Decision Tree	Dos	0,73	0,83	0,85	0,07	0,03	0
7	Decision Tree	U2R	0,74	0,82	0,87	0,07	0,09	0
8	Decision Tree	R2U	0,97	0,99	0,96	0,59	0,06	0

Table 6: Individual models for the KDD99 data set. The accuracy of the whole test set and the F_1 score of the individual attacks.

Ensemble	Accuracy	F_1 score Normal	Dos	Probe	U2R	R2U
7,8	0.977	0.99	0.97	0.58	0.07	0
5,7,8	0.975	0.99	0.96	0.58	0.05	0.01
6, 8	0.973	0.99	0.96	0.58	0.05	0.0
5,6,8	0.973	0.99	0.96	0.58	0.04	0.01
3, 6, 8	0.97	0.99	0.95	0.58	0.05	0.01
1, 5,6,8	0.97	0.99	0.95	0.58	0.05	0.01
1, 5, 7, 8	0.97	0.99	0.95	0.58	0.08	0.01
1,2,6,7,8	0.94	0.96	0.84	0.53	**0.58**	0.01
3,4,6,7,8	0.943	0.97	0.85	0.53	0.36	0.01
2, 3, 6, 7, 8	0.94	0.96	0.84	0.53	**0.59**	0.01

Table 7: Selection of the best ensemble models with soft-voting. The accuracy of the whole data set and the F_1 score of the individual attacks.

2. The Decision Tree models performs better that the Naive Bayes to classify all classes.
3. For both models are the low present attacks difficult to identity. For the R2U attacks this is nearly never successful. For U2R Attack in a few cases. The probe attack is identified only for model 5 and 8 in table 6 with a F_1 score for around 0,59.

After the training we look for an optimal sub-ensemble. We look at each combination equal to five or less for the soft-voting technique of ensemble. This are $\binom{8}{1} + \binom{8}{2} + \binom{8}{3} + \binom{8}{4} + \binom{8}{5} = 218$ combinations. In table 7 a selection of the best ensembles for soft voting are given. From this selection we can give the following observations:

1. In general the ensemble methods performs as good as the best individual model, number 8. The accuracy is slightly higher.
2. The R2U attacks are still not detected, but for two ensembles the U2R are much better detected. The cost is 0.03 accuracy and 0.1 of the F_1 score in comparison with other ensembles.

To save computation time, we calculate for only the models with length equal or less than 5, with accuracy higher than 0,9 the results of hard voting, Neural Network ensemble and decision tree. The selection of the results are given in table 12, 13 and 11. We can give the following observations:

						F_1 score					
#	ML	Feat.	Acc.	Bot	BF.	DDos	Dos	Inf.	Normal	Port	Web
1	DT	BF.	0.642	0.03	0.72	0.35	0.65	0.76	0.74	0.55	0.14
2	DT	Bot	0.686	0.02	0.78	0.59	0.59	0.76	0.77	0.55	0.17
3	DT	DDos	0.524	0.01	0.76	0.22	0.64	0.76	0.65	0.55	0.01
4	DT	Inf.	0.686	0.02	0.78	0.59	0.59	0.76	0.77	0.55	0.17
5	DT	Port	0.685	0.02	0.73	0.36	0.81	0.76	0.78	0.59	0.11
6	DT	Probe	0.642	0.02	0.72	0.35	0.65	0.76	0.74	0.69	0.14
7	DT	Web	0.707	0.01	0.73	0.97	0.56	0.76	0.78	0.85	0.11
8	NB	BF.	0.804	0	0	0	0	0	0.89	0	0
9	NB	Bot	0.804	0	0	0	0	0	0.89	0	0
10	NB	DDos	0.193	0	0	0.11	0	0	0.3	0	0
11	NB	Inf.	0.193	0	0	0.11	0	0	0.3	0	0
12	NB	Port	0.193	0	0	0.11	0	0	0.3	0	0
13	NB	Probe	0.193	0	0	0.11	0	0	0.3	0	0
14	NB	Web	0.044	0	0.01	0.01	0.02	0.	0	0.09	0.04

Table 8: Individual models for the CICDS2017 data set. The accuracy of all the test data, and the F_1 score for the individual attacks.

1. The hard voting approach give not better results, the accuracies and F_1-scores are even worse.
2. The machine learning approaches using Decision Trees and Neural Networks has acceptable accuracies. But both the lack on the F_1-scores of the individual attacks.

