TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

Nurbanu Konayeva IVCM 177189

Application of Active Learning for Botnet Detection

Master's Thesis

Supervisor: Hayretdin Bahsi, Ph.D.

Sven Nõmm Ph.D.

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia Teaduskond
Tarkvarateaduse instituut

Nurbanu Konayeva IVCM 177189

Aktiivõppel baseeruv botnet rünnakute tuvastamine

Magistritöö

Juhendaja: Hayretdin Bahsi,
Ph.D.

Sven Nõmm
Ph.D.

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Nurbanu Konayeva

13.05.2019

Abstract

Network Intrusion Detection Systems face security challenges in detecting modern botnets. While different machine learning (ML) methods were extensively applied for botnet detection, the use of the human-machine interaction in detecting the botnets is still immature.

This paper analyzes how classification models can be adopted in active machine learning with the help of various query selection methods for botnet detection in network intrusion detection systems. Under pool-based sampling scenario, performance of each label query selection of obtaining the most informative label class were analyzed and compared.

Obtained results proved the effectiveness of the human-in-loop for labeling queried instances under the active learning approach in classifying the normal network traffic along with attack types (Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms). Results also revealed that the learning process based on the most informative few labeled samples perform better than fully labeled data pools under supervised learning. However, the expert's performance in providing inaccurate label class may be disturbed by changes in classification and detection accuracy.

This study emphasizes adopting human-in-loop interaction in acquiring labeled instances which improve the learning of classification models, as well as the impact on the detection of botnet when inaccurate label class was provided.

This thesis is written in English and is 80 pages long, including 6 chapters, 35 figures and 20 tables.

Table of contents

1 Introduction 10
2 Background 14
2.1 What is the Botnet 14
2.2 Machine Learning 17
2.2.1 Classification Algorithms 19
2.3 Relevant research 20
2.3.1 Similar research from other fields 22
2.3.2 Traditional Machine learning models 24
3 Methodology 27
3.1 Stage 1. Data acquisition 27
3.1.1 Dataset 27
3.1.2 Extracted Features 29
3.2 Stage 2. Data Pre-processing 35
3.2.1 Feature selection 36
3.2.2 Numerical feature's selection 38
3.3 Stage 3. Classification, Training 41
3.3.1 Decision Tree algorithms 43
3.3.2 Random Forest 45
3.3.3 XGBOOST 46
3.4 Stage 4. Performance Evaluation Metrics 48
3.4.1 Correct and Incorrect Classification 48
3.4.2 Classification Metrics and Performance Evaluation 49
3.4.3 Selected Performance Metrics 51
4 Botnet Detection: Practical Implementation 52
4.1 Selected numerical features for botnet detection 53
4.2 Classification algorithms 55
4.3 Unsupervised Learning 60
4.3.1 DBSCAN Clustering 60
4.4 Semi-supervised Learning 63

4.5 Active learning 64
4.5.1 Active Learning Scenarios 65
4.5.2 Query Strategy 67
5 Botnet Detection: Model Implementation Analysis 69
5.1 Semi-supervised algorithm results for botnet detection 69
5.2 Active learning botnet detection results 73
5.2.1 Queried Labels are True 74
5.2.2 Queried Labels are Wrong 82
6 Conclusion 87
References 89
Appendix 1 – Classification performance comparison for different set of features 93
Appendix 2 – Classification accuracy with top ranked features 94
Appendix 3 – Scatter Plot with three features 95
Appendix 4 – DBSCAN results 101
Appendix 5 – Semi-Supervised Pseudo-Labeling Results 103
Appendix 6 – Active learning performance results when queried label is wrong 107
Appendix 7– Active Learning Evaluation Performance Results 123

List of figures

Figure 1. Client server mode 15
Figure 2. Peer-to-Peer Model 15
Figure 3. Experiment results (KDD 99 Dataset) of using a) the active learning method; b) random selection method [24] 23
Figure 4. Experiment results (AWID Dataset) of using a) the active learning method; b) random selection method [24] 23
Figure 5. Proposed methodology scheme [29] 25
Figure 6. Configuration setup for obtaining the data [5] 29
Figure 7. Feature extraction with the Argus and Bro-IDS Tools [5] 30
Figure 8. Example of simple Decision Tree based on the botnet and normal network traffic 44
Figure 9. Implementation Architecture 52
Figure 10. Accuracy performance results for both binary and multi class 53
Figure 11. Scenario 2. Original data labels 58
Figure 12. Scenario 2. XGB prediction results 58
Figure 13. Scenario 3. Original data labels 59
Figure 14. Scenario 3. XGB prediction results 59
Figure 15. DBSCAN clustering results for Scenario 3 62
Figure 16. Membership Query Method 65
Figure 17. Stream-Based Selection Method 65
Figure 18. Pool-based sampling scenario 66
Figure 19. Pseudo Labelling Technique [62] 70
Figure 20. Semi-supervised learning results for binary label class 72
Figure 21. Semi-supervised learning results for multi label class 72
Figure 22. Binary class: Active learning performance results for Decision Tree 75
Figure 23. Binary class: Active learning performance results for Random Forest 76
Figure 24. Binary class: Active learning performance results for XGBoost 76
Figure 25. Multi class: Active learning performance results for Decision Tree 77
Figure 26. Multi class: Active learning performance results for Random Forest 77

Figure 27. Multi class: Active learning performance results for XGBoost..... 78
Figure 28. Binary class: Active learning performance results for Random Forest with Margin Sample Selection..... 79
Figure 29. Binary class: Active learning performance results for XGBoost with Margin Sample Selection 80
Figure 30. Binary class: Active learning performance results for Random Forest with Entropy Selection 80
Figure 31. Binary class: Active learning performance results for XGBoost with Entropy Selection 81
Figure 32. Multi class: Active learning performance results for Random Forest with Margin Sample Selection..... 81
Figure 33. Multi class: Active learning performance results for Random Forest with Margin Sample Selection..... 82
Figure 34. Binary Class: Active learning accuracy results for Random Forest with Entropy Selection k=10 84
Figure 35. Multi Class: Active learning accuracy results for XGBoost with Margin Sampling Selection k=10..... 85

