Computer Science

Henrik Johansson

Video Flow Classification

Feature Based Classification Using the Tree-based Approach

Degree Project for Master of Science in Engineering, Computer Science
June, 2016

Video Flow Classification

Feature Based Classification Using the Tree-based Approach

Henrik Johansson

This dissertation is submitted in partial fulfillment of the requirements for Master of Science in Engineering degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Henrik Johansson

Approved,

Advisor: Johan Garcia

Examiner: Thijs Holleboom

Abstract

This dissertation describes a study which aims to classify video flows from Internet network traffic. In this study, classification is done based on the characteristics of the flow, which includes features such as payload sizes and inter-arrival time. The purpose of this is to give an alternative to classifying flows based on the contents of their payload packets. Because of an increase of encrypted flows within Internet network traffic, this is a necessity. Data with known class is fed to a machine learning classifier such that a model can be created. This model can then be used for classification of new unknown data. For this study, two different classifiers are used, namely decision trees and random forest. Several tests are completed to attain the best possible models. The results of this dissertation shows that classification based on characteristics is possible and the random forest classifier in particular achieves good accuracies. However, the accuracy of classification of encrypted flows was not able to be tested within this project.

Acknowledgements

I would like to thank Procera Networks for the opportunity to work on this project and a special thanks to Anders Waldenborg who was my supervisor at the company. Another thanks go out to the Swedish National Infrastructure for Computing (SNIC), for providing the necessary resources for this study. A thanks also goes out to Simon Westlinder for the work done on the project and for the work done using the SVM approach. Finally, a thanks goes out to Johan Garcia, who was my supervisor at Karlstad University for support in the entire process, both on the project and on the dissertation

Contents

List of Figures

List of tables

1 Introduction

Currently, there is a trend towards an increasing amount of Internet traffic becoming encrypted. This creates challenges for companies which are earning their living through producing software which can categorize flows. They can do this by using a technique called Deep Packet Inspection (DPI). This technique need to be able to read the content of the payload in order to predict the service or type of service which is used. With encrypted data flow, the DPI technology cannot read the payload and as such, cannot make predictions on payload content. A different way of categorizing the flows, which do not rely on the content of the payload, is needed.

The characteristics of an Internet traffic flow can be extracted in order to predict the type of service which is used. To do this, a model has to be created which takes the extracted characteristics as input and produces a category prediction as output.

In order to create this model, machine learning classifiers are used to train on data where the output is known, in order to produce results on data which is unknown. Producing an as accurate model as possible is paramount.

This project was specified by Procera Networks [1]. Procera Networks supplied data with known services.

1.1 Motivation

Procera [1] devices are embedded with the PacketLogic software which includes the Network Application Visibility Library (NAVL). NAVL is a real-time Deep Packet Inspection (DPI) software engine. In DPI technology, all seven layers of the OSI reference model can be monitored, including the data content of the IP packet which can be used in order to categorize the traffic [2]. However, an increasing fraction of Internet traffic is being encrypted and by the end of 2016, it is estimated that up to 70 percent of Internet traffic will be encrypted [3]. Encryption severely limits techniques which categorize traffic based on payload content such as DPI. This is a major motivation for this study. With the help of statistical techniques involving features of the individual traffic flows, classification can be achieved without parsing the information contained within the payload of packets. Deploying the statistical classification technique alongside of NAVL may increase the accuracy of the classification process as it can give the ability to categorize flows which could previously not be categorized.

1.2 Expected Results

The expected results can be divided into two subsections, my expectations and the expectations of Procera Networks.

1.2.1 My Expectations

At the beginning of the study, there was little to no knowledge to base an expectation. Based on the research which was done at the first few weeks of the study, expectations that the final model might classify video flows with an accuracy of above 90 % and possibly even above 95 % arose. At that point, it was yet unknown if it was an optimistic expectation or a realistic one.

1.2.2 Procera Networks Expectations

Procera did not express an expectation of accuracy or any other metric. The one thing they did express, was that they wanted the model to be able to classify some flows which their current software could not, due to encryption.

1.3 Division of Work

This study is part of a common project. The project is divided into two different approaches, namely decision trees and support vector machines. The decision tree approach is investigated in this thesis, while Simon Westlinder investigated the support vector machine approach. By looking at two different approaches, the chance of finding a good model for classification is improved.

The results from the two different approaches will then be compared to evaluate the better approach for this specific problem.

1.4 Summary of the Results

This is a short summary of the results which can be seen in full detail in chapter 5.

	Decision Tree	**Random Forest**	**SVC-RBF**
Accuracy	0.942	0.965	0.914

Table 1.1: Accuracy of different classifiers

As can be seen in Table 1.1, the random forest classifier achieves the greatest accuracy of classification. It is followed by the decision tree classifier and the SVC classifier. These results suggest that only based on accuracy of the classification, random forest is the best classifier of the evaluated classifiers. It also suggests that the tree approach is better than the SVM approach for this specific problem.

1.5 Dissertation Layout

This section presents a brief overview of the dissertation layout.

Chapter 2 gives some background on machine learning. It gives information about machine learning itself, but also how to use it and how to understand the results of a classification process. It aims to give the reader the necessary knowledge in order to fully understand the rest of the dissertation.

Chapter 3 summarizes some of the work that has previously been done on the subject. Reading previous studies help in understanding some of the choices that has been taken. It also gives the reader a starting point for additional experiments.

Chapter 4 follows the process step by step from receiving the data to getting results from the classifier.

Chapter 5 shows the results of the project. Different tests are done to attempt to achieve the highest possible accuracy. When the best possible requirements for achieving the greatest accuracy has been found out, a result using this information is presented. These results are then compared between the different classifiers.

Chapter 6 concludes the dissertation with an evaluation of the study as a whole and some discussion of what the results actually means.

2 Machine Learning

The purpose of this study is to create a model which can classify flows just by looking at the characteristics of the flow. This model will be created by using a machine learning classifier. To get a better understanding of how this works, an introduction is made to machine learning in general and especially to the decision tree and random forest classifiers which are the classifiers chosen for this study.

In order to create the model, data is fed to the machine learning classifier, which it uses for training. Once training is complete, the classifier can be used to predict results on new data. Tests are done to evaluate the initial performance of the classifier. Additional tests are then done, changing the data slightly to see how it affect the classifier, trying to optimize the classifier to produce the best possible results.

2.1 Machine Learning Overview

Machine learning is the field of scientific study that uses induction algorithms and other algorithms which can be said to "learn" [4]. Machine learning is a subfield of artificial intelligence and cognitive science. Tasks performed in machine learning can typically be put in three different categories, supervised learning [5], unsupervised learning [6] and reinforcement learning [7]. Reinforcement learning will not be discussed in this report.

2.1.1 Supervised Learning

In a supervised learning algorithm, a model is created from labeled training data. Labeled in this case means that for each set of input parameters, the "right answer" is given, otherwise known as the output value. Labeled data is also known as the ground truth, what is known to be true. This learning process is called the training phase and the model created from the training phase can later be used to map new examples of data [5].

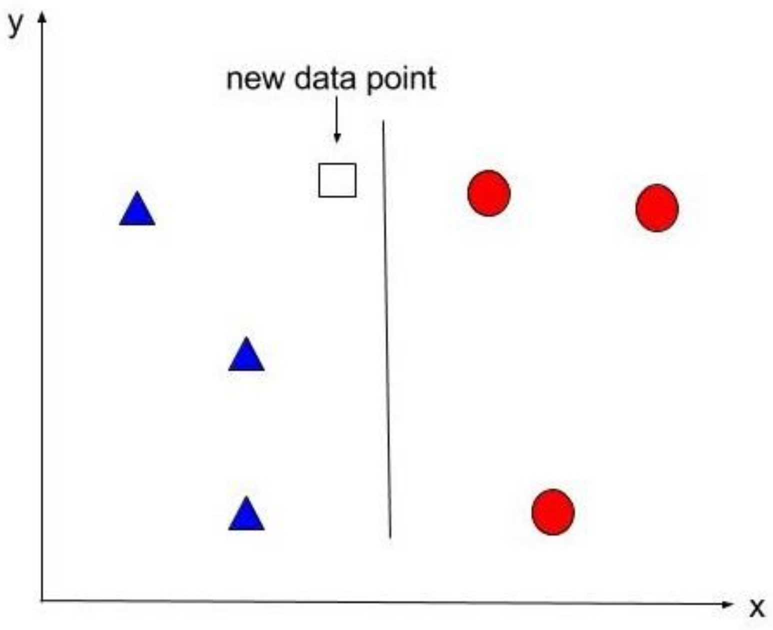


Figure 2.1: Machine Learning model

In **Error! Reference source not found.**, a simple model has been created using six data points of the two features denoted as *x* and *y*. In the middle of the figure a bar is separating the data points into two fields. The data points in the left field are known to be blue triangles while the data points in the right field are known to be red circles. After the training phase which created the model has been completed, a new data point is being processed. It is unknown whether this new data point is to be classified as a blue triangle or red circle. However, the model created by the supervised learning algorithm suggests that it is to be classified as a blue triangle.

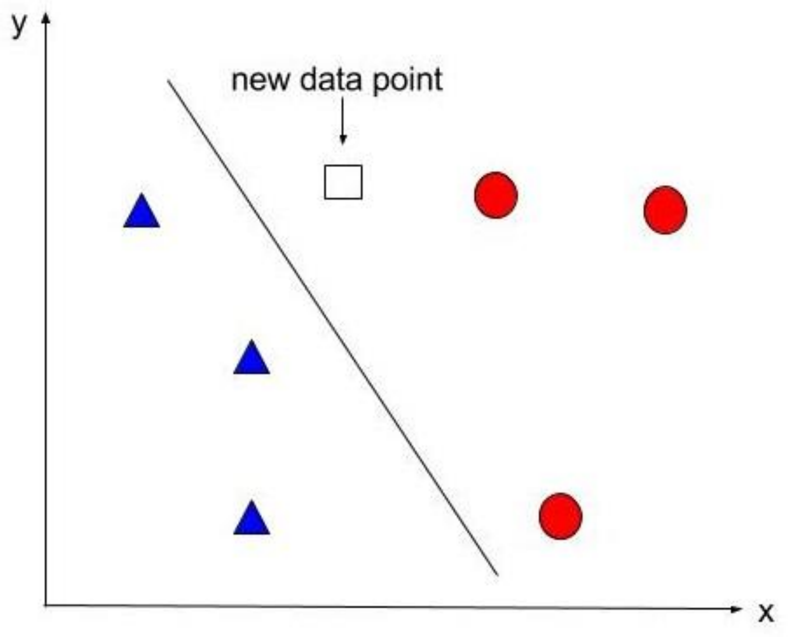


Figure 2.2: Another Machine Learning model

A different model might produce a different result however. In **Error! Reference source not found.**, a different supervised learning algorithm has been used on the same dataset as in **Error! Reference source not found.**. This new algorithm produces a different model. The six data points in the training set are placed just like they did in **Error! Reference source not found.**. The new supervised learning algorithm has produced a different model for separating the two classes. After the training phase, a new data point is presented to the model just like in the previous example. With this new model, the new data point will be classified as a red circle.

Note that changing algorithm is not the only way to produce a different model. Adding additional data points to a training set can produce a different and likely more accurate model.

After the training phase has been completed, a model has been constructed. The next phase is typically the testing phase. This is the phase where the effectiveness of the model will be tested. A new data set will be fed through the model. This data set is known as the testing data set [9]. The testing data set is unlabeled, however the output value of each data point is known to the researcher. It is impossible to tell which model presented in **Error! Reference source not found.** and **Error! Reference source not found.** is more accurate just by looking at the figures. However, with enough testing data, it is possible to evaluate them and choose the more accurate model.

2.1.2 Unsupervised Learning

In unsupervised learning algorithms, a model is created from unlabeled data. The algorithm is not told what the data is, it is simply given data and is expected to find structure within the provided data [6]. One approach to do this is called clustering.

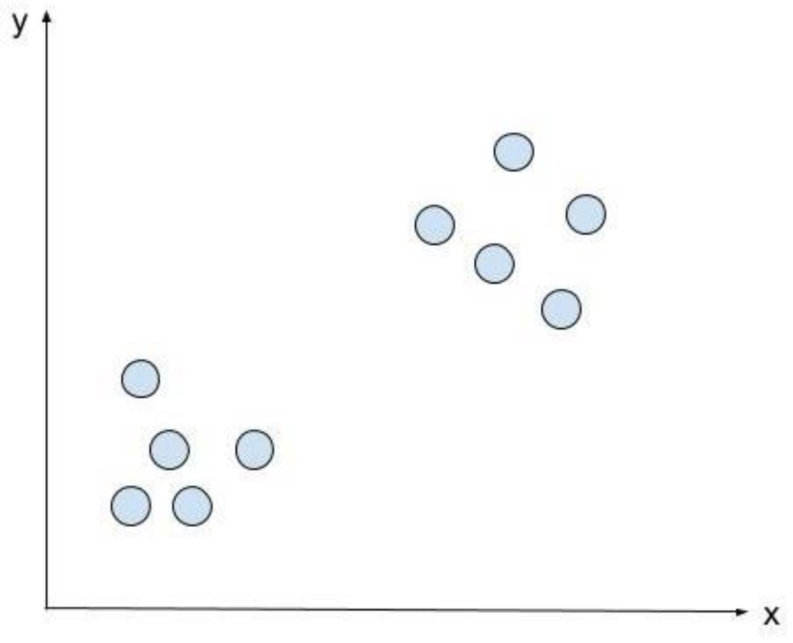


Figure 2.3: Unlabeled data points

From the data points shown in **Error! Reference source not found.**, a clustering algorithm might decide to separate the data points into two clusters, as seen in **Error! Reference source not found.**.

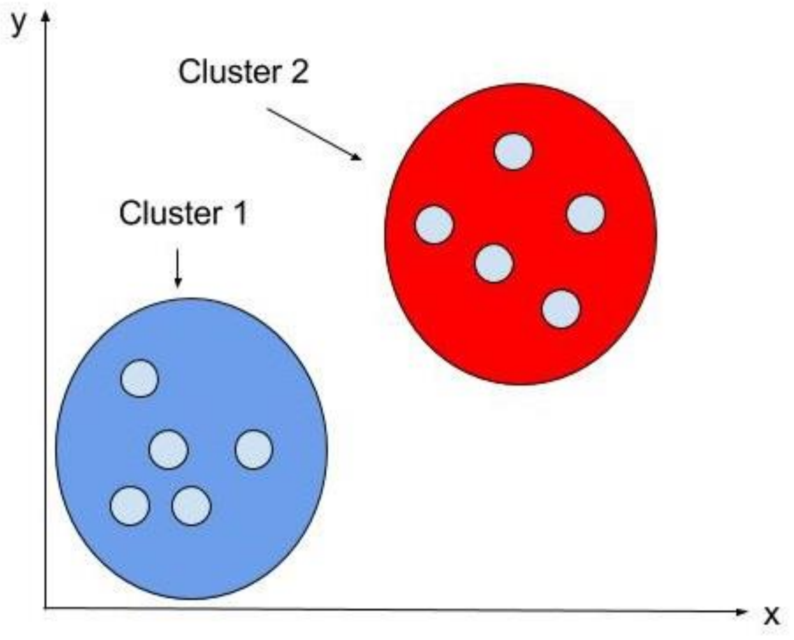
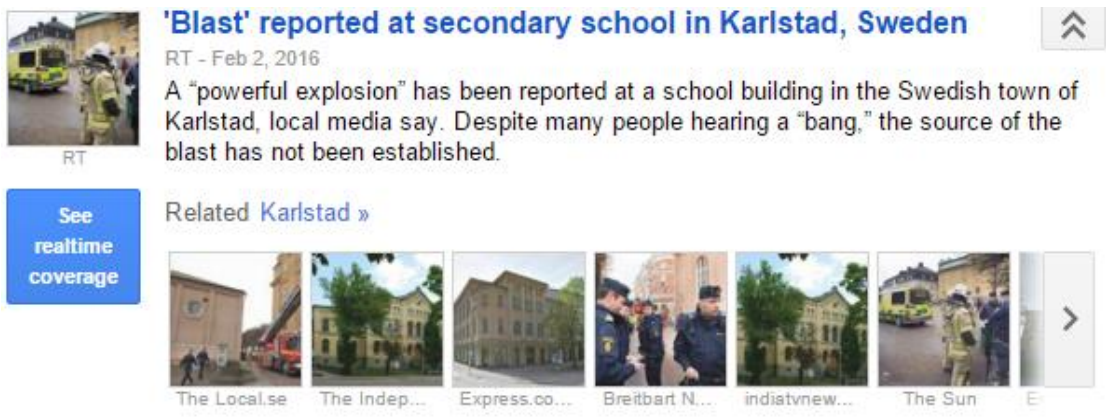


Figure 2.4: Clustered data points

Clustering is for example used in Google News [10] to cluster related news together [11]. In Google News, you can reach different sources reporting on the same event. In **Error! Reference source not found.**, different sources can be seen reporting on an explosion in Karlstad, Sweden.



'Blast' reported at secondary school in Karlstad, Sweden

RT - Feb 2, 2016

A "powerful explosion" has been reported at a school building in the Swedish town of Karlstad, local media say. Despite many people hearing a "bang," the source of the blast has not been established.

Figure 2.5: Google News news article

This is just one example of usage of an unsupervised learning algorithm, other usage of unsupervised learning algorithms has been done in genealogy, to organize computer clusters, social network analysis and many more applications [6] [11].

2.2 Features

In machine learning, a feature is an individual measurable property of the phenomena being observed [8]. In a packet switching network, statistical features can be used on traffic flows in order to differentiate between different activities. The first step is to identify as many features as possible. However, this can usually be a too large set to be managed and contain a lot of redundant information. In order to alleviate this, a feature selection process is imposed in the next step. The result is a subset of features which can be used effectively in machine learning applications.

2.3 Regression and Classification

There are two main ways of differentiating between different problems in machine learning when it comes to desired output.

- **Regression**

  The first type of problem is a regression problem. In a regression problem, the output is continuous [8], as can be seen in Figure 2.6.