6.4 Results - CICDS

As explained in section 3 I train in total 14 models. Four models based on Naive Bayes, for each attack class; and similar four models for the Decision Tree. The overall accuracy and the individual F_1-score of each model can be found in table 8.
For the individual models I can state the following observations:

1. Two models of the Naive Bayes approach are preferred to distinguish attack or non attack data (number 8 and 9). They are perform well on classifying normal data, but lack on distinguish between the different attacks. The other 5 models do not perform very well.
2. The Decision Tree models performs better that the Naive Bayes to classify all classes. The relative accuracy of all classes is quite low.
3. For both models, the low present attacks are difficult to identity. For the Web and Botnet attacks this is nearly never done.

After the training I look for an optimal subensemble. I look at each combination equal to five or less for the soft-voting technique of ensembling. This are $\binom{14}{1} + \binom{14}{2} + \binom{14}{3} + \binom{14}{4} + \binom{14}{5} = 3472$ combinations. In table 9 a selection of the best ensembles for soft voting are given. From this selection we can give the following observations:

1. In general the accuracies of the selected models are much better.
2. For four options of the selected models the Botnet attack is increasingly better detected. But the F1 score is still between 0.29 and 0.42.
3. For two models the F_1 score of the Web attacks are much higher.

Ensemble	Acc.	Bot	BrF.	DDos	Dos	F_1 score Inf.	Normal	Port	Web
3, 7, 8, 9, 14	0.937	**0.29**	0.77	0.99	0.68	0.69	0.96	0.91	0.48
1, 5, 7, 8, 9	0.924	0.04	0.73	0.99	0.91	0.76	0.95	0.89	0.11
4, 7, 8, 9, 14	0.924	0.06	0.8	0.92	0.77	0.69	0.95	0.91	**0.72**
2, 7, 8, 9, 14	0.924	0.06	0.8	0.92	0.77	0.69	0.95	0.91	**0.72**
2, 7, 8, 9	0.923	0.05	0.8	0.98	0.77	0.76	0.95	0.91	0.27
4, 7, 8, 9	0.923	0.05	0.8	0.98	0.77	0.76	0.95	0.91	0.27
5, 7, 8, 9, 13	0.921	0.04	0.48	0.99	0.75	0.53	0.95	0.91	0.04
5, 6, 7, 8, 9	0.922	0.04	0.73	0.99	0.91	0.76	0.95	0.9	0.11
5, 6, 8, 9, 10	0.921	0.04	0.48	0.99	0.75	0.53	0.95	0.91	0.04
5, 7, 8, 9, 11	0.921	0.04	0.48	0.99	0.75	0.53	0.95	0.91	0.04
5, 7, 8, 9, 12	0.921	0.04	0.48	0.99	0.75	0.53	0.95	0.91	0.04
3, 7, 8, 9, 10	0.911	**0.42**	0.5	0.99	0.68	0.53	0.95	**0.59**	0
3, 7, 8, 9, 11	0.911	**0.42**	0.5	0.99	0.68	0.53	0.95	**0.59**	0
3, 7, 8, 9, 13	0.911	**0.42**	0.5	0.99	0.68	0.53	0.95	**0.59**	0

Table 9: Selection of the best ensemble models with soft-voting for the CICIDS data set. The accuracy of the whole test set and the F_1 score of the individual attacks.

4. For all of the other attack types there is for the most of the selected models an better result. Only for the Infiltration attack the result stay the same of is slightly worse.

For the 3472 combinations with number of models equal or less than 5, there are 44 that has an accuracy higher than 0.90. For these combinations I look at ensemble methods as hard voting, neural networks and decision trees. From this methods I make the following observations:

1. For the hard voting technique we found that for all 44 models has the same result. The accuracy is 0.80 and this is mainly due to the right predicted elements for the normal class.
2. For the Decision tree the accuracies of the 44 models are quite low. The highest accuracy is 0.15. The classes that has a F_1 score more than 0.80 are the Bruteforce and DDOS class.
3. For the Neural Network approach, for a selected number of classes I found that the F_1 is high, see table 14 in the appendix. But for the low present attacks, the F_1 is zero.

7 Conclusions

In this chapter, I summarise the multiple models and data sets comparison and identify the most effective way to detect network intrusions with a number of off the shelf-untuned models in an ensemble. First, I will draft a conclusion about the models created, and compare the difference between the two data sets.
In the final section, I will the subject of this thesis set in a broader context.