List of tables

Table 1. Data set Statistics [38] 30
Table 2. FLOW FEATURES [38] 30
Table 3. BASIC FEATURES [38] 31
Table 4. CONTENT FEATURES [38] 31
Table 5. TIME FEATURES [38] 31
Table 6. ADDITIONAL GENERATED FEATURES [38] 32
Table 7. LABELLED FEATURES [38] 33
Table 8. DATA SET RECORD DISTRIBUTION [38] 33
Table 9. Distribution of the data instances among the label class 35
Table 10. The Fisher's score results 39
Table 11. Matching top features for both scenarios (binary, multi) 40
Table 12. Confusion Matrix 48
Table 13. Accuracy metrics of the model trained with top numerical features 54
Table 14. Performance results for different models 56
Table 15. Performance results of the modes according to the given scenarios 57
Table 16. DBSCAN results for given scenarios 62
Table 17. Accuracy results for binary class 71
Table 18. Accuracy results for multi class 71
Table 19. Binary class: Supervised learning results for accurate and wrong labels 83
Table 20. Multi class: Supervised learning results for right label and wrong label 83

1 Introduction

A *botnet* is a collection of compromised network devices that are infected and controlled remotely by a common server. Apart from functioning as an email spamming tool, botnets can penetrate and intrude into an organizational network to steal valuable assets such as financial data, intellectual property as well as stealing a considerable amount of money and perform a massively coordinated cyber-attack. Bots are capable of overloading the target network with requests and coordinating malicious attacks such as Distributed Denial of Service (DDOS) for personal and political motives. According to the survey [1] around 16% of computers connected to the internet are compromised either by active or passive bots. Bitdefender security insights reported that the power of the botnet attack increased in 2018 relatively to the previous year. As stated in the report [2], the most powerful giant botnets formed by cybercriminals are named as Mirai and Satori botnets.

Modern botnets are becoming more vaguer, representing new joint features evolving unknown behavioral class. Recent botnets, continuously randomize the port numbers and domain names, making detecting botnet more challenging for Network-based Intrusion Detection Systems [3]. By randomly delaying transmitting traffic, botnets try to behave like "normal" sessions, which make signature-based methods [4] give more false negative results.

With the use of machine learning, many tasks, like detection of cyber-attacks performed by botnets, can be automated and deployed into security tools. Without being explicitly programmed, machine learning algorithms are used to analyze data and predict the outcome occurrence based on received specified samples.

A branch of artificial intelligence, machine learning aims a constructive study of the systems with the ability to recognize different patterns and predict qualified preference based on the given data. In recent decades more researches integrate machine learning into botnet detection studies and provide various experimental approaches by using generalized knowledge derived from systematic experiences of detection systems, to propose previously unseen computational methods.

Supervised learning algorithm use the data where the correct class is revealed (samples are labeled). Unsupervised learning algorithm does not have any labels attached to supervise the learning (samples are unlabeled). In semi-supervised learning, two different algorithms used, starting with the labeled examples, and then by telling other samples the way they think about unlabeled data. The difference between semi-supervised algorithm from active learning is that, in active learning, the algorithm itself decides which labels (usually most informative ones) human should label. Active learning makes the classification of data learning from other samples, but in additional confirms the decision by querying the user.

Unlabeled data is relatively easy to acquire although expensive to label and generally corporations have limited resources to label all of their data. In such cases AL can provide similar and even better results than fully supervised algorithms with less cost and time spent to acquire label class for the data instances. Most of the studies point out that labeling is pricey and try to implement unsupervised methods. However, many organizations have started to have a Security Operations Center (SOC) capabilities that may help label some amount of the data (real case scenarios with few labeled instances). Although SOC's have labeling capability, they cannot mark all data. But smart usage of limited labeling capability can improve the performance of learning models by applying human-machine interaction which is the crucial concept of active learning.

For this reason, AL was chosen as it can provide aimed results by carefully increasing the size of selected labeled data. Moreover, in real life scenarios, when DDOS attacks are performed by the new botnets, which represent an unknown behavioral class, active machine learning algorithm model will fit the best to analyze the data and interact with the expert to detect the anomaly and classify the external network traffic.

Under Active Learning (AL) method it was aimed to train the model using network packets created by the IXIA Perfect Storm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [5] to predict contemporary botnet attack behavior. This dataset has nine types of attacks, which are Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. All the records include different type of network data labeled as an attack and normal, as well as different attack vectors mentioned above.

Similar studies for botnet detection were approached by using supervised [6] [7] [8], unsupervised [9] and semi-supervised [10] [11] [12] training models. Studies had a different focus on feature selection and applied the algorithms comparing the obtained accuracy to analyze performance. In addition to this, as an example research study [10] applies active learning approach on regression model, with the help of "most likely unknown" (MLU) sampling method to pick the unlabeled sample that has the highest probability of belonging to the unknown class. Very limited feature set (8 features) was used and only three attack vectors are predicted based on the CSET'11 (prepared in conjunction with 20th USENIX Security Symposium in 2011) dataset.

There are several differences with the study [10] regarding the application of active machine learning in the current work referred to as:

- Use of classification models for the active machine learning approach;
- Use of the broader set of the features (39 features – time, flow, connection, basic, additional) based on the network data;
- Improved (regarding the problems referred in many other datasets for network intrusion detection systems) dataset with nine attack types;
- Different methods for sample selection and their comparison;
- Separate scenarios for acquiring accurate label and wrong label (ratio of mistakes between 10% and 50%) by complimenting it with the cost of the prediction accuracy.

The key point of this study application of AL methods to learn from most informative data points queried from a small pool (try to match with real case scenarios) of labeled instances and train the algorithm.

This paper attempts to answer the following questions:

1. Is it possible to adopt classification models for active machine learning in botnet detection?
2. How various sample selection methods for acquiring label class of each instance influences on the learning process of the classification model?
3. How human-machine interaction (labeling the queried instance) with the help of active machine learning improve the overall prediction of the botnet compared to supervised learning?

The formulation of the present research problem statement is resumed in the following points:

- Perform feature selection for botnet detection based on the network data.
- Identifying most appropriate models to work with the active learning approach.
- Querying the most informative data points with the help of several methods for expert labeling.
- Learn from obtained samples and analyze the variability of selected predictions in the binary and multi-class labeled dataset.

Several scenarios are performed to analyze the outcome where it is assumed that:

a) security expert/oracle provides an accurate label for the most informative queried samples.
b) security expert/oracle provides inaccurate labeled instances with error rate of 10%, 20%, 30%, 40% and 50% for the most informative samples queried from the pool.