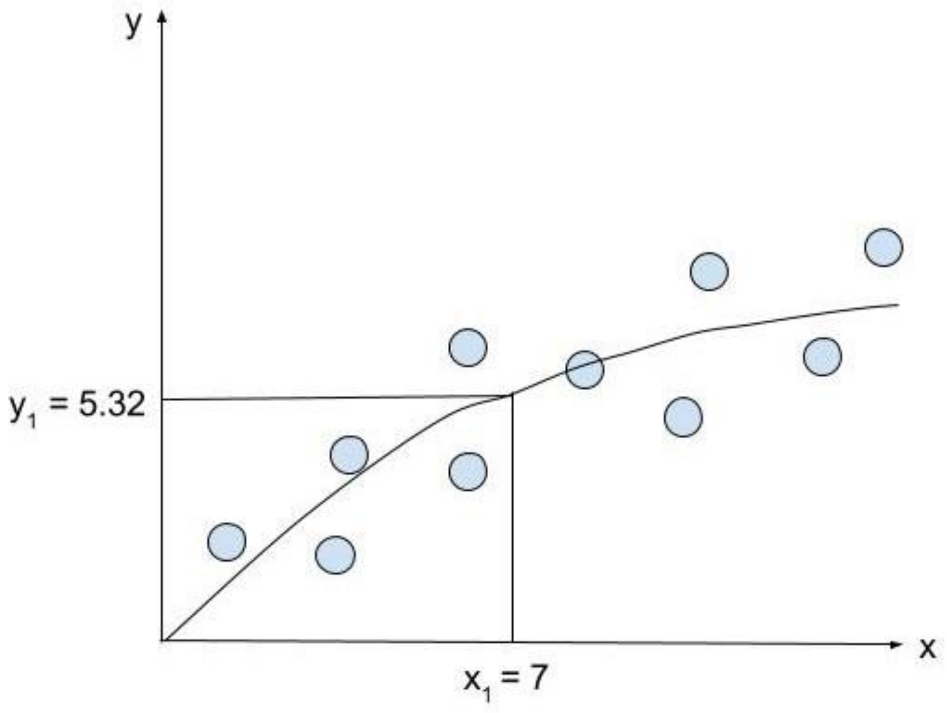


Figure 2.6: Machine Learning regression model

Figure 2.6 is a simplified figure where *x* is the value of a feature and *y* is the output value. The (x_1, y_1) pair shows how the model is used to predict the output value. In this case, our model predicts the output value *5.32* for an input value of *7*.

- **Classification**

  The second type of problem is a classification problem. In a classification problem, the output value is discrete. It can consist of two or more classes [8]. In Figure 2.7, a classification problem is presented where x denotes the value of a feature and y is the output value, also known as the class.

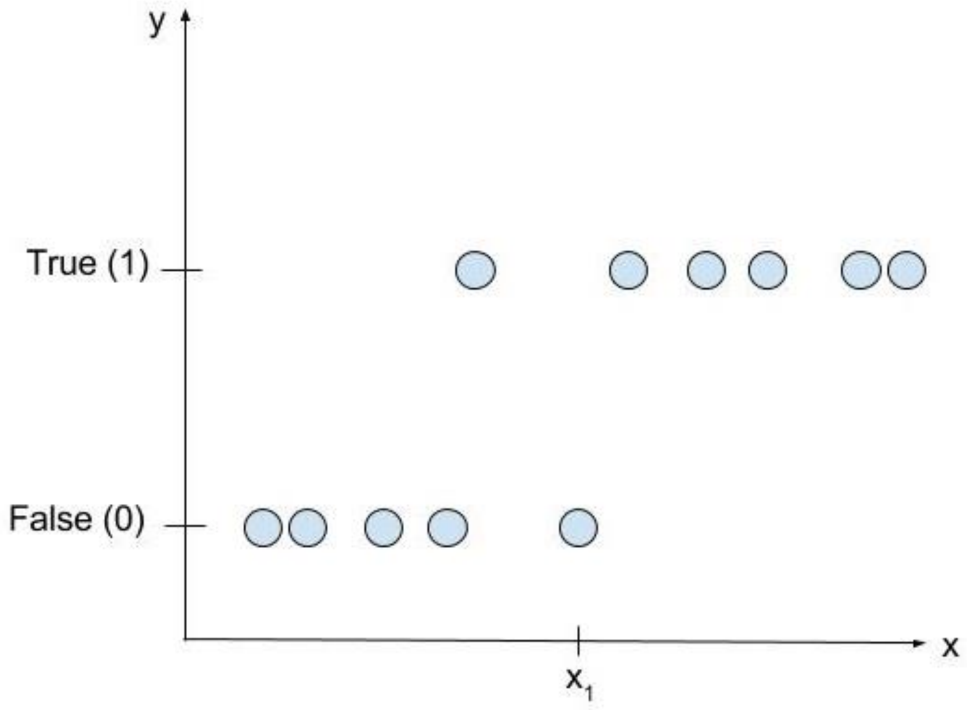


Figure 2.7: Machine Learning classification model

The two output classes available in Figure 2.7 are *True* and *False*. As can be seen, this prints a very different picture when compared to Figure 2.6. In this example, an *x* value must produce an output of *True* or *False*. If a value of x_1 is used as input, what would the output be? Figure 2.7 suggests that a smaller value of *x* would produce an output of *False* and a bigger *x* would produce an output of *True*. In this case, it is very hard to predict the value of x_1. In the regression problem, *7* produced *5.32*. If the correct result would have been *5.40*, *5.32* might still be considered a good result, depending on the sensitivity of the application. In the classification problem however, a wrong result is a wrong result, there is no “close enough”.

2.4 The Underfitting/Overfitting Problem

Underfitting and overfitting are problems that sometimes occurs in machine learning algorithms when trying to create an accurate model from training data [12]. To better illustrate the differences between underfitting and overfitting as well as what a good fit could look like, three different models have been created from the same data set.

- **Underfitting**

  A model which suffers from underfitting, generalizes the data too much. The model is in other words, too simplified.

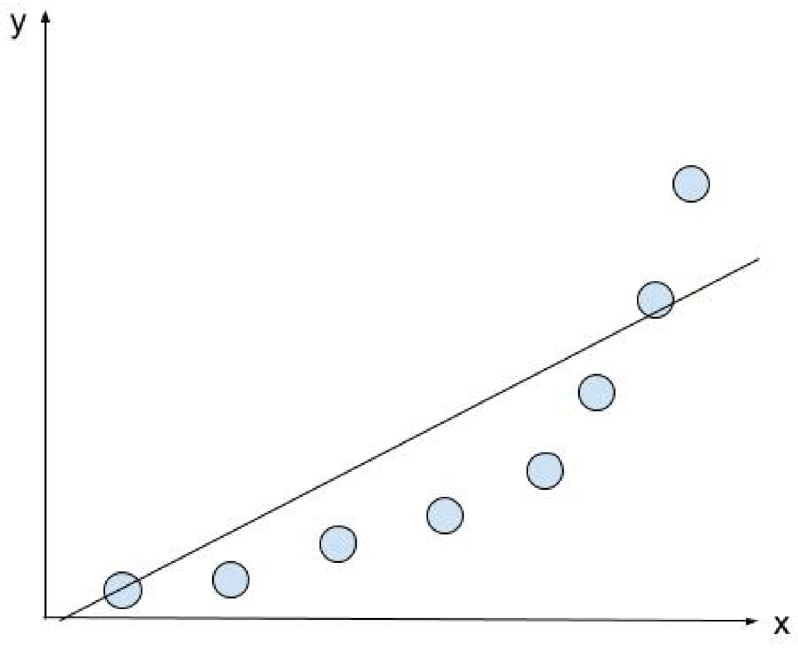


Figure 2.8: An underfitted model

As can be seen in Figure 2.8, the underfitted model is not likely to give you a very accurate output value for any new data point just like it gives a fairly poor output value for the training set.

- **Overfitting**

  A model which suffers from overfitting does the opposite of what an underfitted model does, it does not generalize enough. Overfitting is usually a more common problem in machine learning than underfitting is. Overfitting usually occurs when the model is too complex. One reason a model may have become too complex is when too many features are used relative to the number of data points. The problem with overfitting is that while it performs exceptionally well on the training data set, it does not perform equally well on the validation data set [12].

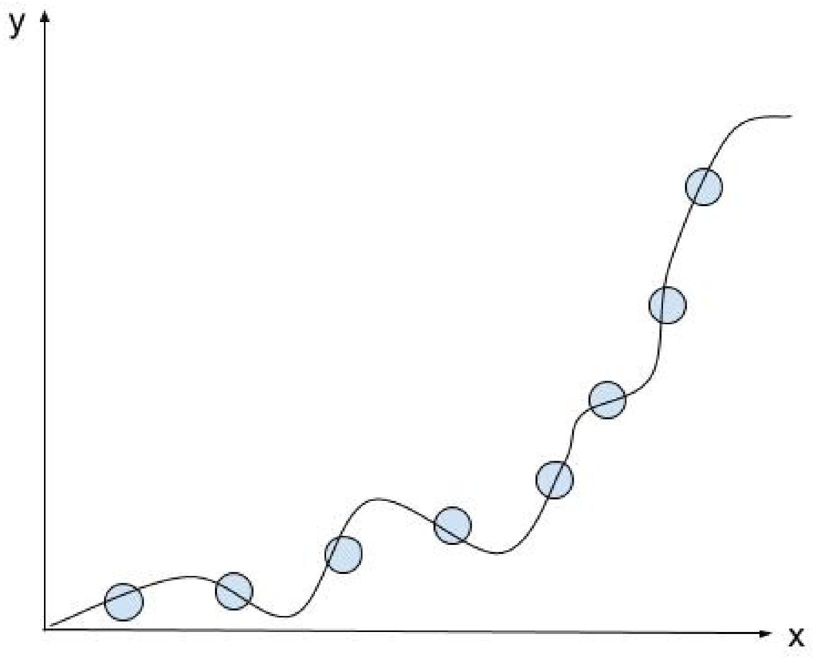


Figure 2.9: An overfitted model

As can be seen in Figure 2.9, the overfitted model performs very well on the training data set especially when compared to the underfitted model. However, due to the model´s data-specific nature, it may not give a good result for new data examples.

- **Good fit**
  A good fit is when the model does not suffer from either of these problems. It has a good balance where individual data points do not have the same importance as they do in an overfitted model but not of as little importance as they do in an underfitted model. The well fitted model captures the general trend of the data [12].

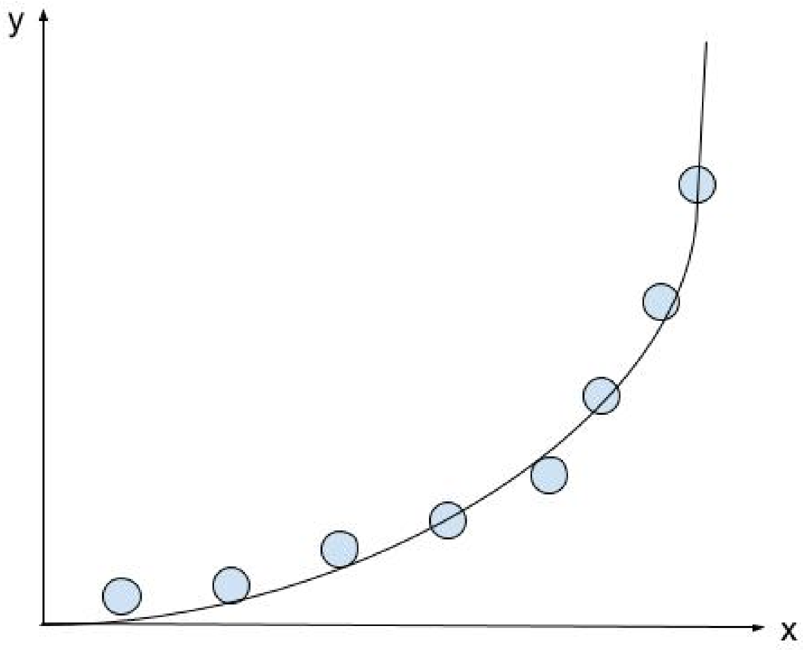


Figure 2.10: A well fitted model

Figure 2.10 illustrates what a well fitted model can look like. A well fitted model is a good model for evaluating new examples.

2.5 Approaches

In machine learning, algorithms can be put in one of many different approaches. These approaches group algorithms which are based on a similar strategy for machine learning. In this section, a few of these approaches are discussed.

- **Bayesian Approach**

  This approach encompasses machine learning algorithms which base their classification on Bayes' theorem. Bayes' theorem describes the probability of an event, based off conditions which may or may not be related to the event. Bayes' theorem can be mathematically stated as:

$$p(A|B) = \frac{p(A)p(B|A)}{p(B)} \quad (1)$$

  where *p* stands for probability and *A* and *B* denotes two different events [13]. Additionally, *p(A|B)* can be read as the probability that *A* is true given that *B* is true.

To exemplify, Bayes’ theorem can be used to figure out the probability of John going out for a walk when it is raining. Suppose that the probability of John taking a walk is 80 percent. Suppose that the probability of rain is 20 percent. If the probability of rain while John is taking a walk is 10 percent, then calculation of the probability of John taking a walk while it is raining by using Bayes’ theorem are done. The probability of John taking a walk, given that it is raining, is 0.8*0.1/0.2 = 0.4 = 40 percent.

To make an example more suited to a machine learning problem, suppose that *A* is a class and *B* is a set of features, *p(A|B)* gives the probability that these feature values is to be classified as class *A*. The goal of a classifier would then be to find the class which gives the maximum probability.

- **Decision Tree Learning**

In the decision tree learning approach, classification can be done based on values of features within the feature vector. A tree is constructed where each node can split into several paths based on a value of a feature. The decision tree algorithms use a “divide and conquer” strategy where the data is split into subsets of data. If a subset is pure, as in only containing one class, there is no need to split that subset further. However, if the subset is not pure, that subset is split further into smaller subsets. After the training has been completed, new data can be introduced and based on the constructed tree, classification of the new data can be performed [14].

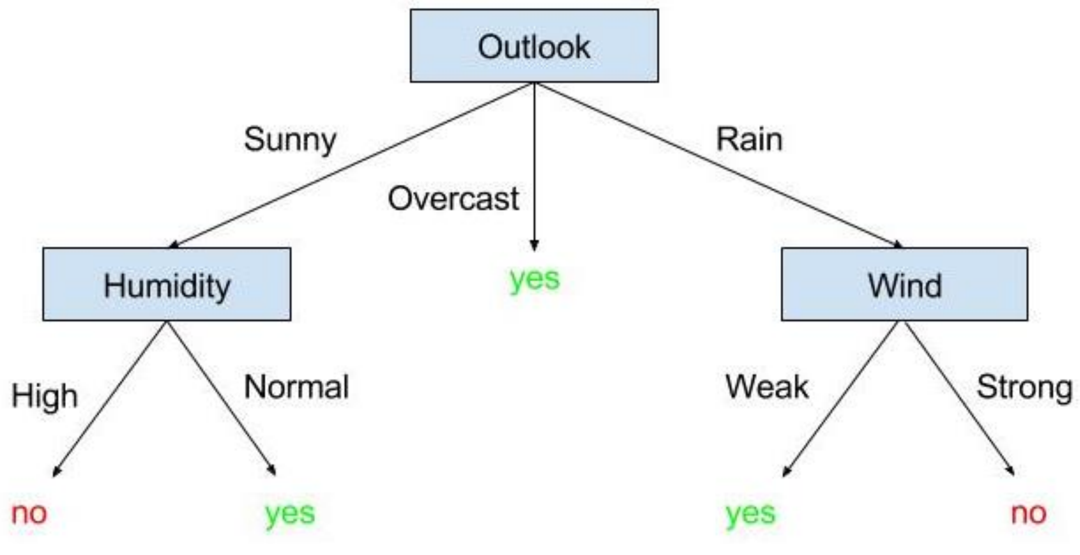


Figure 2.11: Decision tree model

In Figure 2.11, a decision tree model has been created. This figure is inspired by [15].

The model is used to determine whether a tennis player will play tennis on a specific day based on weather conditions of that day. We start at the root node, if it is raining outside, we follow the arrow pointing to the right subtree. Since rain alone will not determine if the tennis player will play or not, another attribute is used to further split the data. If the winds are weak, we continue to the left subtree where we find a *yes*, meaning that on a rainy day with weak winds, the tennis player will play.

Advantages of decision trees is that the models are very easy to understand and handle large datasets very well. One disadvantage of decision tree based algorithms is that they are susceptible to overfitting when the size of the decision tree grows large with relative small amount of training data [16].

- **Support Vector Machines**

  Support vector machines are supervised learning models which defines decision planes to separate objects of different class membership. By defining boundaries, objects of different classes are separated into two or more groups. Examples of this can be seen in **Error! Reference source not found.** and **Error! Reference source not found.** from earlier. The object of the support vector machine is to create these boundaries in order to make the gap as wide as possible [17]. New examples presented to the model will be classified based on which side of the boundary they end up on.

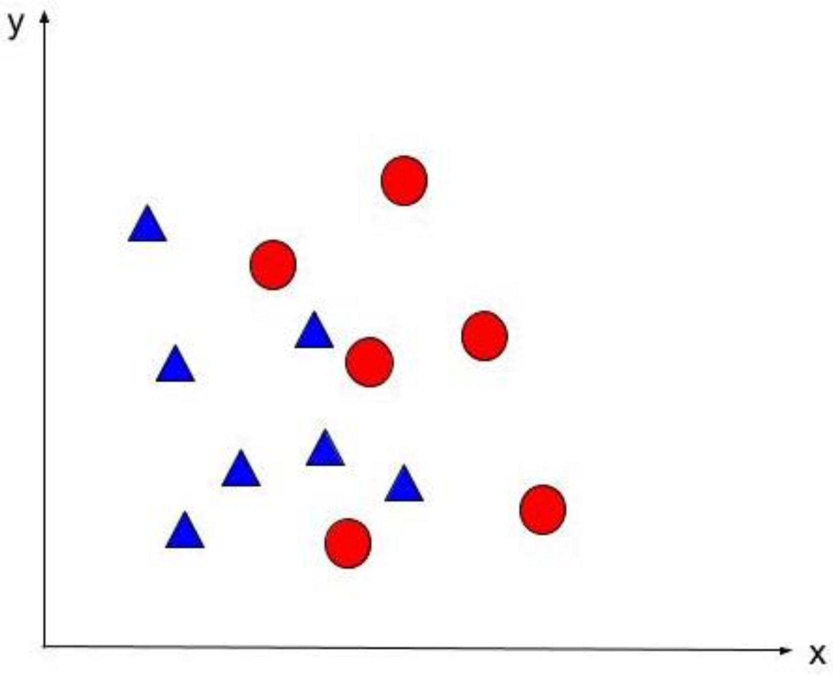


Figure 2.12: Data points which are hard to separate

Support vector machines are known as a hyperplane classifier, a classifier which draws separating lines through the data [17]. Some classification problems are more complex than just drawing a separating line through the data. Figure 2.12 illustrates such a problem. It would not be ideal to separate the two different classes presented in Figure 2.12 with a straight line.