7.1 Ensemble learning

In the previous chapter I give some observations of the ensemble learning models. Now, I will discuss the results for both data sets. This is quite useful, because this give also an indication how comparable both data sets are.

7.1.1 Individual models

For both data sets, I observe the same trends if I compare the models of the Decision Trees and Naive Bayes. The Naive Bayes models has an overall better accuracy in comparison with the Decision Tree approaches. Its best benefit is detecting normal behaviour. The Decision Tree models has an lower accuracy, but are much better in classifying the different attacks.
How can I explain the results of the individual models? For the Decision Trees we see a comparable results with literature. Also the low F_1-score for low present attacks is repeatedly visible in literature. Because these kind of attacks can be seen as just noise instead of intrusion events.
Improvement of the individual models for Decision Trees can be done by selecting more features, or choosing PCA as encoding mechanism. Both we did not to preserve the possibility to give some explainable opportunities.
For Naive Bayes I found some differences with literature. For most proposals in literature the F_1-scores are more equal distributed over the different classes, in stead of one peak for the normal class. This can have some reasons. The first reason can be the impact of the feature selection. Another options is the influence of the simple encoding strategy of our approach. Small experiments to enhance the results of the models show that as well the feature selection as the encoding method has effect on the results that improves the different F_1-scores. A third option can be the implementation of the Naive Bayes, we use the one of the build in library of sci-learn. Other papers made use of manual implementation of the Machine Learning or another platform called WEKA that is created for this kind of experiments.

7.1.2 Ensemble learning

Notwithstanding the low expectations of the results of the Naive Bayes models, they still has impact on the results of the ensemble models.
For both data sets similar things happens. If we concentrate first on the soft voting way of combining the individual models, we found several combinations that has both a higher accuracy and better F_1-scores. For a selection of these, also the F_1-score of one of the low present attacks is increasing. This last is unexpectedly because the first though was that the previous models did not learn the low present attacks.
The soft voting is the only ensemble learning way that increases the F_1-scores and the accuracy. TBased on the data and my research, the reason is: soft voting deals better with the subtle results of the individual models. These nuance fall away if we use for example hard voting. Multi Layer Perceptron (MLP) or Neural Networks focus to much

on overall accuracy and ignore the smaller parts. Therefor it is reasonable that the F_1-score for low present attacks are low.

7.2 Explainable results

In the chapters before, I stressed several times the importance of the opportunity to understand the results. After the prediction of an intrusion detection model, an important decision is made. Often, network intrusion alerts are handled in an automated or semi automated fashion. False positives can lead to security operations alert fatigue or in the worst cases, incorrectly terminated network connections causing a loss of business functionality. Therefore, not only should it be clear how certain the prediction is, but also where it is based on, so that it is possible to verify the decision.
In order to make it possible to understand the results, I made the following decisions. First of all, I have chosen to train different modules, each with a different selection of features. Unfortunately there is no connection between the quality of the predictions of a certain class and a model with features of this class. It is not that a model that has as input the most important features of the attack technique DOS also performs significantly better on the DOS attack.
Another choice I made to improve the explainability of the results is not to choose for PCA encoding. By just scaling the chosen features and converting them into numbers, as much of the raw data as possible is saved. This way I hope it will be possible to quickly calculate which features are important and see their value without having to decode the values.
How the latter approach will work out in practice is an interesting subject for further research. Given my research, the following approach would be a possibility. Because I use ensemble learning, we have to go back to the individual models. Given a prediction, we select the model(s) that best predicted this output. For each of these individual models we can see which features had the most influence on the prediction. This can be done by means of back propagation and gradient calculation. In this way, both the prediction and the causes (important features) can be shown at the same time, which can facilitate a follow-up selection.

7.3 Broader context

A network intrusion detection as discussed in the investigation is fortunately not the only defensive weapon against unwanted intruders. The Mitre Att&ck framework mentions nine different ways to get initial access in a network [1]. This ranges from phishing attacks to exploiting public applications.
The positive thing is that there are more tools to keep intruders out. The negative is that a lot of information is not used in the discussed approach. To come back to the analogy from the introduction, we make a 2d image with this way of ensemble learning. From one position we look at the same data in different ways. We always take the same data information: duration, type and other information about the connection.
An improvement of our approach that can have a significant impact is not to create a 2d but a 3d image. This means that the data must be viewed differently, not from the same point of view. Information needs to be collected at different points in the network, and decisions need to be made again and again. From the first access at the firewall to actions at the endpoints.