The process repeated several times, including XGBoost, Decision Tree and Random Forest models, until the desired accuracy on botnet prediction is achieved.

The present document is designed in the following stages:

- **Chapter 2:** devoted to the background information and related literature;
- **Chapter 3:** deals with current research methodology;
- **Chapter 4:** presents a practical implementation of this thesis for the machine learning process;
- **Chapter 5:** shows key results and analysis of the leading experimental tests;
- **Chapter 6:** discusses the outcomes that are concluded from this study.

2 Background

2.1 What is the Botnet

Initially, botnets were created to serve as a legitimate tool with a specific functionality over the Internet relay chats (IRC). Later on, when different vulnerabilities of the IRC networks discovered, botnets were developed to become a hacker's tool to perform various malicious activities, gain valuable data and information for different purposes.

Currently, botnets hold the meaning of compromised computers connected and coordinated under the control of malicious actor. Those connected computers perform numerous tasks and launch attacks. Most common areas for botnet usage are an organized crime to conduct illegal activities online like:

- email spamming
- denial of service attacks (DoS and DDoS attacks)
- transmitting malware/adware/spyware
- stealing informative sensitive data like user logins and financial reports
- phishing attacks, etc.

The term “botnet” is a simple combination of words “bot” and “net.” The bot is holding the meaning of robot, which represents the infected by malware computer and net shortened from the network, which represents the group of linked systems over the internet. EarthLink Inc. was the first company who named the biggest spam network as a botnet during the lawsuit against Khan C. Smith in 2002 [13]. EarthLink was the company which processed around 10 billion emails per year, with the cost ratio 1$ per 1000 emails. According to the article, the situation was around a man named Tennessee who, used stolen credit card numbers and passwords of EarthLink users to create as many as 1,000 accounts used to send unwelcome emails. According to that time, the man earned 3 million dollars by running the most significant spam. However, he had to pay 25 million dollars back to the company due to the lawsuit court decision. This case becomes an excellent example of how dangerous and harmful botnet technology can be.

Mainly there are two primary models for setting up the botnets:

- Client Server mode

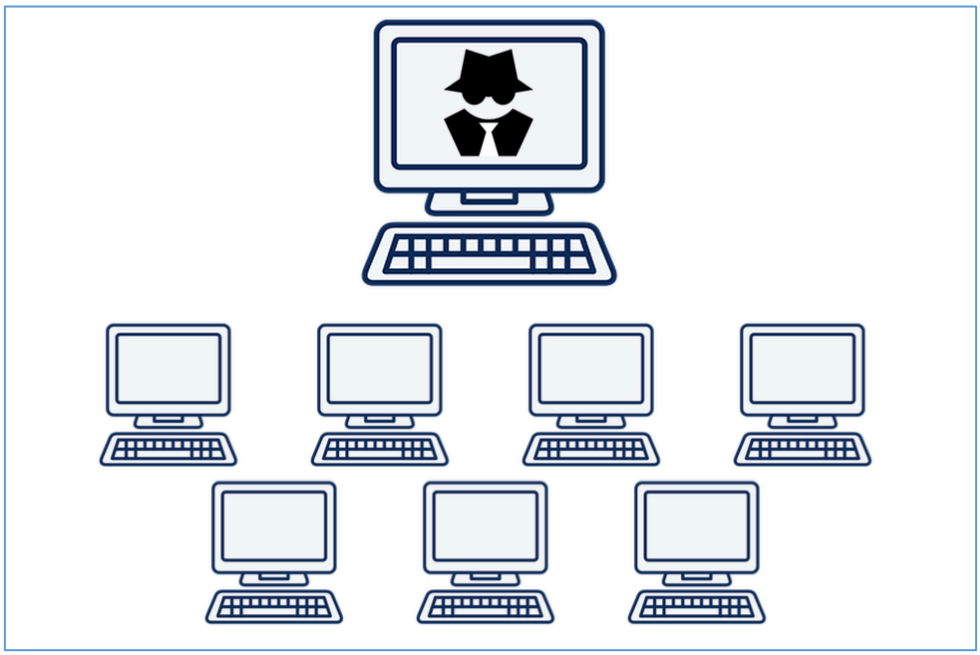


Figure 1. Client server mode

- Peer-to-Peer model

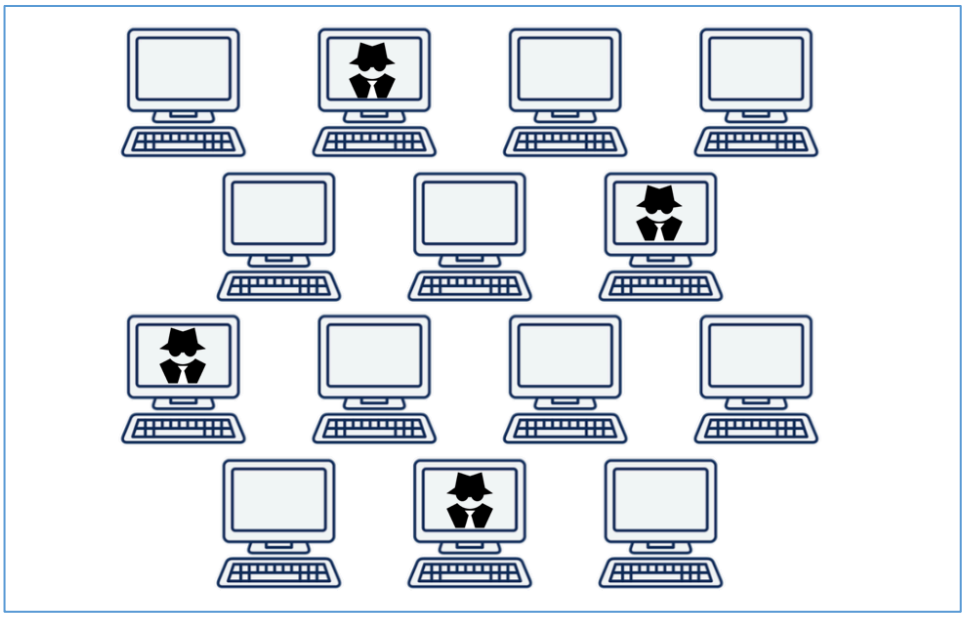


Figure 2. Peer-to-Peer Model

The Client-Server model or Command and Control (C&C) is a very straight forward method of controlling and sending instructions from one single location to other infected

compromised devices. The central server is usually under the attacker's physical control and can stay active for a couple of days, but often, the lifetime of the shelf is quite limited. It was said aa a powerful model, since hackers usually choose a legitimate server, which it is harder to detect for law enforcement. However, recent studies show that botnet detection systems based on C&C are highly positive to give false negative results, due to the evolution of the botnets thought the continuous randomization of the port numbers and domain names.