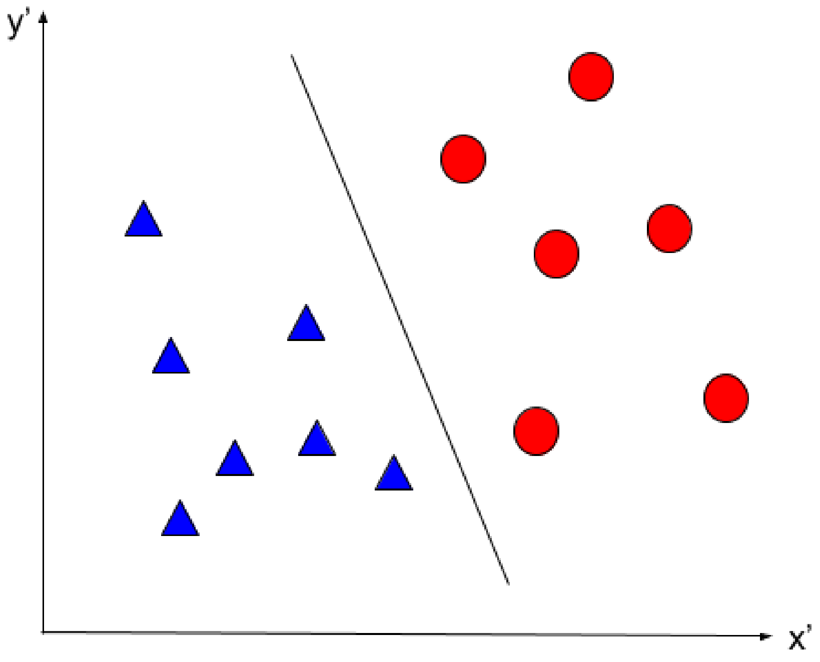


Figure 2.13: Data points after applying a kernel transformation

Support vector machines got a way of dealing with this problem however. Through the use of mathematical functions known as kernels, the objects can be rearranged in order to allow a straight line to properly separate the classes [17]. Figure 2.13 shows the resulting model after the objects in Figure 2.12 has been transformed. Figure 2.12 and Figure 2.13 was inspired by [18].

2.6 Random Forest

One flaw with decision tree classifiers, is their tendency to overfit the data. Random forest is an algorithm which tries to relieve this flaw. In order to do this random forests combines the bagging algorithm [19] with the idea of random selection of features. By constructing a multitude of decision trees and using the random selection, the variance of the trees can be controlled and a more generalized fit to the data than that of a single decision tree [20].

The algorithm used to produce the random forest can be described as follows:

1. For $b = 1$ to B where B is the number of trees:

    - Create a subset of the dataset of size N from the training data. Note: this subset may contain duplicates.
    - Grow a tree by recursively repeating the following steps for each terminal node of the tree until the minimum node size n_{min} is reached.
      1. Select m features at random.
      2. Pick the feature witch best splits the data.
      3. Split the node into two daughter nodes.

2. The output is the ensemble of trees $\{T_b\}_1^B$.

In a classification problem, the class predicted by the random forest algorithm is decided as the majority vote of each trees' prediction.

In a regression problem, the predicted output value is going to be the mean of all the trees' output values [21].

Random Forest Example

Let's make a random forest classifier using three decision trees. The purpose of the classifier is to predict whether a tennis player will play tennis based on the weather conditions of the day.

The features are the weather conditions and the output is the answer to the question “Will he play tennis?”.

Subset	**Outlook**	**Humidity**	**Wind**	**Play Tennis?**
1	Sunny	Normal	Weak	Yes
1	Rain	Normal	Weak	No
2	Rain	High	Weak	Yes
2	Rain	High	Strong	No
3	Overcast	Normal	Strong	Yes
3	Overcast	High	Strong	No

Table 2.1: Dataset with ground truth for random forest classifier

A very small data set has been constructed to be used in this example as can be seen in Table 2.1.

The next step is to create three random subsets of this data. In Table 2.1, a column has been added stating which subset each data point belongs to for this example. In this case, the subsets have been selected such that only one feature is needed to split the data.

Decision trees are now constructed based on each subset. Subset 1 can be split based on the Outlook feature, subset 2 on the Wind feature and subset 3 can be split on the Humidity feature. This creates the decision trees illustrated in Figure 2.14, Figure 2.15 and Figure 2.16.

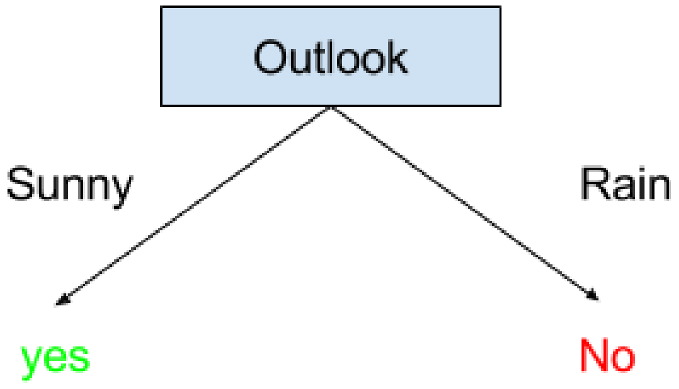


Figure 2.14: Tree 1 in random forest example

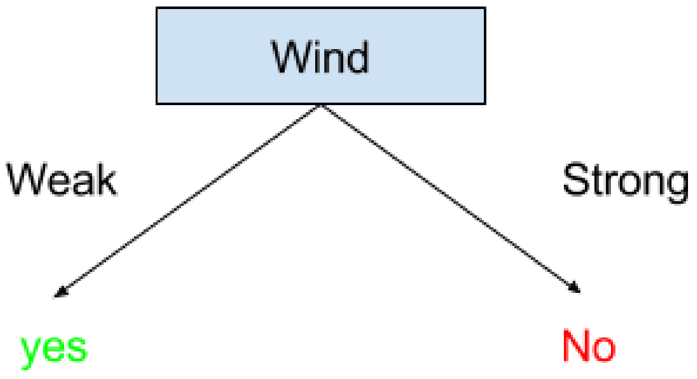


Figure 2.15: Tree 2 in random forest example

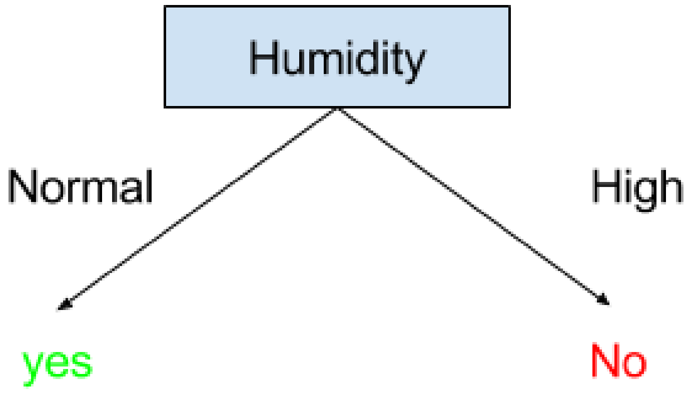


Figure 2.16: Tree 3 in random forest example

These three decision trees make up the random forest classifier. The classifier will use these trees in order to make predictions on new data. In Table 2.2 a dataset without ground truth has been constructed, prediction of the output of these data points can be done using the random forest classifier.

Outlook	Humidity	Wind	Play Tennis?
Sunny	Normal	Weak	?
Rain	Normal	Weak	?
Sunny	High	Strong	?

Table 2.2: Dataset to be used for prediction

The {Sunny, Normal, Weak} data point gets classified as “Yes” by all of the decision trees, which means the random forest classifier will classify this data point as “Yes”.

The {Rain, Normal, Weak} data point will get classified as “Yes” by the second and third decision tree, however the first decision tree will classify it as “No”. The class will be decided by majority vote and since there are more “Yes” than “No”, the random forest classifier will classify the data point as “Yes”.

Filling in the predicted outputs will construct Table 2.3.

Outlook	Humidity	Wind	Play Tennis?	
			Yes	
Sunny	Normal	Weak	Yes	Yes
			Yes	
			No	
Rain	Normal	Weak	Yes	Yes
			Yes	
			Yes	
Sunny	High	Strong	No	No
			No	

Table 2.3: Dataset with the predicted classes

2.7 Cross-validation

Cross-validation is a technique for assessing how the result of a statistical analysis will generalize to an independent dataset. It is mainly used in problems where the goal is to achieve prediction. Cross-validation is used to estimate the accuracy of the predicting model. In a prediction problem, a model is generally given a set of data with known output. The model will train on this data and the model can later be used to predict output values of new data where the output is unknown. In order to be able to estimate the accuracy of the model, part of the training data is used as validation data. The validation data should not be used for training [22].

Assume we have a labeled dataset with 100 data points. We want to use as much as possible of these data points to train the model. We also need some data points for validation. The problem is that whichever way we divide the data points, the size of the training set or the validation set will suffer. Cross-validation solves this problem.

Instead of separating the data points into two subsets, the data set is divided into *k* folds as illustrated in Figure 2.17.

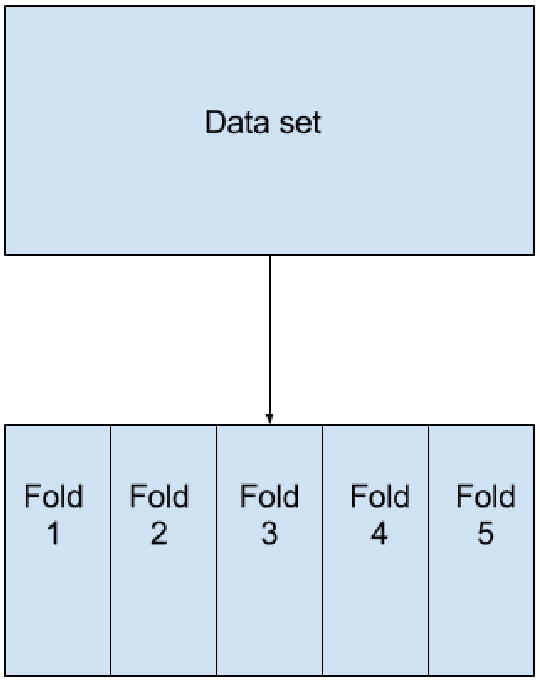


Figure 2.17: Dataset gets divided into 5 folds

In each round of cross-validation, one of the folds will be used as the validation data set while the rest of the folds will be used as training set. The cross-validation will continue for *k* rounds, having every fold be the validation set once. This way, most of the data will be used for training every round while at the same time all the data will be used for validation once. Figure 2.18 shows two rounds of cross-validation.

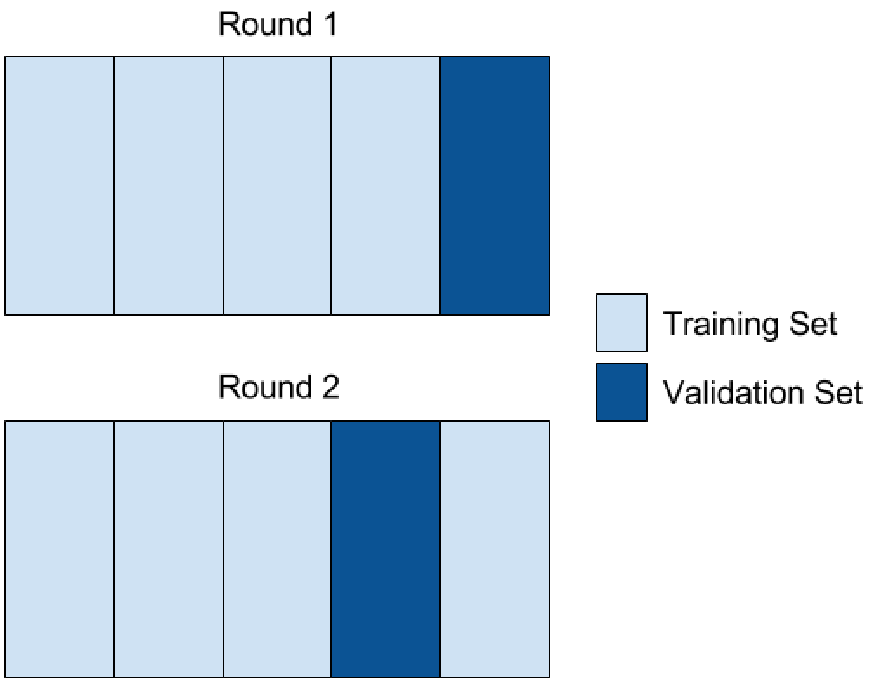


Figure 2.18: Two rounds of cross-validation

After all of the cross-validation rounds are completed, the accuracy of the model is calculated as the average accuracy of all rounds.

A common mistake when performing a cross-validation occurs when trying to select the best possible features. Choosing the best features based on all the data and then performing cross-validation is incorrect use of cross-validation. Instead the best features should be picked from the data that is not used for prediction before the cross-validation. If the cross-validation is used incorrectly, the accuracy of prediction will be inflated compared to the actual accuracy of the model [23].

Stratified k-fold cross-validation

In a stratified k-fold cross validation, the folds are selected in such a way that the proportions are approximately the same for every fold [22]. This means that if 80 % of a dataset is classified as “video”, then each fold will contain about 80 % “video” as well.

2.8 Tools

In this section, some of the tools which were used during this study will be presented.

2.8.1 Python

Python [24] is the programming language which was used to conduct the experiments in this report. Python is a widely used programming language, being one of the most popular programming languages today. What makes Python a good choice for this study, is that it has importable modules for machine learning and for dealing with large amounts of data.

2.8.2 IPython Notebook

The IPython Notebook [25] is the computational environment in which Python code is entered. It contains input and output cells which can contain code, text, plots etc. As such it is a very powerful tool for structuring code and for presenting results in a meaningful way. This is especially useful when dealing with large amounts of data where understanding the data is paramount.

2.8.3 Python Modules

In this study, advantage was taken of Python modules to handle the data, present the data, as well as taking advantage of machine learning classifiers which are already implemented. Below, some of the most important modules in this study are listed and a short description of what they are.

- **Numpy**
  [26] contains the Numpy array data structure which supports large multi-dimensional arrays and matrices. It also gives a lot of tools for providing operations on these arrays, making it a very powerful tool for manipulation of the large amounts of data used in this study.

- **Pandas**
  Pandas [27] is an open-source Python module for data manipulation and analysis. It is

supposed to provide easy to use and high-performance data structures and tools for that purpose. From the Pandas module, it was the *Dataframe* data structure and its associated functions that were primarily used in this study.

- **Scikit-learn**
  Scikit-learn [28] is a tool for data mining and data analysis. It includes the machine learning classifiers which will be used in this study.

2.8.4 SNIC

The Swedish National Infrastructure for Computing (SNIC) provides resources and support for large scale computations as well as data storage for researches working in Swedish universities, research institutes, etc. [29]. Having access to resources provided by SNIC allows us to handle large amount of data which cannot be handled locally, a necessity for this study.

2.9 Evaluation Metrics

When evaluating results of a machine learning classifier, some classification performance metrics will be presented. In order to understand these metrics and to be able to evaluate the results of the machine learning classifier, a look is taken on how accuracy and precision as well as other metrics are defined. When evaluating classification, each classified data point can be put in one of four categories. These four categories are known as *true positives, true negatives, false positives* and *false negatives* [30]. In this example, our data points will be Internet traffic flows and the machine learning classifier will either classify a data point as video or non-video.

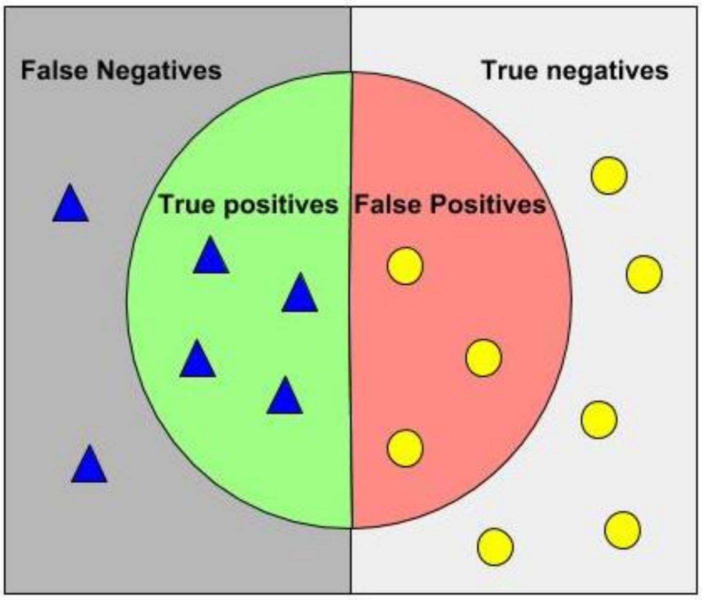


Figure 2.19: Example results of a classification process

In Figure 2.19, blue triangles represent actual video flows, while yellow circles represent non-video flows. The big green/red circle represent the flows which our machine learning classifier has classified as video, while the flows outside the circle has been rejected. This figure was inspired by [31]. The categories which evaluated data points can fall into are:

- **True Positive**

  A true positive can also be called a hit [30]. In our example, a true positive would be when our machine learning classifier accurately classifies a video flow as video. In Figure 2.19, the true positives are contained within the green half circle.

- **True Negative**

  A true negative can also be called a correct rejection [30]. In our example, a true negative would be when our machine learning classifier accurately rejects a non-video flow. In Figure 2.19, the true negatives are contained within the light grey area.

The true positives and the true negatives make up the data points which our machine learning classifier has accurately classified as video or accurately rejected.

- **False Positive**

  A false positive can also be called a false alarm [30]. In our example, a false positive would be when our machine learning classifier has incorrectly classified a non-video flow as video. In Figure 2.19, the false positives are contained within the red half circle.

- **False Negative**

  A false negative can also be called a miss [30]. In our example, a false negative would be when our machine learning classifier has incorrectly rejected a video flow. In Figure 2.19, the false negatives are contained within the dark grey area.

The false positives and false negatives make up the data points which our machine learning classifier has incorrectly classified as video or incorrectly rejected.

Based on the number of occurrences in these different categories, different metrics can show the effectiveness of the machine learning classifier. A closer look is taken at some of these metrics.

- **Accuracy**

  Accuracy is perhaps the easiest metric to understand as it is a very commonly used. Accuracy is defined as:

  $$accuracy = \frac{tp+tn}{tp+tn+fp+fn} \quad (2)$$

  where *tp* is the amount of true positives, *tn* is the amount of true negatives, *fp* is the amount of false positives and *fn* is the amount of false negatives [30]. An accuracy of 75 percent means that three out of four examples was correctly classified or correctly rejected. For the example shown in Figure 2.19, the accuracy would be 9/14 ≈ 64 %.