8 Appendix A: features CICDS

In table 10 an overview of the 79 label features are given.
The first two values give the destination port of the connection and the duration of the connection flow. Features 3 up to 14 give information about the length of all and the individual packets. Feature 15, 16, 37 and 38 give the number of bytes and packet per second of the connection flow and the Forwarded and backwarded packets.
Features 17 up to 30 give numerical values of the Inter Arrival Time (IAT) of the flow, the Forwarded and backwarded packets.
Numbers 31 upto 34 and 44 up to 51 is information about the different flags that relates to the network connection.

Feature	**Feature**	**Feature**
1. Destination Port	28. Bwd IAT Std	55. AvgBwd Segment Size
2. Flow Duration	29. Bwd IAT Max	56. Fwd Header Length
3. Total Fwd Packets	30. Bwd IAT Min	57. FwdAvg Bytes/Bulk
4. Total Backward Packets	31. Fwd PSH Flags	58. FwdAvg Packets/Bulk
5. Total Length of Fwd Packets	32. Bwd PSH Flags	59. FwdAvg Bulk Rate
6. Total Length of Bwd Packets	33. Fwd URG Flags	60. BwdAvg Bytes/Bulk
7. Fwd Packet Length Max	34. Bwd URG Flags	61. BwdAvg Packets/Bulk
8. Fwd Packet Length Min	35. Fwd Header Len	62. BwdAvg Bulk Rate
9. Fwd Packet Length Mean	35. Bwd Header Length	63. SubflowFwd Packets
10. Fwd Packet Length Std	37. Fwd Packets/s	64. SubflowFwd Bytes
11. Bwd Packet Length Max	38. Bwd Packets/s	65. SubflowBwd Packets
12. Bwd Packet Length Min	39. Min Packet Length	66. SubflowBwd Bytes
13. Bwd Packet Length Mean	40. Max Packet Length	67. Init_Win_bytes_forward
14. Bwd Packet Length Std	41. Packet Length Mean	68. Init_Win_bytes_backward
15. Flow Bytes/s	42. Packet Length Std	69. act_data_pkt_fwd
16. Flow Packets/s	43. Packet Length Variance	70. min_seg_size_forward
17. Flow IAT Mean	44. FIN Flag Count	71. Active Mean
18. Flow IAT Std	45. SYN Flag Count	72. Active Std
19. Flow IAT Max	46. RST Flag Count	73. Active Max
20. Flow IAT Min	47. PSH Flag Count	74. Active Min
21. Fwd IAT Total	48. ACK Flag Count	75. Idle Mean
22. Fwd IAT Mean	49. URG Flag Count	76. Idle Std
23. Fwd IAT Std	50. CWE Flag Count	77. Idle Max
24. Fwd IAT Max	51. ECE Flag Count	78. Idle Min
25. Fwd IAT Min	52. Down/Up Ratio	79. Label
26. Bwd IAT Total	53. Average Packet Size	
27. Bwd IAT Mean	54. AvgFwd Segment Size	

Table 10: List of feature of the CICIDS2017 data set.

9 Appendix B: results

			F_1 score			
Ensemble	Accuracy	Normal	Dos	Probe	U2R	R2U
3, 4, 6, 7, 8	0.978	0.99	0.95	0	0	0
1, 3, 8	0.989	0.99	0.99	0	0	0
3, 7, 8	0.989	0.99	0.99	0	0	0
2, 4, 5, 6, 8	0.989	0.99	0.99	0.	0	0
4, 5, 7	0.988	0.99	0.99	0	0	0
2, 5	0.974	0.98	0.95	0	0	0
5, 7, 8	0.978	0.99	0.95	0	0	0

Table 11: Selection of the best ensemble models with an artificial neural Network for the KDD99 dataset, the accuracy of the test set and F_1 score of the individual attacks.

			F_1 score			
Ensemble	Accuracy	Normal	Dos	Probe	U2R	R2U
3, 4, 5, 7, 6	0.905	0.96	0.89	0.56	0.02	0
1, 3, 5, 6, 7	0.905	0.96	0.89	0.56	0.02	0
2, 3, 5, 6, 7	0.905	0.96	0.89	0.56	0.02	0
3, 5, 6, 7	0.905	0.96	0.89	0.56	0.02	0
3,5, 6	0.905	0.96	0.89	0.56	0.02	0
2, 4, 5, 7, 6	0.905	0.96	0.89	0.56	0.02	0
5, 7	0.905	0.96	0.89	0.56	0.02	0

Table 12: Selection of the best ensemble models with decision trees for the KDD99 dataset, the accuracy of the test set and F_1 score of the individual attacks.