Since there are many solutions grounded on this model, recently, hackers prefer Peer-to-Peer model over the C&C for the botnet construction. Another reason for this is that Peer-to-Peer model meant to fix weaknesses of the C&C model by connecting and communicating directly from one infected machine to few others, and they are in their turns continue to do the same. This loop reaches until the whole puzzle is complete and the system is ready to give it a shot. Additionally, even if some of the devices will be removed or shut down, it will not cause problems, since other devices can replace them and pick up the chain fast.

According to “Heimdal” security blog [14] botnets inherited from the Zeus botnet family were very popular and harmful before 2016 in context of stealing personal data and perform massive unauthorized money transfers with the use of bank account details. This case also shows that botnets did evade; they upgrade over time with new joint features to become less evident for the detection systems.

Another example of the botnet with the successful performance of shutting down the Twitter, CNN, Spotify and many other servers and services in September 2016 was the botnet named as Mirai, which did the biggest in history DDoS attack [15]. Attack was performed with the usage of the compromised Internet of Things (IoT) devices. However, even after being discovered it did not stop hackers from creating new versions of the Mirai botnet and making another shot in July 2018, which caused many security discussions among security professionals and researches. The expansion of the Mirai botnet into different types proves that different techniques are applied to behave like regular network traffic, which indeed makes the botnet detection more challenging for different Network-based Intrusion detection and prevention (IDS and IPS) systems [16].

All of this raised the interest of the researchers to try and investigate new methods and models to capture botnets and prevent setup from being compromised. Due to the ability to automate the tasks and working with a large amount of data, recent academic literature highly focused in applying machine learning approaches to maximize the effect of the prediction and accuracy of the models. Various experimental methods by using generalized knowledge derived from systematic experiences of detection systems were studied, to propose previously unseen computational approaches.

2.2 Machine Learning

Google's self-driving cars and robots get a lot of press, but the company's real future is in machine learning, the technology that enables computers to get smarter and more personal [17].

– Eric Schmidt (Google Chairman)

Machine learning (ML) is said to be a part of Artificial Intelligence (AI) filed. With the use of computational methods, ML technique learns information directly from the data without relying on predestined rule-based equations to build algorithms used to predict an output while updating newly available data. It is a natural process that teaches computers to learn from experience to help people process a large amount of data within a short period. The universal principle of predicting the data is to map learning function (f) with the input samples (X) and process an output variable (Y): Y=f(X).

Nowadays many people are familiar with machine learning techniques from the internet, personalized adjusted ads processed from their purchases and actions of the same patterns. It happens because ML algorithms learn data from similar repeated model and behavior in real time and recommend the output based on experience. Apart from online marketing and personalized ads, machine learning widely used for spam filtering, fraud control systems, network security detection systems, and other maintenance, monitoring, and structuring news feed. Those examples are just a tiny part where ML algorithms used in today's practice. There are different variations of Machine Learning Algorithms which can be used and applied depending on the precise needs for the aimed process:

a) *Supervised Learning*: This algorithm establishes a model and gives an output based on provided evidence. A supervised learning algorithm consists of a known

set of inputs with the predictors (knowm set of response) to the output data to train the model and provide a rational response to a new data. With the help of those set of independent variables, the function maps the inputs to desired outputs. With the determined variables and features, the model continues the training process until the desired accuracy level achieved. To develop predictive models Supervised learning techniques uses either regression or classification techniques.

- *Regression* algorithms are used to make predictions regarding numerical entities. Examples are usually pointing to predict the pricing of different variables like products, house, and other goods.

- *Classification* algorithms are used to create a diverse membership for a specific known class type. Email filtering (spam, not-spam) and medical diagnosis (identifying the diseases based on symptoms) are the examples for this type of algorithm.

b) *Unsupervised Learning*: This algorithm does not require any target desired outcome data to train the model. Also known as neural network it approaches more complex processing tasks to cluster training data into different groups by correlating between many input variables. This algorithm widely used in image recognition, face recognition, bank associations and require a significant amount of data for training purpose.

c) *Semi-Supervised Learning*: Combination of *Supervised* and *Unsupervised* machine learning provides this type of algorithm. Usually used when there is not enough labeled data to train an accurate model. Training approach can start with the labeled resources, and unsupervised machine learning algorithms will continue process learning from the outcome. Detection systems can detect well-known fraud and anomaly, and rest can slip without being known and remain unlabeled, which is an excellent example of this type of algorithm.

d) *Active Learning*: Similar approach to *Semi-Supervised* learning with slight modifications. Instead of learning from instances automatically and predicting an output, Active Learning predicts an output by selecting an unlabeled data and querying each iteration to Oracle or human expert who analyses and determines the label of the instance. Labeled data instance can be predefined lower than in

Semi-Supervised learning and still provide a high level of accuracy of the prediction model.

e) *Reinforcement Learning*: With the use of this algorithm machine is trained to iterate an action in a dynamic environment to return specific decision based on trial and error method. The machine learns from every iteration to produce a favorable outcome based on previous experience to provide an optimal and accurate capture. As a result, we get a well-known process for online games with human and computer interaction.

2.2.1 Classification Algorithms

In this thesis, the main idea to obtain proper botnet detection by applying active learning algorithms with the help of classical classification algorithms.

The main idea for machine learning classification algorithms is to categorize given data into desired and distinct numbers of classes. Boundaries conditions for each class determining the assigned target label class for every different subset. Classification can be binary and multi-class. Binary classifier gives an outcome with only two particular levels like normal and anomaly. A multi-class classifier can have more than two practical classes and predict a result for a different type of specific categories. In the example of this thesis multi-class classifier used to distinguish between types of attacks within the anomaly behavior.