- **Precision**

  Another metric for evaluating the effectiveness of a machine learning classifier is precision. Precision is defined as:

$$precision = \frac{tp}{tp+fp} \quad (3)$$

where *tp* is the amount of true positives and *fp* is the amount of false positives [30]. A precision of 90 percent means that out of ten classified data points, nine data points are relevant. In the example shown in Figure 2.19, the precision would be 4/7 ≈ 57 %.

One of the problems with the precision metric is that it is calculated using false positives without taking true negatives into account. This means that in Internet traffic with higher amount of flows not being video, the precision metric for video will go down. Because of this problem, the precision metric shall not be given too much credit during evaluation.

- **Recall**

  Recall, also commonly referred to as true positive rate, is a metric often presented together with precision. Recall is defined as:

$$recall = \frac{tp}{tp+fn} \quad (4)$$

where *tp* is the amount of true positives and *fn* is the amount of false negatives [30]. Recall is a measure of how many of the relevant data points get classified. A recall of 60 percent means that of ten relevant data points, six get correctly classified. In the example shown in Figure 2.19, the recall would be 4/6 ≈ 67 %.

- **F-measure**

  As previously mentioned, precision and recall are often presented together. F-measure is a measure that combines the precision and recall into one measurement. When equal weights are given to precision and recall, the F-measure are known as F_1 measure. The F_1 measure is defined as: [30]

$$f_1 = 2 * \frac{precision * recall}{precision + recall} \quad (5)$$

The F_1 measure is a measure of accuracy of the machine learning classifier. It can be interpreted as a weighted average of the precision and recall. A F_1 score of 1 would mean that all relevant data points got correctly classified and all irrelevant data points got rejected. In the example shown in Figure 2.19, the F_1 score would be 8/13 ≈ 62 %.

Note that the calculated *F_1* score (62 %) is different from the accuracy score previously calculated (64 %). A reason why accuracy is higher in this case, is that the *F_1* score does not take true negatives into account.

Since the F1 score is calculated using precision and the precision suffer from the issues mentioned above, the F1 score will not be given much credit during evaluation.

- **Confusion Matrix**

  A confusion matrix is a table layout which allows visualization of the performance of an algorithm. It is commonly used in conjunction with supervised learning algorithms. In a classification problem, one axis denotes the predicted classes of a classifier, while the other denotes the actual class [30]. An example of a confusion matrix can be seen in Table 2.4, where a classifier has tried to predict pets based on some unknown features. In this example, 18 actual cats were used for classification, where 15 of them were correctly predicted to be cats. Additionally, 14 actual dogs were used, where 12 of them were correctly identified as dogs. The correctly predicted cases goes in a diagonal from the top left box to the bottom right box.

	Predicted Cat	**Predicted Dog**
Actual Cat	15	3
Actual Dog	2	12

Table 2.4: Example of a confusion matrix

- **Receiver operating characteristic (ROC) curve**

  A ROC curve is a graphical plot which can be used to evaluate the performance of a binary classifier at different thresholds. The curve is created from plotting the true positive rate against the false positive rate [30]. You want to maximize true positive rate while simultaneously minimize false positive rate. This means that the perfect score would be a true positive rate of *1* and a false positive rate of *0*. Hence, the closer the curve is to the top left corner, the better the classifier. One common way to assess how good the curve is, is to calculate the Area Under the Curve (AUC). The perfect curve would produce an area under the curve score of *1*. An example of a ROC curve is illustrated in Figure 2.20. In this figure, curves for perfect performance and random guessing are also shown to better understand how the performance of the ROC curve is evaluated.

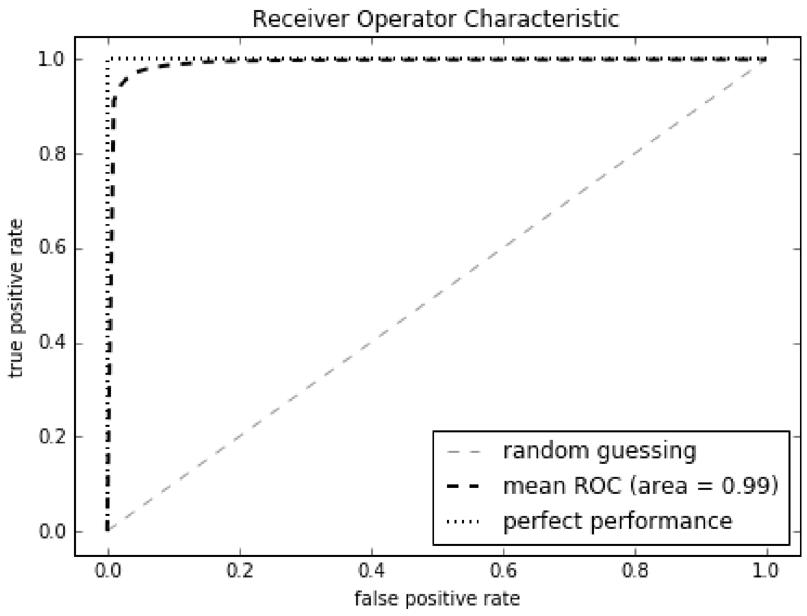


Figure 2.20: ROC curve of a classifier

A classifier normally outputs the predicted class, but some also allow to output the probability of a predicted class. Probability is what is used for different thresholds. When deciding which threshold that is the best to use, decisions has to be made whether you want to maximize true positive rate or minimizing false positive rate. Whichever is better depends on the underlying problem.

Which metric that should be paid more attention to depends on where and how the classifier is used. If it is important that all flows that get classified as video, are actual video flows, then more attention need to be paid to precision than any other metrics.

2.10 Chapter Summary

The purpose of this chapter was to give the reader an introduction to Machine Learning which will aid in fully understanding this report. Additionally, an introduction has been made to the metrics used in evaluating machine learning classifiers as well as the tools which has been used to complete these experiments.

3 Previous Studies

In this section, a look is taken at some of the research which has previously been done on the subject.

3.1 Classification Using Bayesian Techniques, Moore 2005

In the paper written by Moore and Zuev [32], good results were achieved using Naïve Bayes classifiers. In this experiment a large amount of features was used, namely *248*. Some of the features were: TCP port, flow duration, statistical metrics of inter arrival time and payload sizes. Some of the classes that were used for classification were P2P, mail, games and multimedia. The classifiers which were used were Naïve Bayes and a variant of Naïve Bayes called Naïve Bayes Kernel Estimation. Furthermore, a feature selection process was performed, were the features got ranked and the optimal amount of features to use was discovered through experimentation. The dataset was collected from the same location during two 24 hour periods, one year apart.

The results showed a mediocre result for the Naïve Bayes classifier, achieving a 65.26% accuracy. Using the Naïve Bayes Kernel Estimation with a preceding feature selection process achieved a much better result at 96.29% accuracy.

More tests were done to see how these models created from old data would perform on newer data taken one year later. As could probably be expected, all classifiers performed worse on the new data. Most notably the Naïve Bayes classifier dropped to an abysmal 20.75% accuracy. In contrast, the Naïve Bayes Kernel Estimation classifier with preceding feature selection managed to achieve an accuracy of 93.73%. Simply by using the feature selection process on the Naïve Bayes classifier, its accuracy improves to 93.38%.

What can be taken away from this research, is the importance of feature selection as it seemed to be the most important factor in producing accurate classifiers in this experiment. The experiments also give an idea of how the classifiers accuracy will drop over time. The drop from 96.29% to 93.73% may not seem significant, but it may make a huge difference especially when the number of flows are huge.

3.2 Classification Using Five Algorithms, Williams 2006

In the paper written by Williams et al. [33], experiments to compare five different machine learning algorithms were conducted. The features which were used for this experiment consisted of a total of 22 features including bytes, packet size and inter-arrival time in both directions. Along the full set of features, two subsets were also used to see how using a smaller amount of features impacted the performance. The classes which were used for classification were FTP-data, HTTP, SMTP, DNS, Telnet, and interestingly Half-Life. The dataset was gathered from a public network dump, using four 24-hour traces taken in the early 2000´s. The algorithms which were used had few parameters to tune and were easy to implement. The chosen algorithms were: Bayesian Network, C4.5 Decision Tree, Naïve Bayes, Naïve Bayes Tree where Naïve Bayes were used with two different ways of modelling features.

The result of the experiment show that most of the algorithms performed well with accuracies greater than 95%. An outlier was the Naïve Bayes kernel estimation classifier which only managed to achieve about 80% accuracy. The experiment also showed that using a subset of features did not affect too harshly, the largest change was a loss of 2.5 units of percentage.

From these experiments, conclusions can be made that different classifiers can achieve similar results and feature selection can be effective at reducing the amount of features without a big loss in accuracy.

3.3 Classification Using Combiners, Dainotti 2011

In the paper written by Dainotti et al. [34], an experiment is carried out to determine the effectiveness of combiners in early classification of Internet traffic flows. Combiners use different methods to combine the predictions from multiple classifiers in order to get a better prediction. The ten initial packets of each flow were used in the experiment, going through all flow lengths, from one to ten. Features were selected differently for different machine learning classifiers but included *payload size, inter-packet time*, *protocol, bitflow duration* as well as statistical features of *payload size* and *inter-packet time*. Twelve classes were used for classification, including *http*, *bittorrent* and *skype*. In this study, there were no classes for any kind of video traffic.

There were six kind of machine learning classifiers, which included representatives from the

Bayes [13], *Decision trees* [14], *Lazy learning* [35], *Rule* [36] and *Artificial neural network* [37] approaches. Additionally, a payload inspecting classifier as well as a port based classifier were used. After the classification process had been completed for the individual classifiers, combiners were used on eight different combinations of classifiers. Each combination always contained the three classifiers with the highest accuracy. The combiners also used six different methods for classifying.

The results for the individual classifiers showed promising results for most classifiers. The exceptions were the Naïve Bayes [38] and the port based classifier, netting an accuracy of 43.7 and 15.6 percent respectively. The top performer was the classifier based on the C4.5 algorithm [39], which managed to get an accuracy of 97.2 percent.

The results for the combiners showed that it can be an effective way of increasing accuracy if chosen correctly. Most of the combiners performed worse than the individual top performer. The two exceptions were the BKS [40] and Wernecke [41] combiners, which both managed to score a higher accuracy than the individual performer on eight out of nine combinations. For these eight combinations, the two combiners performed identically, netting a top accuracy of 98.4 percent.

Lastly, the result for different amount of analyzed packets, showed that combiners are even more useful as the number of analyzed packets goes down. The BKS and Wernecke combiners managed to reach their top accuracy already after four packets. Meanwhile it took the classifier based on the C4.5 algorithm seven packets to reach its top accuracy. Generally, increasing the amount of packets did not negatively affect the results for any classifier or combiner.

In this study there was no video traffic being considered. However, it showed the potential in use of combiners to improve results of individual classifiers as well as overall high accuracy.

3.4 Classification Using SVMs, Tabatabaei 2012

In the paper written by Tabatabaei et al [42], an experiment is carried out to determine how effectively Internet traffic flows can be classified using early detection. In order to provide early classification, every odd number of packets between 3 and 15 were used in the experiment. They also completed the experiment for all the packets in the flow for comparison. In the experiment, 40 features were selected where some of the most discriminating ones was packet length, inter arrival time, active and idle time in both directions. Seven classes for classification were used, including *BitTorrent*, *Skype* and *P2P live-streaming*.

The machine learning classifiers which were used for the experiment were two types of SVMs [17], SVM fuzzy pairwise and SVM fuzzy one vs all. Additionally, the k-Nearest Neighbors algorithm [43] were also used in the experiment. The results of the experiment showed that seven packets provided the best accuracy of classifying the Internet traffic flows. Performing classification on seven packets were even better than performing the classification on the entire flow. This would suggest that early classification is not necessarily a disadvantage compared to doing it on an entire flow. The classification was most accurate using the SVM fuzzy one vs all algorithm, reaching a precision of 84.9 percent. P2P livestreaming was successfully identified 98.28 percent of the time. However, sometimes other traffic was wrongly identified as P2P live-streaming, most notably web browsing which were wrongly identified as P2P livestreaming 12.28 percent of the time.

3.5 Video Classification, Nossenson 2015

In the paper written by Nossenson and Polacheck [44], an experiment was carried out to try and classify different types of video traffic. The two types of video traffic were Video on Demand (VOD) and live video flows. Two different experiments were conducted, one using single flow classification and the other using multi flow classification. The multi flow classification experiment will not be discussed here. In their experiment, they did not make use of machine learning to create their classifier models. They used two different classifiers which used only one feature each for classification. The first classifier looked at the average packet size in the up direction and the second classifier looked at the average packet size in the down direction. The entire flow was used to calculate the average packet size. The dataset which were used, was collected specifically for this research project by a large Israeli Internet service provider. The dataset only contain video, which are labeled either as VOD or live video.

The first classifier achieves a result of 74% accuracy. This is a worse result than a trivial classifier which would predict VOD for all flows. The trivial classifier would achieve an accuracy of 87%.

The second classifier achieves a result of 96% accuracy. This shows that the average packet size in the down direction is a more effective feature than the average packet size in the up direction.

There is not much in this project that can be used within this study. Even though they both handle the problem of video classification, they are too dissimilar when it comes to the problems they are trying to solve.

3.6 Early Classification Using Payload Features, Peng 2016

In the paper written by Peng et al. [45], very promising results for early classification of Internet traffic packets was presented. The aim of their study was to evaluate the effectiveness of statistical features when compared to the individual packet sizes. They chose to use statistics from the six initial data packets, meaning packets which has a payload. The features which were used in the experiment was ps_1, ps_2, ps_3, ps_4, ps_5, ps_6 and the statistical features *average*, *standard deviation*, *maximum*, *minimum*, *geometric mean* and *variance*. Note: ps_x is an abbreviation for payload size of the *x*:th packet. The features were bundled into six different feature vectors so that evaluation of the different features could be done. The data from which to extract these features came from three different data sets. These data sets were collected at three different points of time, namely in 2000, 2009 and 2013. There were several classes for classification of data, including *http, ftp, DNS* and *streaming media*.

Ten well-known machine learning classifiers were used, covering five different approaches. The approaches covered in the experiment were *Bayes, Boosting, Rule, Trees* and *Lazy learning*. The result of the experiment show accuracies of above 95 percent for most machine learning classifier and feature vector combinations. Two of the machine learning classifiers performed relatively bad on all feature vectors, namely Naive Bayes and adaptive Boost M1. The best performing feature vector contained the packet sizes of all six packets, as well as the average size and the variance of the packet sizes. While the results of the experiment seem promising, there is no way of extracting the accuracy of correctly classifying streaming media.

3.7 Chapter Summary

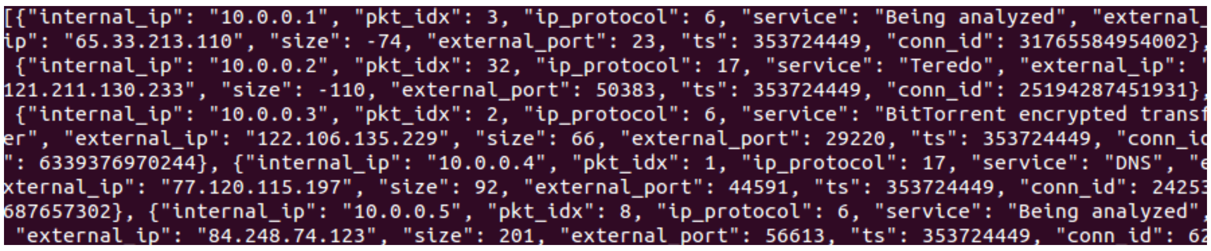
In this chapter, a summary of some related projects were presented. From these projects, some example results can be extracted. The projects also show which features has been used for classification of flows, which is also something that has been taken into account for this study. All things considered however, these related projects are too different to make any real conclusions related to this study.

4 Study Preliminaries

This chapter will describe the process from receiving the dataset until constructing a finished machine learning classifier. The chapter will start by looking at the dataset which was provided by Procera, and which data is filtered out of the dataset. Additionally, a closer look is taken at the features which was used in this study and the process of creating the feature vector is described. Furthermore, a description of the process of training the machine learning classifier and evaluate the results are also supplied.

4.1 The Dataset

The dataset is constructed by Procera using information gathered from one of their Deep Packet Inspection (DPI) devices. The dataset is sent in the form of a JSON file. The beginning of the JSON file is shown in Figure 4.1.



```
[{"internal_ip": "10.0.0.1", "pkt_idx": 3, "ip_protocol": 6, "service": "Being analyzed", "external_
ip": "65.33.213.110", "size": -74, "external_port": 23, "ts": 353724449, "conn_id": 31765584954002},
 {"internal_ip": "10.0.0.2", "pkt_idx": 32, "ip_protocol": 17, "service": "Teredo", "external_ip": "
121.211.130.233", "size": -110, "external_port": 50383, "ts": 353724449, "conn_id": 25194287451931},
 {"internal_ip": "10.0.0.3", "pkt_idx": 2, "ip_protocol": 6, "service": "BitTorrent encrypted transf
er", "external_ip": "122.106.135.229", "size": 66, "external_port": 29220, "ts": 353724449, "conn_id
": 6339376970244}, {"internal_ip": "10.0.0.4", "pkt_idx": 1, "ip_protocol": 17, "service": "DNS", "e
xternal_ip": "77.120.115.197", "size": 92, "external_port": 44591, "ts": 353724449, "conn_id": 24253
687657302}, {"internal_ip": "10.0.0.5", "pkt_idx": 8, "ip_protocol": 6, "service": "Being analyzed",
 "external_ip": "84.248.74.123", "size": 201, "external_port": 56613, "ts": 353724449, "conn_id": 62
```

Figure 4.1: Start of JSON file from Procera

The structure of the JSON file is that objects come one after the other and each object consists of several attribute-value pairs, where each object represents a packet. The attributes are as follows:

- **internal_ip**
  The internal IP address. These IP addresses are anonymized by Procera.

- **pkt_idx**
  The packet index, which is the index of the packet within the traffic flow. This means that the first packet within a flow will have the pkt_idx *1*.

- **ip_protocol**
  The IP protocol which is used.

- **service**
  This is the ground truth as predicted by the Deep Packet Inspector. The service is shown on a per packet basis, however it may change under a duration of a flow. So the service for the packet *1* may have the service "Being Analyzed" which could change to "Call of Duty" by packet *5* within the same flow.

- **external_ip**
  The external IP address.

- **size**
  The size of the packet. This also includes information about the direction of the packet, where a negative size means that the packet travels in the up direction with that size, while a positive size means that the packet travels in the down direction with the same size.

- **external_port**
  The external port number.

- **ts**
  A timestamp which can be used to determine the time between packets.

- **conn_id**
  A unique connection identifier which can be used to link packets within the same flow.