Ensemble	Accuracy	Normal	F_1 score Dos	Probe	U2R	R2U
5,6,7,8	0.975	0.99	0.96	0.59	0.06	0
5, 6, 7	0.905	0.96	0.89	0.56	0.02	0
5, 6, 8	0.975	0.99	0.96	0.59	0.06	0
6, 5, 8	0.975	0.99	0.96	0.59	0.06	0
5, 7, 8	0.975	0.99	0.96	0.59	0.06	0
5, 8	0.975	0.99	0.96	0.59	0.06	0
5, 7	0.905	0.99	0.89	0.56	0.02	0
5, 6	0.905	0.99	0.89	0.56	0.02	0
7, 8	0.975	0.99	0.96	0.59	0.06	0
6, 8	0.975	0.99	0.96	0.59	0.06	0

Table 13: Selection of the best ensembled models with hardvoting for the KDD99 dataset, the accuracy of the test set and F_1 score of the individual attacks.

Ensemble	Acc.	Bot	BF.	DDos	Dos	F_1 score Inf.	Normal	Port	Web
4, 5, 6, 7, 8	0.966	0	0	0.99	0.90	0	0.98	0.92	0
0, 4, 6, 7, 8	0.961	0	0	0.99	0.9	0	0.98	0.87	0
1, 4, 6, 7, 8	0.958	0	0	0.99	0.85	0	0.98	0.91	0
3, 4, 6, 7, 8	0.958	0	0	0.99	0.85	0	0.98	0.91	0
4, 6, 7, 8	0.953	0	0	0.98	0.81	0	0.97	0.91	0
2, 4, 6, 7, 8	0.942	0	0	0.98	0.77	0	0.97	0.84	0
4, 6, 7, 8, 10	0.942	0	0	0.99	0.81	0	0.96	0.82	0
4, 6, 7, 8, 11	0.942	0	0	0.99	0.81	0	0.96	0.82	0
4, 6, 7, 8, 9	0.942	0	0	0.99	0.81	0	0.96	0.82	0
4, 6, 7, 8, 12	0.942	0	0	0.99	0.81	0	0.96	0.82	0

Table 14: Selection of the best ensembled models with an artificial neural Network for the cICIDS2017 dataset, the accuracy of the test set and F_1 score of the individual attacks.

References

[1] Mittre att&ck. `https://attack.mitre.org/`, 2020. [Online; accessed 15-July-2020].

[2] Preeti Aggarwal and Sudhir Kumar Sharma. Analysis of kdd dataset attributes-class wise for intrusion detection. *Procedia Computer Science*, 57:842–851, 2015.

[3] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[4] Saleh Alaliyat. Video-based fall detection in elderly's houses - scientific figure on researchgate. `www.researchgate.net`. [Online; accessed 29-July-2020].

[5] Amira Sayed A Aziz, EL Sanaa, and Aboul Ella Hassanien. Comparison of classification techniques applied for network intrusion detection and classification. *Journal of Applied Logic*, 24:109–118, 2017.

[6] Ali Borji. Combining heterogeneous classifiers for network intrusion detection. In *Annual Asian Computing Science Conference*, pages 254–260. Springer, 2007.

[7] Facundo Bre. Prediction of wind pressure coefficients on building surfaces using artificial neural networks - scientific figure on researchgate. `https://www.unb.ca/cic/datasets/nsl.html`, 2017. [Online; accessed 29-July-2020].

[8] Varun Chandola. Anomaly detection: A survey, 2007.

[9] Mei-Hung Chiu. Basic structure of a decision tree - scientific figure on researchgate. `www.researchgate.net`. [Online; accessed 29-July-2020].

[10] Abdoulaye Diop, Nahid Emad, Thierry Winter, and Mohamed Hilia. Design of an ensemble learning behavior anomaly detection framework. *International Journal of Computer and Information Engineering*, 13(10):551–559, 2019.

[11] Mohamed Amine Ferrag, Leandros Maglaras, Sotiris Moschoyiannis, and Helge Janicke. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419, 2020.

[12] Gianluigi Folino and Pietro Sabatino. Ensemble based collaborative and distributed intrusion detection systems: A survey. *Journal of Network and Computer Applications*, 66:1–16, 2016.