Even though the primary purpose is common for all classification algorithms, various mathematical and logical approaches are different for them to deal with the specific problem. Well-known and widely used classification algorithms with the brief explanation are listed below:

- ***Decision trees*** classify the data into a tree structure by breaking down into smaller subsets. Thus, achieved with the help of consecutive rules based on the most significant differentiators in the input variables.

- ***K-nearest neighbor*** classifies an object by the majority vote weight of the closest neighbors. The targeted object assigned to the most convenient and familiar class among its nearest k neighbors.

- ***Random forests*** classify the objects by constructing multiple decision trees and attaching it to the class with the most votes from all the trees.

- ***Support Vector Machines*** classification algorithm plots training data points in n-dimensional space (depending on the features set number) with a clear gap between them. New examples predicted according to the nearest category where they fall in the map.

- ***Naive Bayes*** classifier inspired by the Bayes theorem. The probabilistic classifier uses works under a simple assumption where attributes are conditionally independent.

- ***Logistic Regression*** classification is a statistical method that performs binary classification, where label outputs are binary. An output determined by the analysis of the dataset by defining one or more independent variables.

- A ***Neural Network*** consists of units of layers or components with direct connections among them. Neural networks are needs enormous computational complexity but could be applied to many different tasks. This algorithm provides a good result if the job requires to work with images.

2.3 Relevant research

Due to evasion of the botnet generation Zhicong Qiu, David J. Miller and George Kesidis proposed semi-supervised active learning algorithms [11] to detect unknown anomaly botnet behavior. Detection of the botnet based on information taken from the sequence of packet sizes in a specifically given flow. Experimental setup mainly used three different PCAP files taken from:

a) LBNL traces and used as a regular (normal) traffic [18], which was collected by monitoring medium-size enterprise network for more than 100 hours covering 22 primary subnets with different protocols mentioned in data pre-processing step of the same research paper, such as TCP and three-way handshake.

b) Zeus PCAP files taken from VRT Zeus [19] and ISOT Zeus [20]. Zeus bots were incorporated to the dataset, due to its well-known detection evasion techniques,

like using random ports and proxy server, which makes traditional methods challenging to detect. Authors mentioned that Zeus variations are trendy and commonly used for the botnet applications, especially the ones used for cybercrime activities.

After obtaining three different PCAP files with normal and anomaly traffic, authors had datasets with the botnet having both C&C and non-C&C traffic for the training and testing purpose. Total flows of the standard (normal) and botnet flow are as follows, LBNL [18] 9972 number of streams, VRT Zeus [19] 64, and ISOT Zeus [20] 23 flows respectively.

Authors used 1/3 of the dataset with randomly (positive and negative) labeled samples to train Bayesian Network, remaining dataset split into half, as unlabeled regular (normal) traffic and unlabeled botnet for active learning approach. For testing the anomaly detection, they used the ratio of web flows, falsely identified as a botnet and the rate of a botnet, classified correctly as a botnet. The comprehensive trade-off between two metrics was combined to visualize ROC AUC curves to see the number of active labeling of the algorithm. For AL approach they moved forward with MLU (most likely unknown) sampling to pick the unlabeled sample with the highest priority of belonging to the specific class, According to the discussion, their experimental setup produced a highly effective solution which compared with similar studies, [21] [22] [23] and proposed feature representation where most ineffective working systems could replace. The accuracy of the computation reached about 88% in ROC AUC performance and about 90% for the supervised learning models. Also, they mentioned that some of the related works were ignoring the qualitative aspect of the traffic data used in the training and testing stages, which usually cause the poor performance of the methods used.

There are many differences studied and applied in this thesis compared to the provided literature [11] review. The novelty of this study is the use of the different active learning methods, where sampling of the unlabeled instances achieved with the help of three different ways, namely Random Sampling, Margin Sampling Selection, and Entropy Selection, whereas literature [11] provided only one sampling method results based on MLU. Also, after initial training and testing of the dataset, three classification models are selected to be trained with the active learning scenario, which is XGBoost, Decision Tree and Random Forest, while in the literature [11] Bayesian Network used as the central model of the study. Nevertheless, the performance results for detection achieved as 95%

accuracy while having fewer data points, which is comparatively better from the literature [11] . The remarkable difference in this thesis with the explained research is the nine types of attacks studied and classified in the different active learning scenarios and sampling methods. Another variation of this thesis, the experimental setup, providing the performance results and cost of acquiring incorrect label class from the Oracle/Expert, whereas literature [11] experimental setup includes only one scenario with the assumption of acquiring all right labels.

All the obtained results are compared and analyzed in section 5, and detailed information explained in the further parts of this thesis.

2.3.1 Similar research from other fields

Although there can be found many literature resources on ML techniques applied for botnet detection, only limited amount of them genuinely dedicated to the active learning approach. Similar reperch gap identified by Kai Yang, Jie Ren, Yanqiao Zhu, and Weiyi Zhang and published as "Active Learning for Wireless IoT Intrusion Detection" in December 2018, [24]. According to the content of the article, authors underline the similar problem of insufficient labeled training data, whereas AL a subfield of ML solves the problem by using a limited number of labeled samples. By querying the user/expert under the specific strategy, algorithms aim to receive new labeled data and continue the process of training until desired results obtained. Thus, reduces the cost and time of getting clean and labeled data.

Experimental setup starts with the unsupervised model training to obtain the anomaly samples in the dataset. Then iteratively active learning approach was applied to reach the threshold of the performance where recall and precision used. The active learning applied in three steps:

a) Supervised learning
b) Label selection
c) Labelling by the expert

For the algorithm classification, XGBoost *distributed gradient boosting library which implements machine learning algorithms under the Gradient Boosting framework* [25]. XGBoost can solve many data science problems fast and accurate while having fewer

parameters with a more straightforward structure. Authors found this algorithm appropriate for the IoT resources due to scalable, portable and distributed characteristics.

After applying and comparing the result in Figure 3 [24] and Figure 4 [24] obtained from supervised algorithms and AL with the usage of two different datasets, authors conclude that AL method can improve the performance of traditional supervised learning and optimize cost and time for computation.