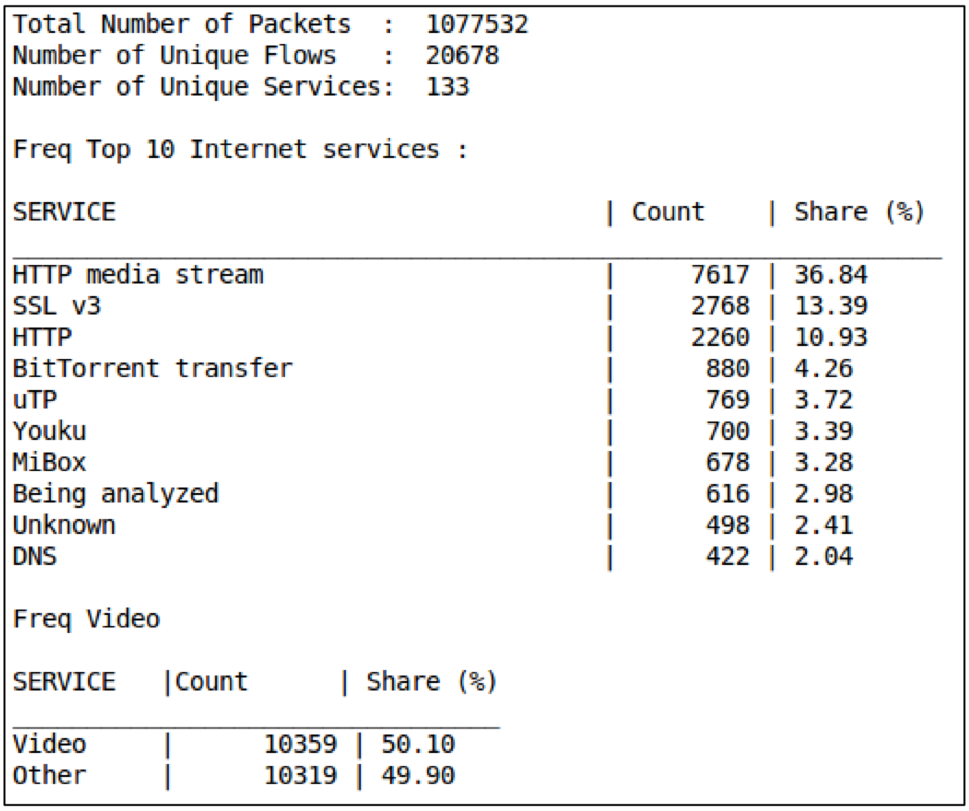
Based on discussions with Procera regarding their stance on how early classification were to be made and on the research previously done on classification of Internet flows, it was decided that the dataset was to contain the 100 initial packets of each flow. This was regarded as a high amount of packets, based on the research. However, it gave room for experimentation.

One problem with the dataset was that it was not effectively stored. This was not a big problem using the local storage but the data also needed to be stored in the storage provided by SNIC.

To alleviate this problem, the dataset was revamped. A few columns which were not used, was dropped. The "service" column had its values changed to service indexes. A table was constructed to be used for translation between the two. A column has also been added to the dataset, “video_flag”, which contains the ground truth which will be used in our machine learning classifier. Which services that was to be regarded as video was received in a list format from Procera. The final in reducing the storage needed for the dataset, was to save it as a different format, namely HDF5 [46].

A few different datasets are prepared this way. Starting with the biggest containing all data, to smaller and smaller subsets of this data. The smallest dataset contains approximately 10k videoflows and 10k non-videoflows. The reason for constructing these subsets is to be able to do tests within reasonable time and on data containing a decent amount of both video and non-video.

In Figure 4.2, stats for the 10k video 10k non-video dataset before filtration are shown.



```
Total Number of Packets  :  1077532
Number of Unique Flows   :  20678
Number of Unique Services:  133

Freq Top 10 Internet services :

SERVICE                                 | Count     | Share (%)
_______________________________________________________________
HTTP media stream                       |      7617 | 36.84
SSL v3                                  |      2768 | 13.39
HTTP                                    |      2260 | 10.93
BitTorrent transfer                     |       880 | 4.26
uTP                                     |       769 | 3.72
Youku                                   |       700 | 3.39
MiBox                                   |       678 | 3.28
Being analyzed                          |       616 | 2.98
Unknown                                 |       498 | 2.41
DNS                                     |       422 | 2.04

Freq Video

SERVICE   |Count       | Share (%)
_________________________________
Video     |      10359 | 50.10
Other     |      10319 | 49.90
```

Figure 4.2: D[illegible] before filtration

4.2 Data Filtration

The next step was to filter the data. The dataset contained a lot of information that was of no use for creating the classification model. Types of data that were removed:

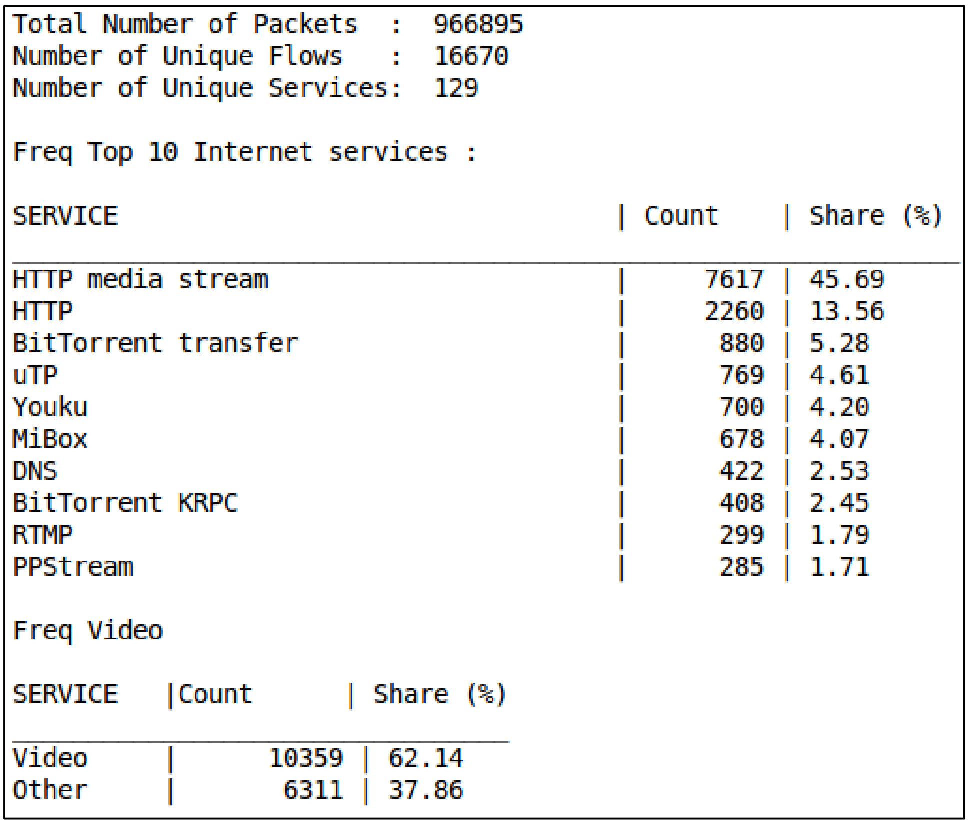
- **Flows starting at pkt_idx > 1**
  Some flows started before the beginning of the data capture. As the early packets are used to determine characteristics, these flows have to be removed.

- **Flows of length < 10**
  Short length flows has limited use for this study, as attempts are made to identify video flows in order to perform actions on these flows during their duration. Because of this, very short flows are removed.

- **Flows without ground truth**
  Supposedly the dataset would contain data with ground truth. However, this is not always the case. Some flows will end up with "Being Analyzed", "Unknown" or other similar "services". These flows obviously need to be removed before using algorithms before feeding the data to a machine learning classifier which uses supervised learning.

- **Flows without "real" ground truth**
  Just like the previous category, these flows do not have ground truth which can be translated to "video" or "not video". They do deserve in some sense their own category as an actual value containing some kind of information is still given as their service. These are flows that receive the service "SSL" or "SSH". The "SSL" and "SSH" services may carry video or non-video data. As there is no sure way to tell which, these type of services are removed.

In Figure 4.3 for the 10k video 10k non-video dataset after filtration are presented.



```
Total Number of Packets   :  966895
Number of Unique Flows    :  16670
Number of Unique Services:  129

Freq Top 10 Internet services :

SERVICE                                  | Count     | Share (%)
_______________________________________________________________
HTTP media stream                        |      7617 | 45.69
HTTP                                     |      2260 | 13.56
BitTorrent transfer                      |       880 | 5.28
uTP                                      |       769 | 4.61
Youku                                    |       700 | 4.20
MiBox                                    |       678 | 4.07
DNS                                      |       422 | 2.53
BitTorrent KRPC                          |       408 | 2.45
RTMP                                     |       299 | 1.79
PPStream                                 |       285 | 1.71

Freq Video

SERVICE    |Count       | Share (%)
__________________________________
Video      |      10359 | 62.14
Other      |       6311 | 37.86
```

Figure 4.3: Dataset after filtration

4.3 Initial Feature Selection

The next step of the process is to decide on features on the data which may be useful for classification. The features were selected based on previous research as well as from discussion with our supervisor. The selected features were:

- **BytesDwUp**

  The total amount of bytes sent in both the up and down direction.

- **BytesDw**

  The total amount of bytes sent in the down direction.

- **BytesDwPayl**

  The total amount of payload bytes sent in the down direction.

- **NrDownPkts**
  The total amount of packets sent in the down direction.

- **NrDownPaylPkts**
  The total amount of payload packets sent in the down direction.

- **PosSumDwnPkts**
  The position sum of the packets in the down direction. This is calculated as the sum of the packet indexes which correspond to a packets being sent in the down direction. So if we have packets in the down direction at indexes *5* and *7*, the position sum would be calculated as *5+7 = 12*.

- **PosSumDwnPaylPkts**
  The position sum of the payload packets sent in the down direction.

- **PosSumDwnPktsEven**
  The position sum of packets in the down direction. Only packets in the down direction which are on even indexes are counted toward the sum.

- **PosSumDwnPaylPktsEven**
  The position sum of payload packets in the down direction. Only payload packets in the down direction which are on even indexes are counted toward the sum.

- **StdDevDwnPkts**
  Standard deviation of the position of packets in the down direction. Basically a measurement for how spread out the packets in the down direction are.

- **StdDevDwnPaylPkts**
  Standard deviation of the position of payload packets in the down direction.

- **DwnPaylRun**
  The longest streak of payload packets in the down direction.

- **DwnPaylRun200**
  The longest streak of payload packets in the down direction with the condition that all packets have a maximum inter arrival time of *200*ms.

- **IATUp**
  Inter arrival time of packets in the up direction.

- **IATDwn**
  Inter arrival time of packets in the down direction.

- **PaylSize-x**
  The size of the payload of individual payload packets, where *x* denotes the number of the payload packet. So the feature “PaylSize-4” will contain the payload size of the 4th payload packet.

- **PaylMean**
  The mean of all payload sizes.

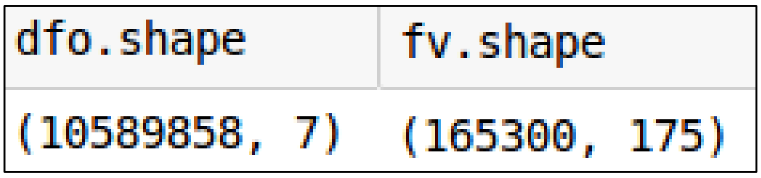
- **PaylVar**
  The variance of all payload sizes.

- **PaylStd**
  The standard deviation of all payload sizes.

Additionally, most of these features can be gathered at every other packet, which for example means that we can have the total amount of bytes sent at 2, 4 … 18 and 20 packets. This means that this 1 feature can be spread into 10 features or more, if desired. However, this is a massive amount of features, which can preferably be reduced without significantly affecting the finished classifiers performance.

4.4 Feature Vector Creation

The next step of the process is to create the feature vector. The feature vector is the structure in which data is placed in a format that can be used by the machine learning classifier. The Procera dataset contained information on a packet basis. The feature vector is going to contain information on a flow basis. Each flow will contain the information listed in the previous section. This means that this vector will be much shorter but much wider than the Procera dataset.



dfo.shape	fv.shape
(10589858, 7)	(165300, 175)

Figure 4.4: Shape of original dataset and feature vector created from it

In Figure 4.4, the difference in shape between the original dataset *dfo* and the feature vector *fv* which has been created from the original dataset after filtration can be seen. The *dfo* dataset, contains 100k video and 100k non-video flows. The shape is written as (*x*, *y*), where *x* is the amount of rows in the data and *y* is the amount of columns. As can be seen, the feature vector is not smaller than the original dataset. The reason for this is that a lot of information has been extracted and put into features.

4.5 Machine Learning

The next step is to create the classifier. In order to do this, the Decision Tree classifier and the Random Forest classifier supplied by the Sci-kit learn module are used.

4.5.1 Separation of Data

Before data is being fed to the machine learning classifier, separation of the data is needed. The feature vector contains information about each flows ground truth. This data is separated from the features which will be used for classification. Additionally, both the feature data and the ground truth need to be separated into training data and validation data. The separation process is illustrated in Figure 4.5.

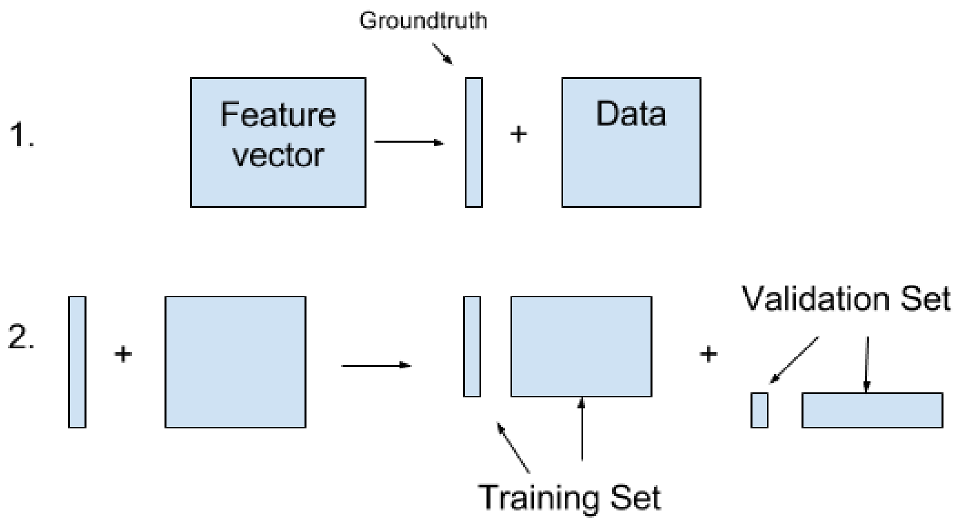


Figure 4.5: Separation of data

4.5.2 Training the Classifier

The training set is then used to create the model by using it to train the classifier. There are also some classifier-specific parameters which can be chosen to tune the classifier to behave in a specific way as seen in 5.2 and 5.3. The training process can be seen in Figure 4.6.

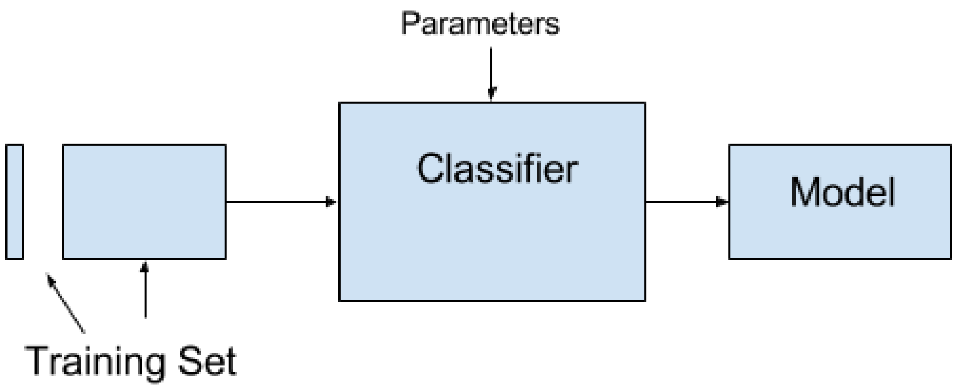


Figure 4.6: Training the classifier

4.5.3 Evaluating the Model

After the model has been created, it can be fed with the validation set excluding the ground truth. The model outputs the predicted classes for the validation set. The predicted values can then be compared to the ground truth to produce a confusion matrix, described in section 2.9.

The information contained within the confusion matrix can then be used to further evaluate the accuracy of the model. This process is illustrated in Figure 4.7.

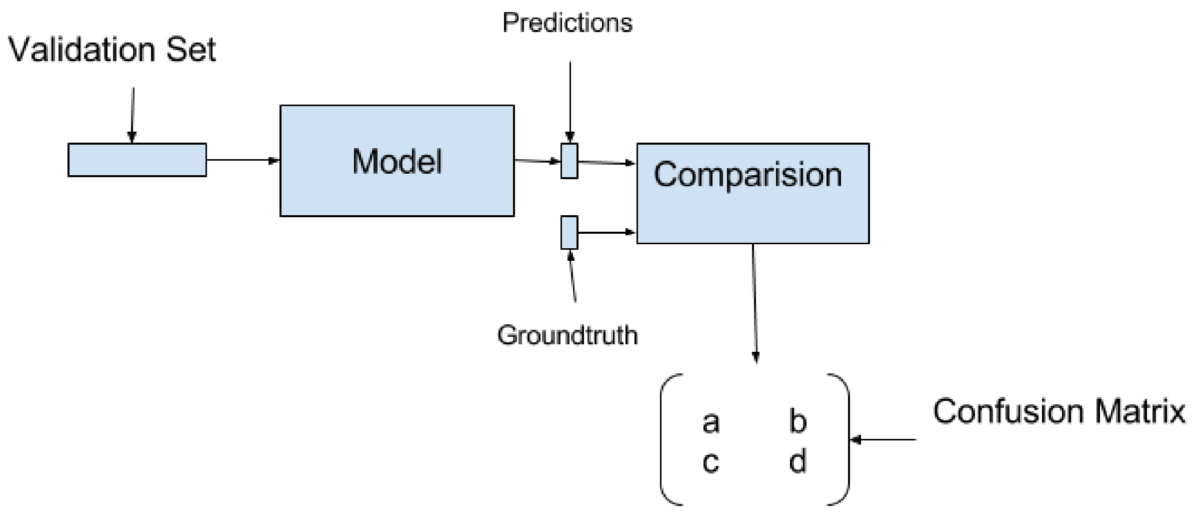


Figure 4.7: Evaluating model

4.6 Chapter Summary

In this chapter, the process of the study was described in some detail. Starting from the data set retrieved from Procera, a description of the process of filtering the data and preparing it for the classifier was presented. A description of the process of creating the model using a machine learning classifier was also presented. Furthermore, a discussion of how the model can be used to evaluate the performance was also shown.