[13] Xianwei Gao, Chun Shan, Changzhen Hu, Zequn Niu, and Zhen Liu. An adaptive ensemble machine learning model for intrusion detection. *IEEE Access*, 7:82512–82521, 2019.

[14] H Günes Kayacik, A Nur Zincir-Heywood, and Malcolm I Heywood. Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets. In *Proceedings of the third annual conference on privacy, security and trust*, volume 94, pages 1723–1722, 2005.

[15] Kristopher Kristopher Robert Kendall. *A database of computer attacks for the evaluation of intrusion detection systems*. PhD thesis, Massachusetts Institute of Technology, 1999.

[16] Kurniabudi Kurniabudi, Benni Purnama, Sharipuddin Sharipuddin, Darmawijoyo Darmawijoyo, Deris Stiawan, Samsuryadi Samsuryadi, Ahmad Heryanto, and Rahmat Budiarto. Network anomaly detection research: a survey. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 7(1):37–50, 2019.

[17] James Le. Datacamp - support vector machines tutorial. `datacamp.com`, 2015. [Online; accessed 29-July-2020].

[18] Yihua Liao and V Rao Vemuri. Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*, 21(5):439–448, 2002.

[19] Hongyu Liu and Bo Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20):4396, 2019.

[20] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[21] United States Cyber Security Magazine. Cyber attack: The only safe computer is a "dead" computer. `https://www.uscybersecurity.net/cyber-attack/`, 2015. [Online; accessed 20-June-2020].

[22] Matthew V Mahoney and Philip K Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *International Workshop on Recent Advances in Intrusion Detection*, pages 220–237. Springer, 2003.

[23] John McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294, November 2000.

[24] Ali H Mirza. Computer network intrusion detection using various classifiers and ensemble learning. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2018.

[25] Sara Mohammadi, Hamid Mirvaziri, Mostafa Ghazizadeh-Ahsaee, and Hadis Karimipour. Cyber intrusion detection by combined feature selection algorithm. *Journal of information security and applications*, 44:80–88, 2019.

[26] Mehdi Moradi and Mohammad Zulkernine. A neural network based system for intrusion detection and classification of attacks. In *Proceedings of the IEEE International Conference on Advances in Intelligent Systems-Theory and Applications*, pages 15–18, 2004.

[27] University of New Brunswick. Nsl-kdd dataset. `https://www.unb.ca/cic/datasets/nsl.html`, 2019. [Online; accessed 24-June-2020].

[28] Mrutyunjaya Panda and Manas Ranjan Patra. Network intrusion detection using naive bayes. *International journal of computer science and network security*, 7(12):258–263, 2007.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] Sornxayya Phetlasy, Satoshi Ohzahata, Celimuge Wu, and Toshihito Kato. A sequential classifiers combination method to reduce false negative for intrusion detection system. *IEICE Transactions on Information and Systems*, 102(5):888–897, 2019.

[31] Leonid Portnoy. *Intrusion detection with unlabeled data using clustering*. PhD thesis, Columbia University, 2000.

[32] Accenture Security. Ninth annual cost of cybercrime study. `https://www.accenture.com/us-en/insights/security/cost-cybercrime-study`, 2019. [Online; accessed 24-July-2020].

[33] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116, 2018.

[34] BS Sharmila and Rohini Nagapadma. Intrusion detection system using naive bayes algorithm. In *2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, pages 1–4. IEEE, 2019.

[35] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, Miami University, 2003.

[36] Shailesh Singh Panwar, YP Raiwani, and Lokesh Singh Panwar. Evaluation of network intrusion detection with features selection and machine learning algorithms on cicids-2017 dataset. In *International Conference on Advances in Engineering Science Management & Technology (ICAESMT)-2019, Uttaranchal University, Dehradun, India*, 2019.

[37] Ralf C Staudemeyer and Christian W Omlin. Extracting salient features for network intrusion detection using machine learning methods. *South African computer journal*, 52(1):82–96, 2014.

[38] Iwan Syarif, Adam Prugel-Bennett, and Gary Wills. Unsupervised clustering approach for network anomaly detection. In *International conference on networked digital technologies*, pages 135–145. Springer, 2012.

[39] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. IEEE, 2009.

[40] Tran Manh Thang and Juntae Kim. The anomaly detection by using dbscan clustering with multiple parameters. In *2011 International Conference on Information Science and Applications*, pages 1–5. IEEE, 2011.

[41] Unknown. KDD. `https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Tasks`, 1999. [Online; accessed 5-June-2020].