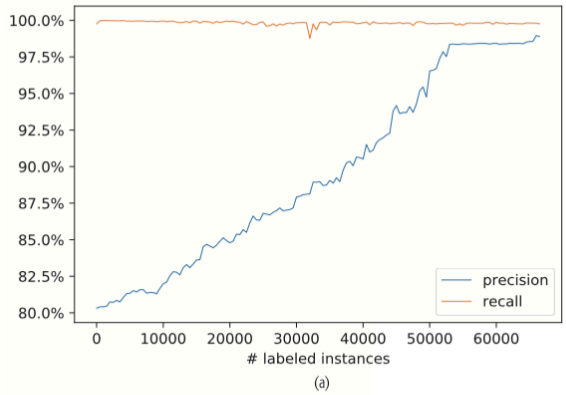


Figure 3. Experiment results (KDD 99 Dataset) of using a) the active learning method; b) random selection method [24]

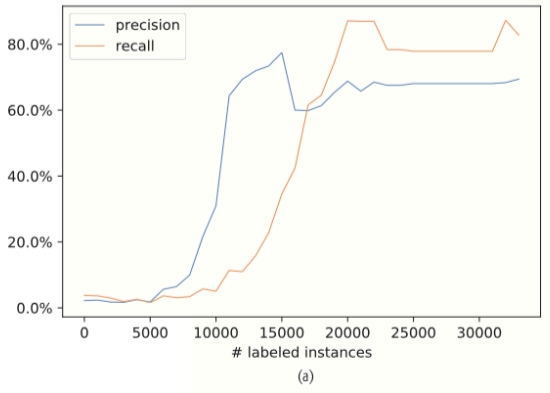


Figure 4. Experiment results (AWID Dataset) of using a) the active learning method; b) random selection method [24]

Figure 3 and *Figure 4* taken from [24] describes the comparison of results obtained from supervised ML and AL application. Precision and Recall performance metrics were calculated, and different datasets were analyzed.

The similar idea of utilizing AL algorithm was performed in "Active learning for semi-supervised structural health monitoring" research paper [26] by L. Bull ∗, K. Worden, G. Manson, N. Dervilis. Authors noted the simplicity in getting unlabeled data due to the digitalization and wide range of technology used in health monitoring. Complexity of

obtaining label class in a wide range of data, solved by AL where limited amount of labeled data is used. Authors worked on the project using MATLAB, and successfully obtained active learning experiment with the help of cluster-adaptive heuristic label propagation, where the process enabled by the heuristic hierarchal framework. This project supported by the UK Engineering and Physical Sciences Research Council (EPSRC).

2.3.2 Traditional Machine learning models

Nevertheless, there are a smaller number of studies with active learning application in botnet detection, many traditional supervised, unsupervised and currently being explored semi-supervised machine learning algorithms are used in this field.

One of the literature works, written by Vaibhav Nivargi, Mayukh Bhaowal, and Teddy Lee, studied botnet detection [6] by comparing different machine learning models such as Naïve Bayers, k-NN classifier, Decision tree, and others. They decided to choose two different methodologies for their experimental setup. The first methodology based on Binary Detection focusing on binary profiling and hex dumps as feature selection, and second was IRC log-based detection using a public communication channel to get the traffic. Those methods approached separately and in combination to analyze and distinguish between the outcome. One of the datasets used for Binary detection included a large number of executables, labeled as botnets, which was taken from Computer science department of John Hopkins University [27]. The second dataset with the labeled IRC logs acquired from Computer science department of Northwestern University, where dataset collected for the wireless overlay network architectures and botnet detection research [28]. Users extracted around more than a million features and selected only most informative 10,000 features obtained with the usage of chi-square selection based on the highest chi-square scores. The results obtained from the experiment evaluated using accuracy and F1 to observe the usage of the classifiers based on a specific focus on features. Both models gave similar accuracy results approaching almost 99%. From the results, it was discussed that the more extensive datasets and feature dependency correlate the performance for specific model algorithms.

A paper [29] by Gagandeep Kaur proposed semi-supervised learning algorithm under the description of “A Novel Distributed Machine Learning Framework for Semi-Supervised Detection of Botnet Attacks.” Author scheme a plan for generating a labeled dataset with

the usage of distributed KMeans clustering applied in Distributed Decision Tree base algorithm for botnet detection shown in Figure 5 [29]. For the experimental setup ISCX dataset [30] consisting of separate training and testing pcap files with the size 2,119,199 KB and 5,141,869 KB respectively. Since the main focus of the work was to detect more botnet-based attacks, data pre-processing was applied to sanitize a large amount of data and remove unnecessary traffic flow. Author converted pcap file into CVS (comma separated value) for training and testing and left nine main features namely Source Number, time, Source IP Address, Destination IP Address, Source Port Number, Destination Port Number, Protocol Type, Data Length in bytes, Info. The network traffic or any dataset usually obtained without any labeling, and very costly to do it manually, author proposed model of unsupervised KMean learning approach of clustering instances lying closest to the normal or anomaly traffic, and label samples to train the obtained sample with the help of Decision Tree models.

Performance of the model evaluated by precision, recall F1 score and confusion matrix, whereas accuracy varied from 84% to 88% and False positive rate for Gini and entropy is 1,4% and 1,3% respectively. “Gini Index” and “entropy” used in the context of measuring the impurity of the decision tree model. Gini Index (G) measures a nodes impurity whereas entropy is a measure of dispersion.