5 Results and Evaluation

This chapter contains the results produced by the machine learning classifiers.

In the pre-classification tests section, a look is taken at how manipulation of features can be done, before sending the feature vector to the classifier. This is done both to possibly improve the results or to remove unnecessary features before classification.

In the following two sections, tests are conducted using the decision tree and random forest classifiers respectively. These tests are conducted in order to decide which parameter values the classifiers will use in the final test. The purpose of this, is to achieve the greatest result possible in the final test.

These three initial sections are linked since the outcomes of the tests in the pre-classification section will be used in both parameter tuning sections.

In the final results section, the results of the preceding tests are combined in order to achieve the greatest possible result. This includes results for both classifiers, which allows for a comparison between them.

Furthermore, a summary of Simon´s results are presented which are then compared to these results.

5.1 Pre-classification Tests

In this section, tests are carried out by manipulating the data in various ways before it is fed to the classifier. The two initial tests are conducted to learn which data is the most useful to feed to the classifiers. This way, no unnecessary data need to be used during the subsequent tests. In the third test, different scopes for classification are looked at. This is done to show a different way of possibly improving the results. The test conducted in this section uses the 10k video, 10k non-video dataset, the stats of which is presented in Figure 4.3.

5.1.1 Feature Testing

One of the important aspects of this study, was to figure out which features were to be used and which features were to be ignored. At the beginning of the testing, the features used was the entire feature set discussed in section 4.3, including features at every second packet ranging from 6 to 20. During the testing, the possibility of not using the range thresholds and which features that could be ignored was something that came up as something that needed to be tested.

In order to see if the range was needed to produce good results, one feature vector was created containing the range and another simply containing the features at the last packet, which in this case was *20*. In Figure 5.1, these “feature packages”, are known as *6-20* and *20* respectively.

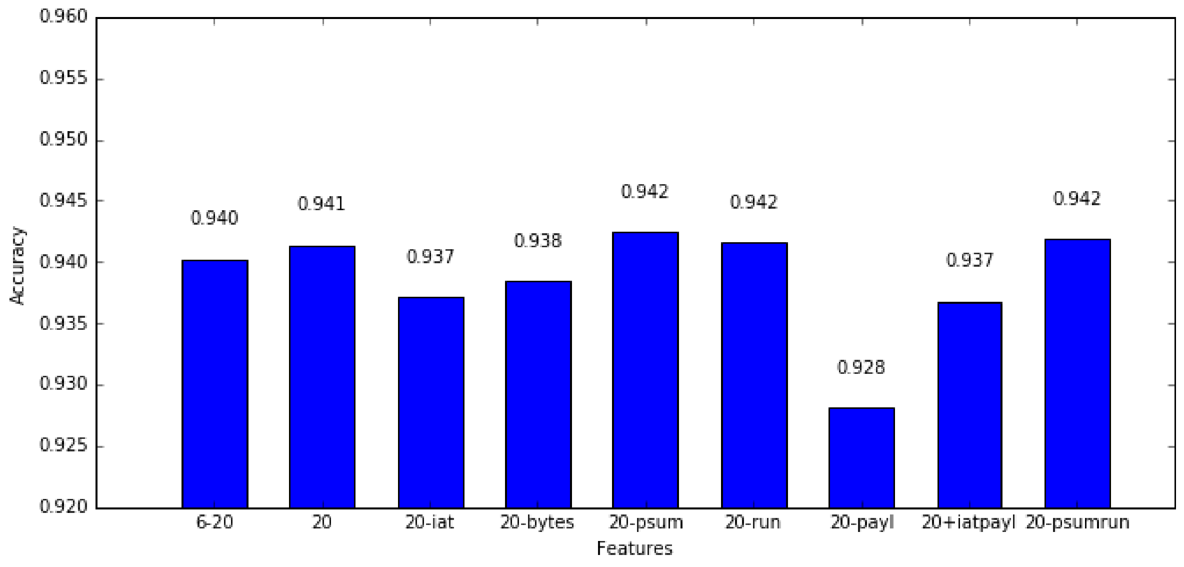


Figure 5.1: Testing the accuracy of feature packages

To produce the results shown in Figure 5.1, the following was used:

- **Classifier:** Random Forest
- **Criterion:** Gini
- **Number of trees:** 60

These parameters will be discussed in more detail in section 5.3.

The results for *6-20* and *20* shows that the feature package that do not contain the range, has a *0.1* percentage unit advantage over the feature package that do contain the range. It would be dangerous to conclude that this means that not having the range is better than having the range, however it would at least suggest that having the range may be unnecessary.

Building on these result, a few feature packages without range was constructed to see if more unnecessary features could be removed. To construct these packages, similar features were packed together and was either included together or not included at all. These feature packages were as follow:

- **20-iat**
  Based on the *20* feature package, it has all the features of *20* except the ones which contains inter arrival time information.

- **20-bytes**
  Based on the *20* feature package, it has all the features of *20* except the ones which contains information on the amount of bytes being transferred.

- **20-psum**
  Based on the *20* feature package, it has all the features of *20* except the ones which contains information related to position sums.

- **20-run**
  Based on the *20* feature package, it has all the features of *20* except the ones which contains information related to the longest run of payload packets.
- **20-payl**
  Based on the *20* feature package, it has all the features of *20* except the ones which contains information on sizes within payload packets.

- **20+iatpayl**
  Based on the *20* feature package, it has only the features which are related to inter arrival time and payload sizes.

- **20-psumrun**
  Based on the *20* feature package, it has all the features of *20* except the ones which contains information on position sums and the longest run of payload packets.

Looking at the result of the feature packages where one type of feature is removed, some interesting results can be seen. First off, removing the information on position sums as well as longest payload run, does not seem to negatively affect the accuracy of the classifier. However, removing other types of features do negatively affect the accuracy of the classifier. Most noteworthy, removing the information on payload sizes makes the accuracy suffer a loss of over 1 unit of percentage.

With these results in mind, a test of what would happen if only the seemingly most important features inter arrival time and payload sizes were used, was conducted. Another test, to remove both of the features which did not seem to have an impact on the classifier were also performed. As the results shows, using the *20+iatpayl* package gave the same result as *20-iat*. Additionally, removing both position sums and longest run of payload packets, did not negatively affect the results.

When trying to make conclusion out of these results, there needs to be an understanding that the packages are not equal. What is meant by that is that there is way more features based on inter arrival time than there is of the longest run of payload packets. As such, it would be nonsensical to conclude, from these tests alone, that longest run is a less effective feature than inter arrival time. There is also the possibility that accepting some loss in accuracy if that means losing some features, may be the best option.

5.1.2 Amount of Packets

When looking at how the amount of packets might impact the accuracy of the classifiers, there are a couple of things to consider. Amount of features depend on the amount of packets which are considered. It does not have to be this way, but it is the current state of the features. As such if the accuracy raises for a greater amount of packets, that might be a sign that more features means higher accuracy, not necessarily that the features produced for that amount is better. Another possibly important factor is to accurately classify a flow as early as possible, which in the case that a smaller amount of packets produces a little worse accuracy, it may be a tradeoff worth having, depending on the priorities of the application.

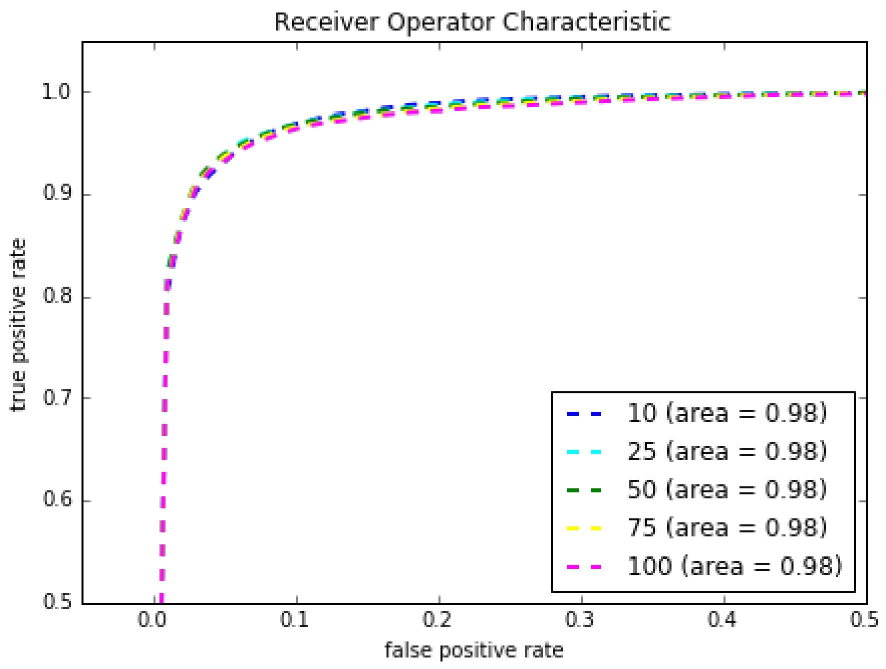


Figure 5.2: ROC curve for different amount of packets

To produce the results shown in Figure 5.2, the following was used:

- **Classifier:** Random Forest
- **Criterion:** Entropy
- **Number of trees:** 60
- **Max features at each split (see 5.3.3):** 20

In the graph it is very hard to see the majority of the roc curves. This is because the all the ROC curves are almost the same. The amount of packets does not seem to impact the accuracy of the classifier for the sizes chosen. In fact, the classifier which used features for 100 packets has the worst ROC curve. However, with these minor differences is not considered significant and may have occurred because of the randomness which exist in random forests. Without any detectable differences between packet sizes, *10* packets are decided as the amount of packets to base the creation of features on.

5.1.3 Different Classification Scope

In this test, a look is taken at the performance of the classifier when it tries to classify one type of video service. It is then compared to the results of classifying all video services as one class.

The service which were used were the "HTTP media stream" service, which is the most common type of video service within the dataset.

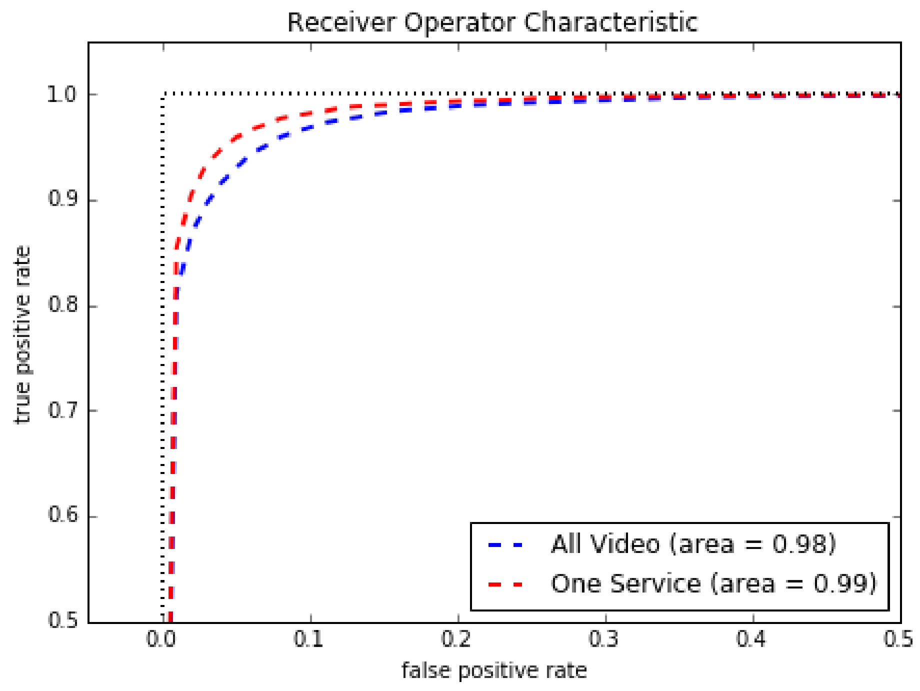


Figure 5.3: ROC curve for two classification scopes

To produce the results shown in Figure 5.3, the following was used:

- **Classifier:** Random Forest
- **Criterion:** Entropy
- **Number of trees:** 60
- **Max features at each split:** 20

As can clearly be seen by looking at the two different ROC curves, the one service classifier managed to achieve a greater accuracy than the classifier for all video. There are several possible reasons for this, one might be that within the services which are classified as video, there might be services which are problematic. Problematic in this case means that their features would suggest that they should be classified as "other" instead of video. By focusing on just one service, these kind of services will no longer negatively affect the accuracy, hence the improvement.

5.2 Decision Tree Parameter Tuning Results

There are quite a few options available when tuning a decision tree classifier. However, there is only a couple which will be tested. The reason for this is that a lot of the parameters are there to control the shape of the tree similarly to the *max depth* parameter. These are parameters such as the minimum number of samples required to split a node and maximum allowed leaf nodes. The parameters which will be looked at are the *criterion* and the *max depth* parameters. The remaining parameters will retain their default values.

5.2.1 Criterion

The criterion parameter decides the criterion for which to best split the data. The feature which best splits the data based on this criterion is chosen at each split. There are two different criterions to choose between for the decision tree classifier, namely *gini* and *entropy*, where *gini* is the default value. The difference in how these two different criterion splits the data will not be part of this dissertation.

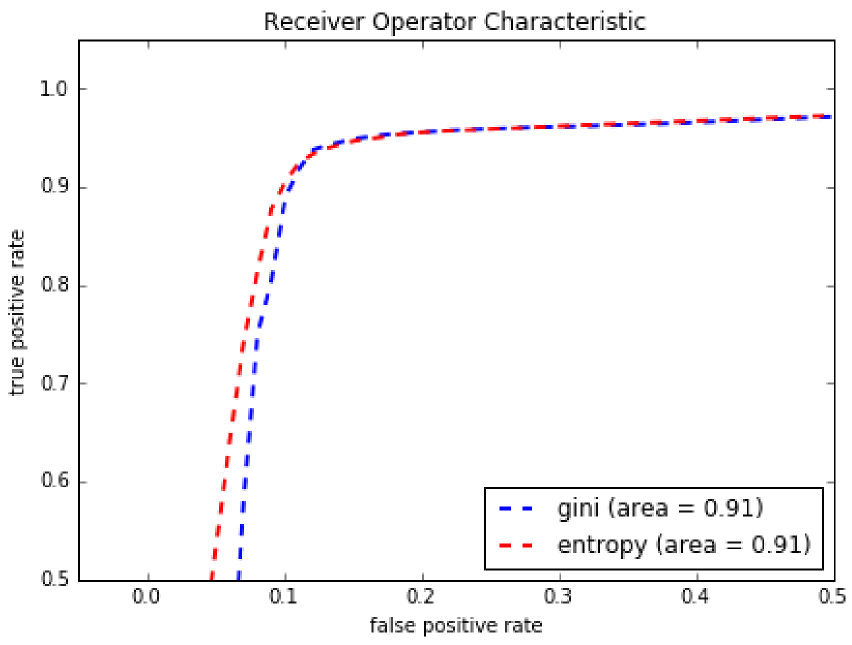


Figure 5.4: ROC curve for different criterion

To produce the results seen in Figure 5.4, the following was used:

- **Classifier:** Decision Tree
- **Max Depth:** 13

In the graph, it appears that *entropy* has the slight advantage over *gini*. One suspicion is that this does not actually mean that *entropy* is better than *gini*. The suspicion comes from the result of the random forest tuning of the same parameter which can be seen in section 5.3.1. The expectation was that the criterion would have similar results in both decision trees and in random forests because of the connection they share. One possible reason that the results of the decision tree became so different, may be because not enough trees were used, so the average values did not quite balance out. In random forests, each forest contains many trees, allowing the accuracies to balance out.

In any case, *entropy* is chosen as the criterion of choice. If it is as expected, that the random forests comparison is more accurate, then either criterion would be a good choice.

5.2.2 Max Depth

The *max depth* parameter, as can be expected, is used to control the depth of the decision tree. Setting a max depth limits the overfitting of the data. As such, the decision tree will generalize better to new data and hopefully produce better results. Before starting the tests, a decision tree without a *max depth* set, was produced. This tree had a depth of *33*, as such, tests were done with the *max depth* parameter ranging from *1* to *40.*

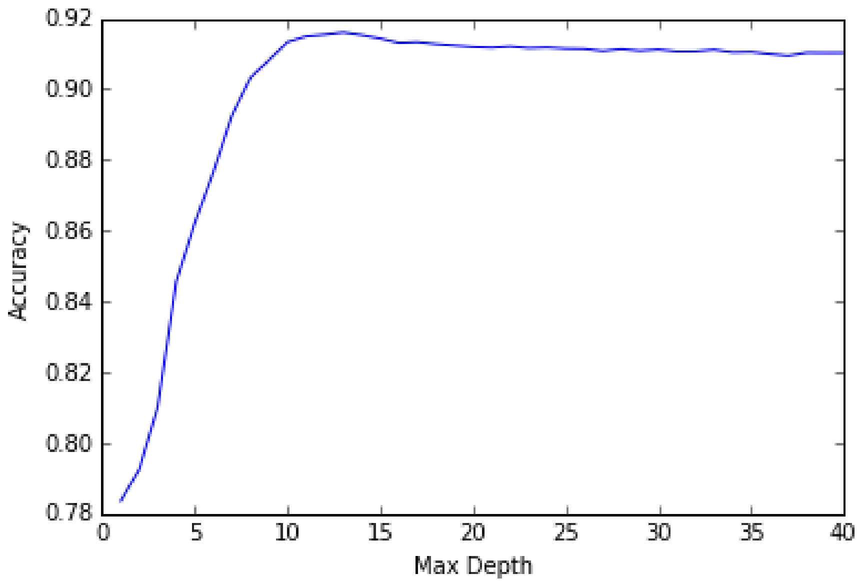


Figure 5.5: Accuracy vs max depth

To produce the results shown in Figure 5.5, the following was used:

- **Classifier:** Decision Tree
- **Criterion:** Gini

Looking at the results, conclusions can be made that either this particular decision tree does not suffer much from overfitting or the max depth value does not manage to fix the problem. However, there is almost a *1* unit of percentage of improvement using a *max depth* of *13* which is not an insignificant improvement.

5.3 Random Forest Parameter Tuning Results

When trying to tune the parameters of the random forest classifier, there are quite a few options present. However, based on research, the main ones to pay attention to, is the number of trees used and the amount of features considered at each split. Other parameters include max depth of trees and minimum allowed samples within a leaf. As a bonus, a look is also taken at the criterion parameter which could potentially impact the performance of the accuracy by choosing different features. Other parameters will retain their default values.

5.3.1 Criterion

This parameter decides the method of deciding which feature that best splits the data. It has two available options, *gini* and *entropy,* where *gini* is the default value. A ROC curve is produced to find which is the better option.

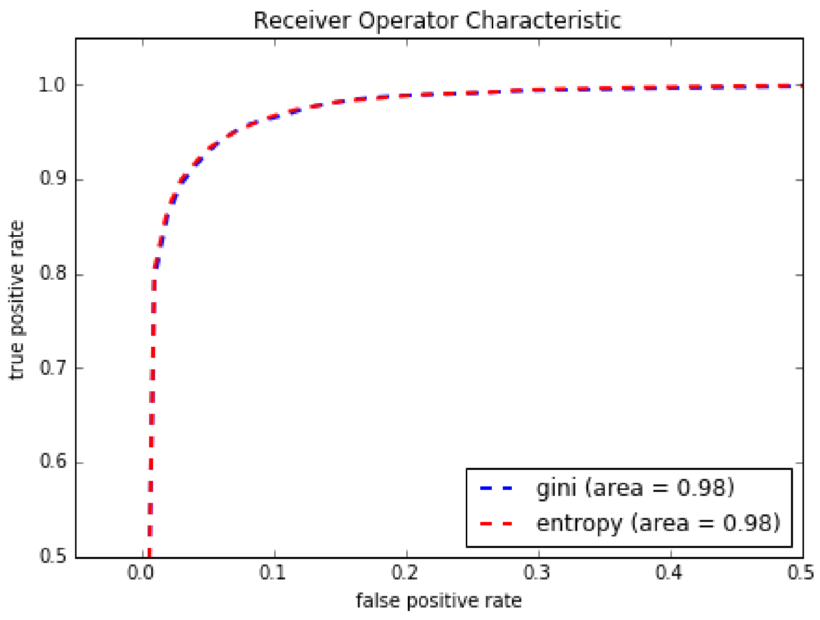


Figure 5.6: ROC curve for different criterion

To produce the results shown in Figure 5.6, the following was used:

- **Classifier:** Random Forest
- **Number of trees:** 60
- **Max features at each split:** 20

The result show that the difference is insignificant and it seems unlikely that the choice of criterion will matter for this classifier. However, where the two curves do differ, the advantage is in favor of *entropy*.

5.3.2 Number of Trees

Having as many trees as possible is good to increase the accuracy of our classifier, however it will also make our classifier slower. If there is barely any difference in the results of using *5* or *5000* trees, there would be no reason to choose *5000* trees. As such, there is a search to find the "sweet spot", where the accuracy of our classifier no longer increases with the adding of new trees.

Accuracy Test

Based on some initial sampling, testing of the accuracy was done up to a size of *100* trees.

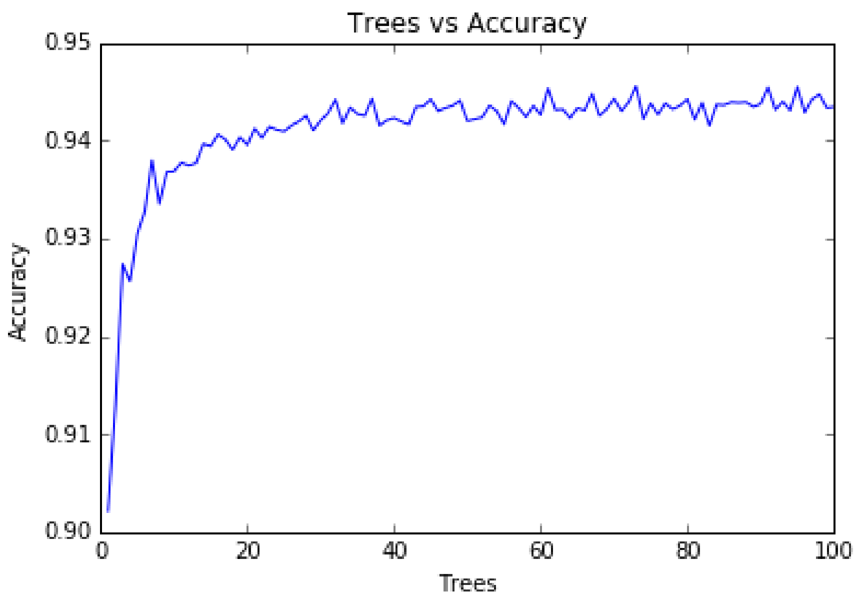


Figure 5.7: Trees vs accuracy

To receive the results presented in Figure 5.7, the following was used:

- **Classifier:** Random Forest
- **Criterion:** Entropy
- **Max features at each split:** 20

When looking at these results, special attention needs to be paid to the accuracy values. If the value of the first data point is ignored, the difference between the lowest accuracy value and the highest accuracy value is only about 2 units of percentage. The problem with the graph presented in Figure 5.7, is that it seems to still slowly be increasing. To test if this is true, another value was added to the graph to confirm or deny this possibility. A greater number of trees was needed and the amount chosen was *300*. With the help of this extra data point, a decision can be made whether the accuracy is increasing fast enough for a higher amount of trees to be chosen.

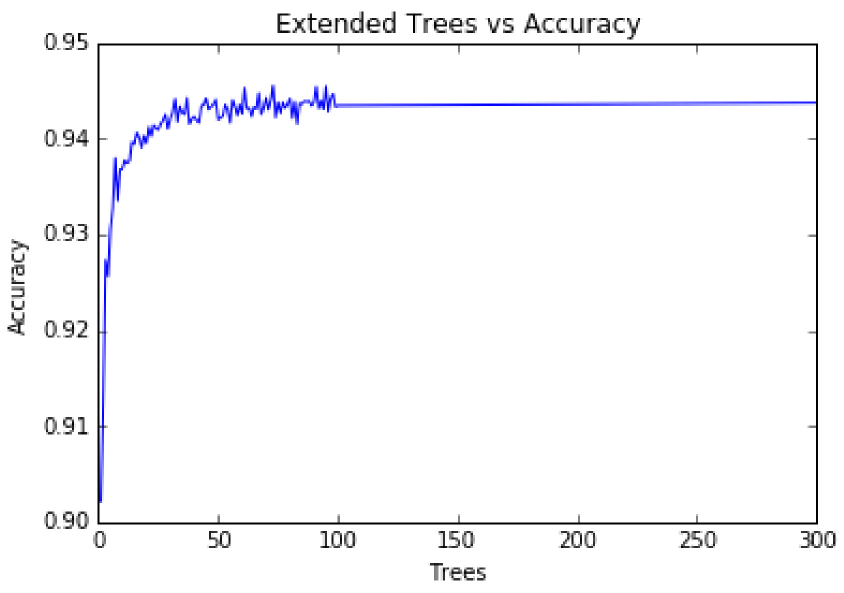


Figure 5.8: Extended trees vs accuracy

To receive the results shown in Figure 5.8, the same classifier and parameters as in Figure 5.7 was used. Another value was simply added to the graph.

Looking at the extended graph in Figure 5.8, conclusions can be made that while the accuracy may still be increasing beyond 100 trees, the incline at which it does so would suggest that it can be ignored. Based on the results in Figure 5.7 and Fig, Figure 5.8, *60* trees are chosen as a good enough number of trees to produce a good result.

ROC Test

While the accuracy of classification seemed to make it accurate enough using just *60* trees. There was a risk that using relatively few trees would negatively affect the probabilities produced to create the ROC curve. This was something which had to be investigated so that a possible flaw of using less trees were not left undiscovered. To test this, the ROC curve for *60* trees as well as the ROC curve for *300* trees was plotted.

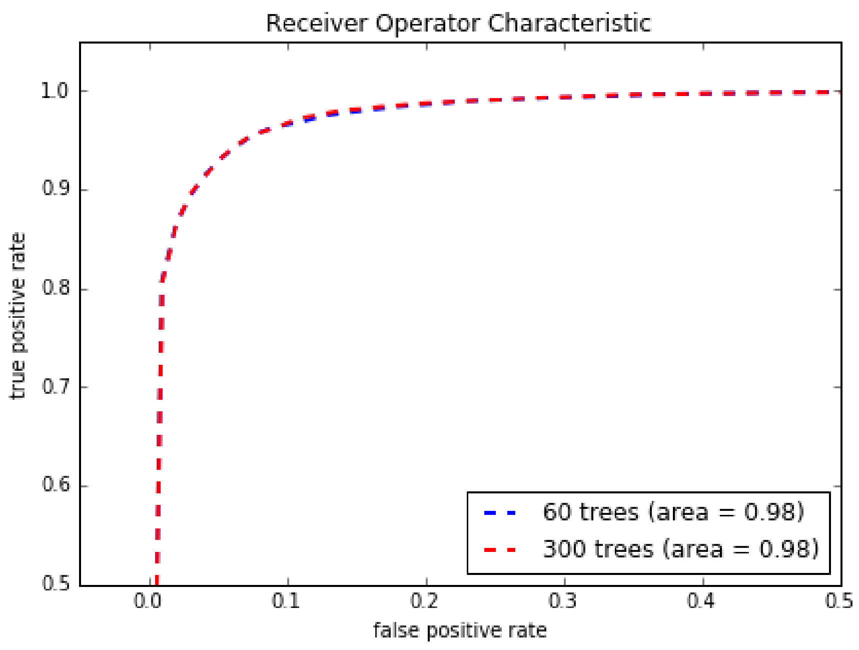


Figure 5.9: ROC curve of 60 and 300 trees

To produce the results seen in Figure 5.9, the following was used:

- **Classifier:** Random Forest
- **Criterion:** Entropy
- **Max features at each split:** 20

Looking at the two ROC curves, it is hard to see that there are actually two curves present. Looking carefully, at least *1* location can be spotted where the ROC curve for *60* trees attain slightly worse values than that of the curve produced by *300* trees. The difference between the two is so small that it can be deemed insignificant. As such, *60* trees are chosen as a good number of trees for this classifier.

5.3.3 Max Features at Each Split

The other important parameter which need to be tested is the max features parameter. The max feature parameter controls how many features are considered at each split of a node. A higher value will make each split more useful, but will make the individual trees less random which may hurt the classifier as a whole. Having a very low amount of features considered may also be a bad idea. The point of tuning this parameter is to find the amount of features to consider which provides the highest accuracy.

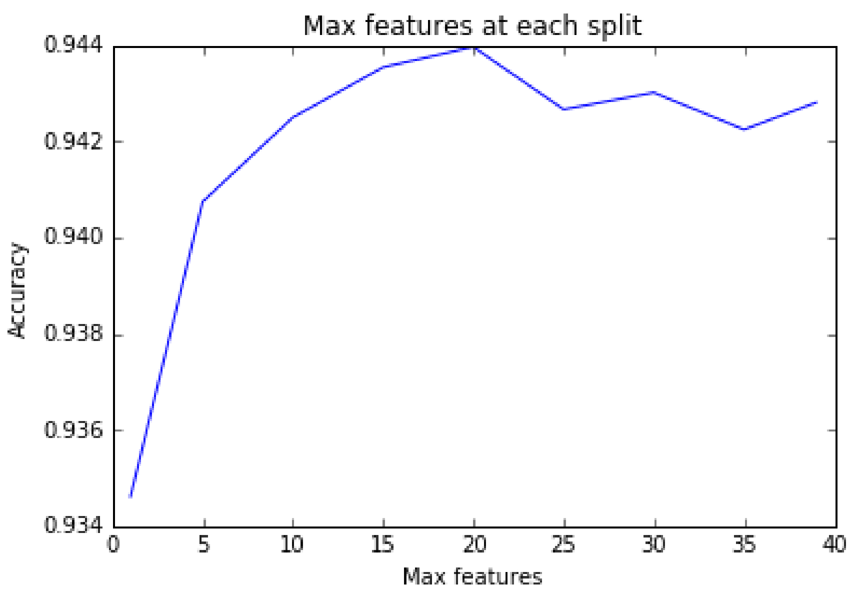


Figure 5.10: Accuracy for different max feature values

To receive the results presented in Figure 5.10, the following was used:

- **Classifier:** Random Forest
- **Number of Trees:** 60
- **Criterion:** Entropy

A step size of *5* was used with multiple runs to average out the results.

The results show that the accuracy increases until *20* max features. After *20* features, the accuracy goes down slightly. Looking at the actual values of the different accuracies, there is only about a *1* unit of percentage difference when comparing the best and the worst accuracies. Based on these results, it is decided that the amount of features to consider at each split is *20* features.

5.4 Final Results

In this section, the information gained from the testing done in sections 5.1, 5.2 and 5.3, are used to try and achieve a greater accuracy for both classifiers. A bigger dataset is also used. The dataset contains about *100 000* “other” flows and about *100 000* video flows, before filtration. First, the results for the decision tree classifier is presented, followed by the results of the random forest classifier. Additionally, a comparison of the results of the two classifiers is also presented.

5.4.1 Decision Tree

To produce the results of the decision tree classifier, the following was used:

- **Classifier:** Decision Tree
- **Criterion:** Entropy
- **Max depth:** 13

The total time required for classification was *22.900* seconds. This time consisted of a training time of *22.872* seconds and a testing time of *0.028* seconds. This time was taken doing a cross-validation using 10 folds. In a 10-fold cross-validation, all data is tested once but all data is trained on nine times. The training time does not matter for the final model as once the model is made, there is no further need to train the model. The time for testing is interesting since this is indicative of the time that it will take to do predictions on new data once the model has been implemented.

The results of the decision tree classifier can be presented in the form of a confusion matrix, as can be seen in Table 5.1. By inspecting the confusion matrix, the results seem quite promising. Out of more than *100 000* video flows, less than *5000* are classified as “other”.

	Predicted Other	**Predicted Video**
Actual Other	57474	5098
Actual Video	4545	98181

Table 5.1: Confusion matrix from the decision tree classifier

The information contained in the confusion matrix can be used to compute different metrics. The metrics which were computed are presented in Table 5.2.

	Precision	**Recall**	**F1 Score**	**Overall Accuracy**
Other	0.927	0.919	0.923	0.942
Video	0.951	0.956	0.953	

Table 5.2: Metrics from the decision tree classifier

The metrics presented in Table 5.2, show a high overall accuracy of *94.2* % for the decision tree classifier. For the precision, recall and F1 score, video has greater values than other. This means that there is a greater percentage of video predicted as video than other being predicted as other and within these classes, the video class contain more relevant flows, percentage wise.

Because of the reasons explained in section 2.9, the precision and F1 score will not be paid much attention to.

The recall metric show that 95.6 % of videos are correctly classified as videos. This is a very good value since getting as much video as possible predicted as video is important.

A comparison between these results and the results of the random forest classifier is done in section 5.4.3.

5.4.2 Random Forest

To produce the results of the random forest classifier, the following was used:

- **Classifier:** Random Forest
- **Criterion:** Entropy
- **Number of trees:** 60
- **Max features at each split:** 20

The total time required for classification was *71.222* seconds. The time consisted of a training time of *69.749* seconds and a testing time of *1.473* seconds. The time was taken doing a cross-validation using 10 folds. In a 10-fold cross-validation, all data is tested once but all data is trained on nine times.

The result of the random forest classifier come in the form of a confusion matrix. The confusion matrix can be seen in Table 5.3. The values contained within the confusion matrix suggests a very good accuracy of classification. Of over *100 000* video flows, only about *3000* flows were not correctly classified as video.

	Predicted Other	**Predicted Video**
Actual Other	60052	2520
Actual Video	3288	99438

Table 5.3: Confusion matrix from the random forest classification

From the information contained within the confusion matrix in Table 5.3, different metrics can be computed. These metrics are presented in Table 5.4.

	Precision	**Recall**	**F1 Score**	**Overall Accuracy**
Other	0.948	0.960	0.954	0.965
Video	0.968	0.968	0.968	

Table 5.4: Metrics from the random forest classification

As can be from the values presented in Table 5.4, the overall accuracy for the random forest classifier reached a very high score of *96.5*%.

The recall score for video flows is just slightly higher than the overall accuracy, landing at *96.8*%.

The metrics from the random forest classifier indicates a very balanced model where most metrics are close to the overall accuracy.

5.4.3 Decision Tree/Random Forest Comparison

The difference in time taken between the classifiers is 3x as long training time and 52.5x as long testing time for the random forest classifiers. Having so much longer testing time may make the random forest classifier unusable with its current parameters. This must be taken into consideration before deciding on which classifier to use.

The results for the decision tree classifier shown in Table 5.2, can be compared to the results of the random forest classifier shown in Table 5.4.

	Precision	**Recall**	**F1 Score**	**Overall Accuracy**
Other	0.021	0.041	0.031	0.023
Video	0.017	0.012	0.015	

Table 5.5: Difference in metrics between the two classifiers

The values presented in Table 5.5 has been calculated by subtracting the metric values of the decision tree classifier from the metric values of the random forest classifier. The values show that for all calculated metrics, the random forest classifier outperforms the decision tree classifier. Especially the recall score for “other” seem to suffer for the decision tree classifier. This would suggest that the biggest difference between the two classifier is that more flows that are supposed to be classified as “other” are instead classified as video in the decision tree classifier. The overall accuracy difference between the classifiers amount to about *2.3* %.

ROC-curve comparison

To better illustrate the difference between the two classifiers, a ROC-curve is produced for each classifier. The ROC-curves can be seen in Figure 5.11.

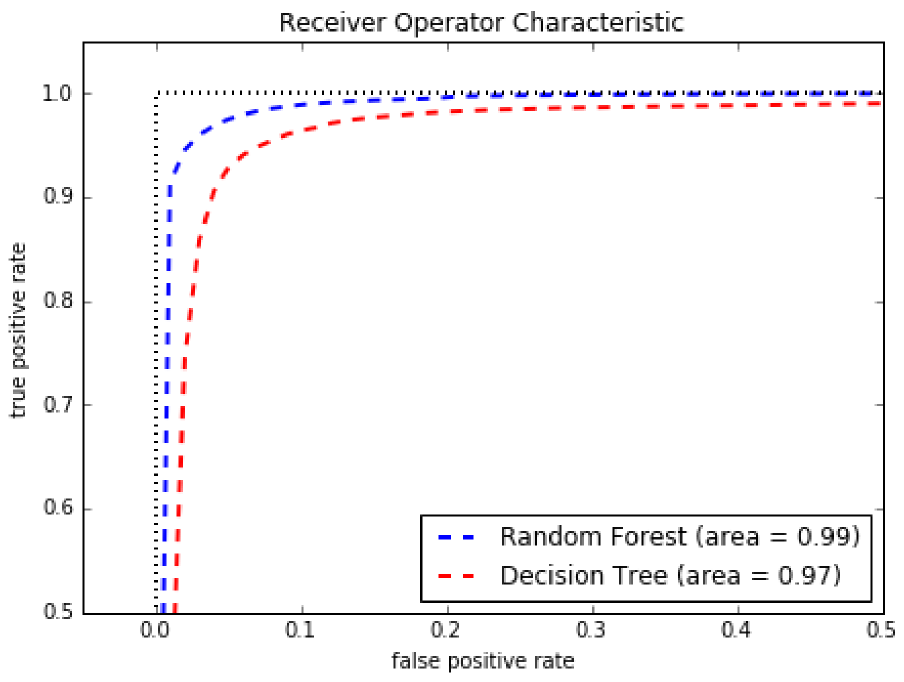


Figure 5.11: ROC comparison, decision tree vs random forest

As with the metric values, the ROC-curve of the random forest classifier outperforms the ROC-curve of the decision tree classifier. One thing that must be noted, is that the difference between the classifiers depend on the chosen threshold. So the difference between the classifiers might be relatively large at their optimal value but if you want to maximize true positive rate at the cost of false positive rate, the difference shrinks. However, the ROC-curves do solidify that if the strength of the classifier is the only thing in consideration, random forest is the obvious choice between the two.