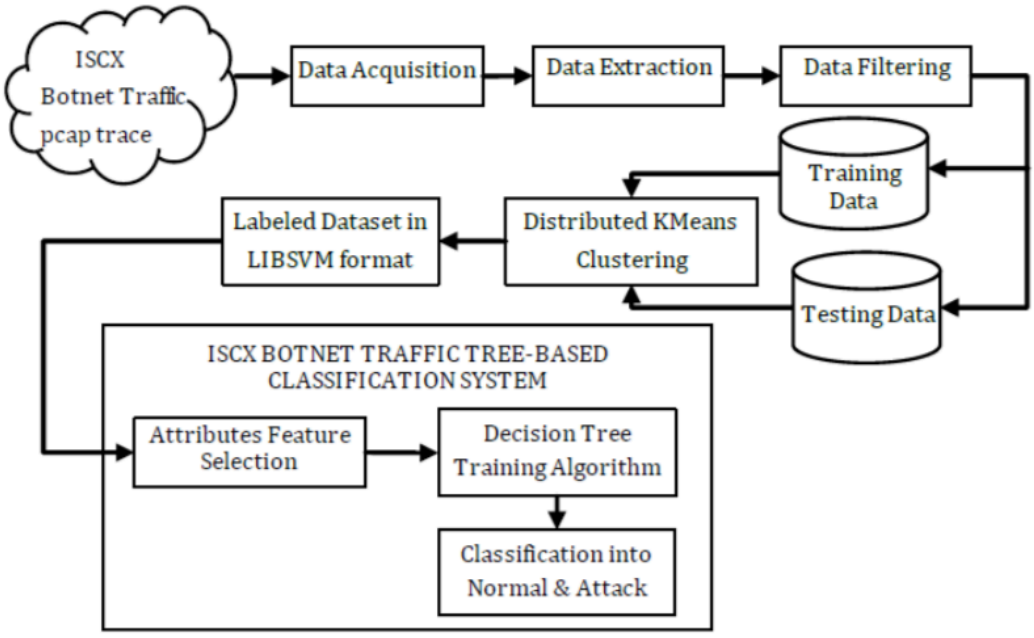


Figure 5. Proposed methodology scheme [29]

Current research focuses on the application of active learning algorithm for botnet detection. After precise selection three models assigned to take part in the active learning algorithm application scenario. The primary use of the learning process achieved with the help of the sample selection methods like Random Selection, Margin Sampling Selection, and Entropy Selection to identify most informative data points regarding the label class and use them to train the model and get the performance results. Along with the accurate label class queried from Expert, wrong labeling and cost of the calculations also included in the experimental setup of this thesis.

3 Methodology

The strategy of this master thesis designed in 4 main stages that are:

1. Data acquisition
2. Data pre-processing
3. Data processing (Training, Classification)
4. Model performance validation

Summarily, data acquisition stage involves the explanation of the collection and gathering features and instances converting and preparing them with appropriate standards and format to be used. Data pre-processing stage explains the filtering and ranking features by using appropriate feature selection methods, to identify most discriminative ones depending on the label class, while eliminating the redundancy and avoiding overfitting. And finally, training and testing stages where various algorithms were trained and validated with the given input to detect the anomaly behavior and botnet attack type.

3.1 Stage 1. Data acquisition

3.1.1 Dataset

At this stage UNSW-NB15: A Comprehensive Data set for Network Intrusion Detection systems was used [5]. This dataset created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). The unavailability of the broad network-based data set that would include various low footprint intrusions and practical information, reflecting contemporary network traffic scenarios was one of the main reasons for generating this dataset. Most of the previous researches utilized KDD98 [31], KDDCUP99 [32], and NSL-KDD [33] data sets which were prepared decades ago and currently outdated in respective of the network traffic and network attacks, that has been evaded from intrusion detection systems recently. Hybrid solution of the real normal traffic data and synthetic attack vectors used in the creation of the data set samples. Authors mentioned that existing and some novel methods used in generating the features for the research purpose.

The quality of the currently existing NIDS (network intrusion detection systems) data sets was composed and evaluated by the two essential characteristics that are the normal range of the traffic and latest thread inclusion. Some datasets match those characteristics namely KDDCUP99 and improved version of the same dataset NSL-KDD.

However, many IDS researchers who used KDDCUP99 dataset in their projects [34] [35] [36] [37] noted some essential disadvantages that affect the evaluation of the models performed for the IDS. One of the problems states that training and testing sets have different records, where training set found to be having an enormous amount of redundant data. Thus, bring to the bias detection due to the numerous records. Despite the fact of having different attack vectors which expected from those models in the intrusion detection, training and testing sets had unbalanced records among malicious traffic with missing values and missing attack types in the testing set. Another one refers to attack data packets TTL (time to live) value given as 126 or 253, whereas those values are mostly 127 and 254 respectively. However, TTL values 126 and 253 do not occur in the training records of the attack. And the main one that it doesn't contain evaded new attack samples representing new joint features, reported as low footprint attacks. Even though, creators of the KDDCUP99 released upgraded and improved version of dataset, where duplicates are removed from training and testing sets, new NSLKDD data set was still missing the major drawback where new comprehensive modern attack scenarios were not considered. For those reasons the UNSW-NB15 data set chosen since it was created to improve major drawbacks mentioned above.

To get the hybrid combination of the real modern normal and anomaly network traffic the authors of the UNSW-NB15 dataset applied IXIA PerfectStorm tool3. The anomaly traffic contains nine different families of the attack's types namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Information about attacks taken from the publicly available information security and vulnerability dictionaries that have continuously updated. Tcpdump5 tool was utilized to capture the network traffic. The entire period of the simulation took 31 hours with a size of 100 GBs. To separate each pcap file into 1000 MB files tcpdump tool used. And finally, to obtain 49 features and class label from the pcap files, the Argus and Bro-IDS are utilized along with the twelve C# language algorithms that have developed and applied for the in-depth analysis of the flow packets. All samples labeled from the ground truth table which contains all the simulated attacks types.

Full configuration setup of obtaining the data presented in Figure 6 taken from [5] where Tool IXIA is utilized. The final and the total number of the records collected for the training set is 175,341 and 82,332 records for the testing set.