5.5 Summary of Simon´s Results

This is a summary of the best results taken from Simon Westlinder´s parallel study [47].

In the results performed by the different algorithms based on support vector machines, the most accurate classifier was the SVC classifier using a radial basis function kernel. Classification were done using the first 50 packets. The dataset used for this result was the *100 000* video flows and *100 000* “other” flows which was also used in section 5.4. However, due to performance issues, only *9000* flows of each category was used in the evaluation. A feature selection step was also used, using the first 25 individual payload sizes and the average, standard deviation and variance of the 50 first payload sizes. That makes a total of 28 features.

	Predicted Other	**Predicted Video**
Actual Other	8708	372
Actual Video	1177	7823

Table 5.6: Confusion matrix from SVC classifier

The confusion matrix produced by the classification is shown in Table 5.6. The information contained in the confusion matrix is used to derive the metrics shown in Table 5.7.

	Precision	**Recall**	**Overall Accuracy**
Other	0.8809	0.9590	0.9143
Video	0.9546	0.8692	

Table 5.7: Metrics from SVC classifier

The SVC classifier achieves an overall accuracy of *91.43* %. For video flows, recall is sacrificed for a greater precision, what this means is that identifying video flows is relatively poor but less “other” flows are incorrectly classified as video flows.

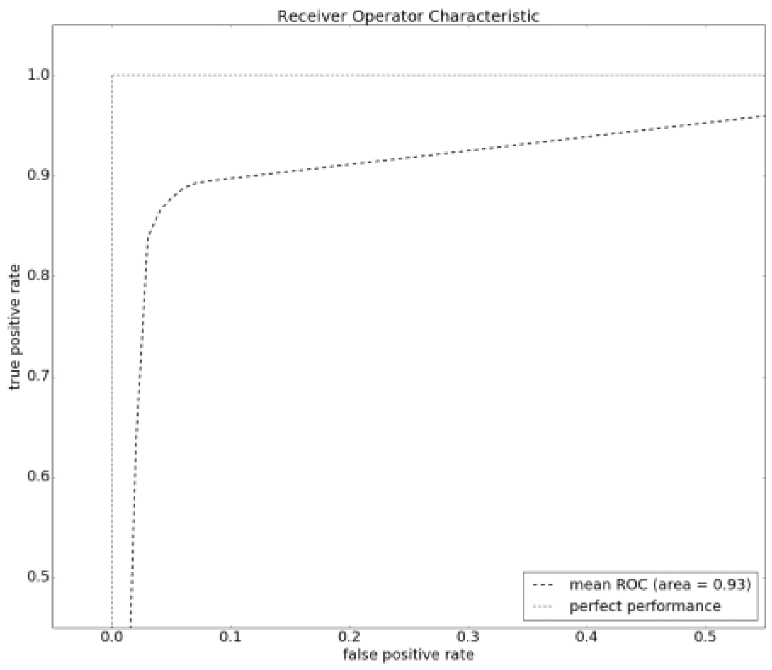


Figure 5.12: ROC-curve, SVC classifier

A ROC-curve for the classifier is shown in Figure 5.12. As can be seen in the graph, the false positive rate is kept low however relatively few video flows are successfully classified as video. The ROC-curve verifies the results presented in Table 5.7.

5.6 Evaluate Results

This is a comparison between the results of the tree based classifiers and the SVC classifier.

For any conclusion made, one must take into account the differences between the tests. The SVC classifier uses a smaller feature set for classification but makes the classification based on information gained on a greater amount of packets.

	Precision	**Recall**	**Overall Accuracy**
Other	-0.0461	0.04	-0.0277
Video	0.0036	-0.0868	

Table 5.8: SVC classifier metrics as compared to the decision tree metrics

In Table 5.8, the metrics from Table 5.2 has been subtracted from the metrics of Table 5.7. This shows how accurate the SVC classifier is compared to the decision tree classifier.

The overall accuracy of the SVC classifier is almost *3* percentage units worse than that of the decision tree classifier. The SVC classifier manage to get less false positives for video flows but at the cost of correctly identifying almost *9* percentage units less video flows.

Based on this comparison and the comparison between the decision tree classifier and the random forest classifier in section 5.4.3, there is no need of comparing the metrics of the SVC classifier with the metrics of the random forest classifier. However, a ROC-curve comparison between all three classifiers will be provided.

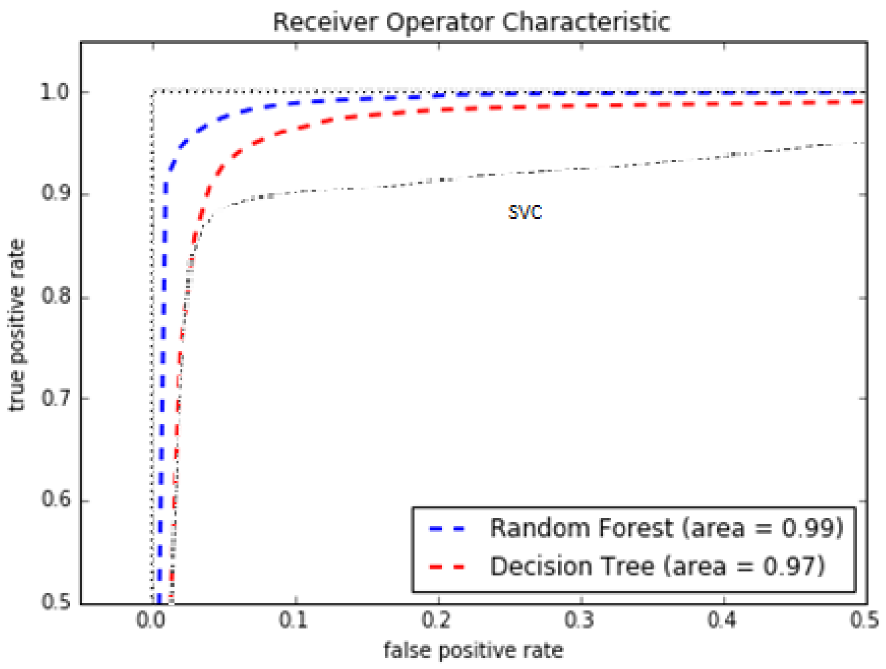


Figure 5.13: ROC-curve comparison, SVC, decision tree and random forest classifiers.

The ROC-curve comparison shown in Figure 5.13, has been created by merging Figure 5.11 and Figure 5.12. Because of this, the quality of the SVC ROC-curve has degraded but it is still useful for analysis.

As can be seen in the figure, the ROC-curve for the SVC classifier has a similar accuracy to the decision tree classifier for small false positive rate. If false positive minimization is not the priority however, the decision tree classifier has much greater accuracy than the SVC classifier. The random forest classifier provides the greatest accuracy of all three classifiers.

As previously mentioned, the SVC classifier did not use as much data for classification because of the time it takes to train and make predictions on a large dataset. The decision tree and random forest classifiers did not have these issues which is another advantage of the decision tree and random forest classifiers besides the accuracy advantage.

6 Conclusions

In the previous chapter, an evaluation of the results of the decision tree classifier and the random forest classifier was presented. In this chapter, the aim is to give an evaluation of the study as a whole. To achieve this, attempts are made to answer the following questions:

- "Can we successfully identify video flows based on their characteristics in terms of features such as payload sizes and inter-arrival time?"
- "Can the classification models predict the category of flows which the Procera DPI software cannot?"

Conclusions are drawn based on these questions. Additionally, an evaluation of the study is presented and a look is taken at possible work that can be done to extend this study. The dissertation is finalized with some concluding remarks.

6.1 Concluding Results

Based on the results presented in chapter 5, classification of video flows based on the characteristics of the flow is definitely possible. For different applications there may be different needs from the classifier, not just in overall accuracy but some applications might need to minimize false positives while others might be more forgiving with false positives in order to receive a greater true positive rate.

One of the goals of the classifier is that it is supposed to be able to correctly classify encrypted flows. To know if this expectation has been met, a large dataset containing encrypted flows with ground truth is needed. As no such dataset could be obtained, this could not be tested. However, if the characteristics of the encrypted video flow and the characteristics of unencrypted video flows, are very similar. Then, it would be reasonable to expect that the model will be able to do reasonable predictions.

6.2 Evaluation of Study

Overall this study has gone reasonable well. Especially when considering the lack of experience of using Python and the modules and also having no prior knowledge of machine learning. Some hardships were avoided because the supervisor had organized resources capable of handling large amounts of data prior to the start of the study. A starting point for research was also provided. This way, there wasn't a lot of wasted time setting things up before the actual work on the study began. A lot of time was spent converting the dataset into a feature vector. Mainly because of the inexperience but also because of finding things out about the dataset which had to be filtrated or handled in some different way. The feature creation process also suffered from performance issues, where creating the feature vector even on a relatively small dataset took a very long time. As such, quite a bit of time had to be spent trying to improve the performance. A different feature set was decided upon even after the middle point of the study. After the trouble of creating the feature vector, using the machine learning classifiers were relatively easy. It was easy to get results, and not too much time had to be spent on the machine learning part, only adding cross-validation and some small functions. Optimizations were carried out to try and get the best possible results for the classifiers, before the final test on the larger dataset.

During the study, continuous support from our Karlstad University supervisor was provided, at least meeting once every week. The meetings with our Procera supervisor was done at an approximately biweekly basis, where we reported on the progress made and to discuss the next phase of the study.

6.3 Future Work

There are a lot of work that can be done related to this study but in this section only the most immediate work will be listed. Some ideas for future work is:

- **Encrypted traffic**
  One of the problems with the dataset is that it has no ground truth for encrypted traffic. As such, no predictions can be done on encrypted traffic to evaluate the accuracy the classifiers have for such traffic. Preferable, training should be done on encrypted traffic to ensure that their characteristics can be correctly identified. The Procera DPI cannot

handle encrypted traffic. Hence, the need for the classifier to be able to correctly classify encrypted traffic.

- **Mobile network traffic**
  Another possible future endeavor could be to evaluate the classifiers on a different kind of network traffic, such as mobile network traffic. The classifiers also be trained on this data, or possibly create another model for this purpose.

- **Feature engineering**
  The features that were used in this study were selected mostly from which features previous studies has found successful. However, these studies did not specifically try to identify video flows. It might be a good idea, to dwell deeper into what separates video flows from other flows and try to find the perfect features for this purpose.

- **Feature selection**
  Instead of doing the feature engineering step above, there is also the option of keeping the current features but do more exhaustive testing to find a smaller feature set that improves or retains the same accuracy.

- **Only actual video flows**
  There is a suspicion that some of the flows with the ground truth set as video, is in fact not actual video traffic. If this is the case, that certainly can mess with the accuracy of the classifiers. Removing these fake video flows would help towards creating a more accurate model.

6.4 Concluding Remarks

Attempts were made to successfully classify video flows within an Internet traffic network. Based on the high accuracies gained, the attempt is considered successful. It illustrates the power of the machine learning field of study when it comes to classification problems. Since a greater percentage of Internet traffic is becoming encrypted, for companies relying on classification of traffic as their business model, being able to correctly classify encrypted flows is paramount.

This study can be seen as a small part of a bigger picture where a much greater amount of classes can be used for classification. This depends on the purpose of the classification. As the classification based on characteristics of the flow can never be absolute, it is important to consider the fault tolerance of the application. The model is also going to have to be recreated continuously as the characteristics of Internet traffic changes. This can happen slowly over time, or very quickly as a new application with different characteristics to similar applications gain popularity.

7 References

[1] "Procera," [Online]. Available: https://www.proceranetworks.com. [Accessed 4 February 2016].

[2] C. Fuchs, "Implications of Deep Packet Inspection (DPI) Internet Surveillance for Society," *The Privace & Security - Reseach Paper Series,* no. 1, pp. 1-125, 2012.

[3] Sandvine, "Global Internet Phenomena Spotlight - Encrypted Internet Traffic," 2015.

[4] R. Kohavi and F. Provost, "Glossary of Terms," *Machine Learning,* vol. 30, pp. 271-274, 1998.

[5] M. Cord, P. Cunningham and S. J. Delaney, "Supervised Learning," in *Machine Learning Techniques for Multimedia*, Springer-Verlag Berlin Heidelberg, 2008, pp. 21-49.

[6] R. Rojas, "Unsupervised Learning and Clustering Algorithms," in *Neural Networks,* Berlin, Springer-Verlag, 1996, pp. 101-123.

[7] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, Cambridge: The MIT Press, 2012.

[8] A. Smola and S. V. Vishwanathan, Introduction to Machine Learning, Cambridge: The Press Syndicate, 2008.

[9] A. Zheng, Evaluating Machine Learning Models, O'Reilly Media, Inc, 2015.

[10] Google, "Google News," [Online]. Available: https://news.google.com/. [Accessed 12 February 2016].

[11] Coursera, "Unsupervised Learning," [Online]. Available: https://www.coursera.org/learn/machine-learning/lecture/olRZo/unsupervised-learning. [Accessed 12 February 2016].

[12] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM,* vol. 55, no. 10, pp. 78-87, 2012.

[13] A. B. Downey, Think Bayes, O´Reilly Media, Inc, 2013.

[14] J. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, pp. 81-106, 1986.

[15] V. Lavrenko and N. Goddard, "Decision Tree 1: how it works," [Online]. Available: https://www.youtube.com/watch?v=eKD5gxPPeY0. [Accessed 9 March 2016].

[16] Y. Mansour, " Pessimistic decision tree pruning based on tree size," in *Proc. 14th International Conference on Machine Learning,* 1997.

[17] F. Melgani and L. Bruzzone, "Classification of Hyperspectral Remote Sensing Images With Support Vector Machines," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 42, no. 8, pp. 1778-1790, 2004.

[18] Statsoft, "Support Vector Machines," [Online]. Available: http://www.statsoft.com/textbook/support-vector-machines. [Accessed 12 March 2016].

[19] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, no. 2, pp. 123-140, 1996.

[20] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

[21] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, New York: Springer, 2009.

[22] L. Tang, P. Refaeilzadeh and H. Liu, "Cross-Validation," in *Encyclopedia of Database Systems*, Springer US, 2009, pp. 532-538.

[23] Statistical Learning, "Youtube," [Online]. Available: https://www.youtube.com/watch?v=S06JpVoNaA0. [Accessed 17 May 2016].

[24] Python, "Python," [Online]. Available: https://www.python.org/. [Accessed 13 March 2016].

[25] IPython, "The IPython Notebook," [Online]. Available: http://ipython.org/notebook.html. [Accessed 13 March 2016].

[26] Numpy, "Numpy," [Online]. Available: http://www.numpy.org/. [Accessed 13 March 2016].

[27] pandas, "Python Data Analysis Library," [Online]. Available: http://pandas.pydata.org/index.html. [Accessed 13 March 2016].

[28] scikit learn, "scikit learn," [Online]. Available: http://scikit-learn.org/stable/. [Accessed 1 May 2016].

[29] SNIC, "SNIC," [Online]. Available: http://www.snic.vr.se/about-snic. [Accessed 13 March 2016].

[30] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters,* vol. 27, p. 861–874, 2006.

[31] Wikipedia, "Precision and recall," [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall. [Accessed 3 March 2016].

[32] A. Moore and D. Zuey, "Internet traffic classification using bayesian analysis techniques," in *SIGMETRICS '05 Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, New York, 2005.

[33] N. Williams, S. Zander and G. Armitage, "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification," *ACM SIGCOMM Computer Communication Review,* vol. 36, no. 5, pp. 7-15, 2006.

[34] A. Dainotti, A. Pescapé and C. Sansone, "Early Classification of Network Traffic through Multi-classification," in *Traffic Monitoring and Analysis*, Springer Berlin Heidelberg, 2011, pp. 122-135.

[35] D. Aha, "Editorial on Lazy Learning," *Artificial Intelligence Review,* vol. 11, pp. 7-10, 1997.

[36] J. Fûrnkranz, D. Gamberger and N. Lavrac, "Rule Learning in a Nutshell," in *Foundations of Rule Learning*, Springer-Verlag Berlin Heidelberg, 2012, pp. 19-55.

[37] K. Gurney, An introduction to neural networks, London: UCL Press, 1997.

[38] P. Ioan, "An approach of the Naive Bayes classifier for the document classification," *General Mathematics,* vol. 14, no. 4, pp. 135-138, 2006.

[39] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman Publishers, Inc, 1993.

[40] Y. S. Huang and C. Y. Suen, "The behavior-knowledge space method for combination of multiple classifiers," in *Computer Vision and Pattern Recognition*, 1993.

[41] K.-D. Wernecke, "A Coupling Procedure for the Discrimination of Mixed Data," *Biometrics,* vol. 48, pp. 497-506, 1992.

[42] T. S. Tabatabaei, F. Karray and M. Kamel, "Early Internet Traffic Recognition Based on Machine Learning Methods," in *IEEE Canadian Conference on Electrical and Computer Engineering*, 2012.

[43] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory,* vol. 13, no. 1, pp. 21-27, 1967.

[44] R. Nossenson and S. Polacheck, "On-Line Flows Classification of Video Streaming Applications," in *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*, Cambridge, MA, 2015.

[45] L. Peng, B. Yang, Y. Chen and Z. Chen, "Effectiveness of Statistical Features for Early Stage Internet Traffic Identification," *Int J Parallel Prog,* vol. 44, pp. 181-197, 2016.

[46] M. Folk, G. Heber, Q. Koziol, E. Pourmal and D. Robinson, "An overview of the HDF5 technology suite and its applications," in *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, New York, 2011.

[47] S. Westlinder, *Internet Video Traffic Classification: A Machine Learning approach with Packet Based Features using Support Vector Machine,* Unpublished.