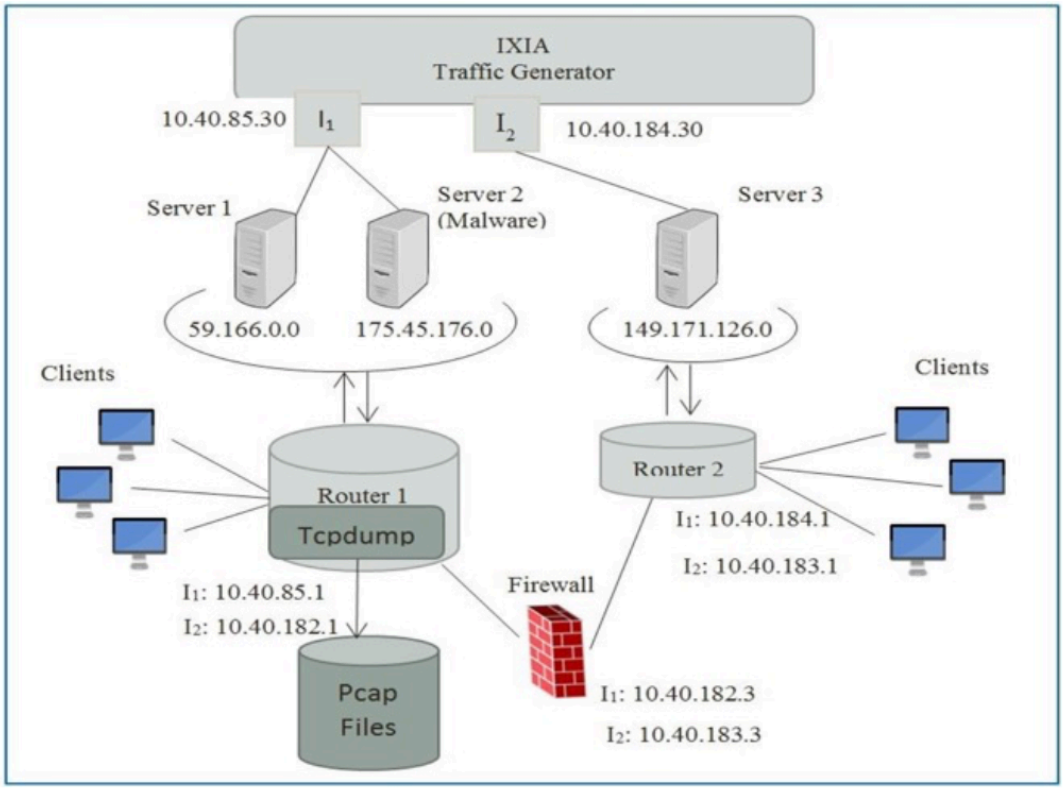


Figure 6. Configuration setup for obtaining the data [5]

3.1.2 Extracted Features

The architecture of obtaining the CVS data set files from pcap files for UNSW-NB15 training and testing dataset is presented in Figure 7 taken from [38]. Features extracted with the Argus and Bro-IDS Tools are consist of packet-based and flow-based features. Table 1 represents the data set statistics, explaining and presenting the numbers of flows and bytes and time spent for the experimental setup while obtaining the data.

All features categorized into three main groups: Basic, Content and Time along with the synthetic attack categories labeled accordingly. Additionally, flow features and additional features presented and described in Table 2 and Table 6. All features described in Tables 3-5. Table 7 and Table 8 represents the labels class along with the data set distribution among the network traffic classified as normal and anomaly depending on the type of the attack.

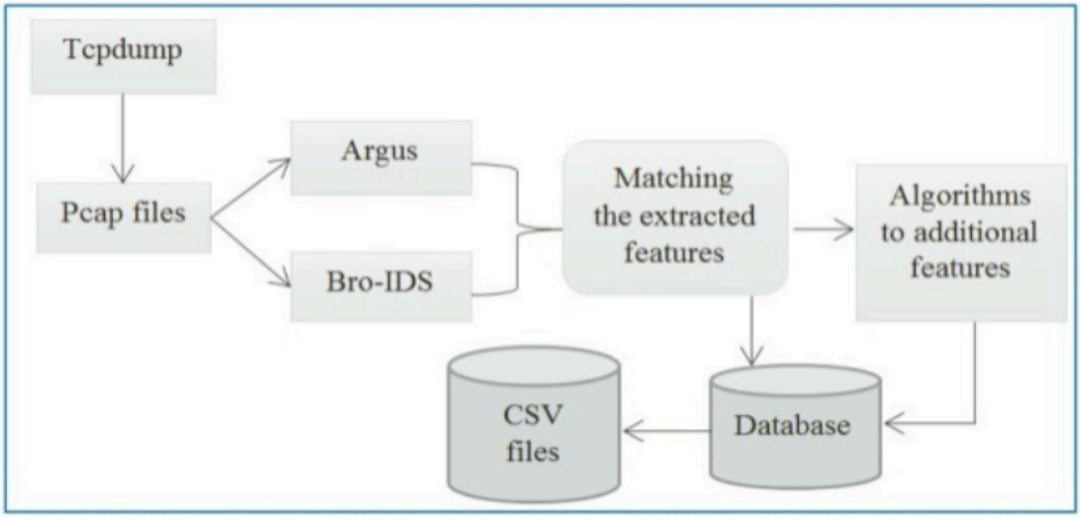


Figure 7. Feature extraction with the Argus and Bro-IDS Tools [5]

Table 1. Data set Statistics [38]

Statistical features		16 hours	15 hours
Number_of_flows		987,627	976,882
Src_bytes		4,860,168,866	5,940,523,728
Des_bytes		44,743,560,943	44,303,195,509
Src_Pkts		41,168,425	41,129,810
Dst_pkts		53,402,915	52,585,462
Protocol types	TCP	771,488	720,665
	UDP	301,528	688,616
	ICMP	150	374
	Others	150	374
Label	Normal	1,064,987	1,153,774
	Attack	22,215	299,068
Unique	Src_ip	40	41
	Dst_ip	44	45

Table 2. FLOW FEATURES [38]

#	Name	T.	Description
1	proto	N	Transaction protocol

Table 3. BASIC FEATURES [38]

#	Name	T	Description
2	state	N	The state and its dependent protocol, e.g. ACC, CLO, else (-)
3	dur	F	Record total duration
4	sbytes	I	Source to destination bytes
5	dbytes	I	Destination to source bytes
6	sttl	I	Source to destination time to live
7	dttl	I	Destination to source time to live
8	sloss	I	Source packets retransmitted or dropped
9	dloss	I	Destination packets retransmitted or dropped
10	service	N	http, ftp, smtp, ssh, dns, and (-) (if not much used service)
11	sload	F	Source bits per second
12	dload	F	Destination bits per second
13	spkts	I	Source to destination packet count
14	dpkts	I	Destination to source packet count

Table 4. CONTENT FEATURES [38]

#	Name	T	Description
15	swin	I	Source TCP window advertisement
16	dwin	I	Destination TCP window advertisement
17	stcpb	I	Source TCP sequence number
18	dtcpb	I	Destination TCP sequence number
19	smean	I	Mean of the flow packet size transmitted by the src
20	dmean	I	Mean of the flow packet size transmitted by the dst
21	trans_depth	I	The depth into the connection of http request/response transaction
22	res_body_len	I	The content size of the data transferred from the server's http service

Table 5. TIME FEATURES [38]

#	Name	T	Description
23	sjit	F	Source jitter (mSec)

24	djit	F	Destination jitter (mSec)
25	sintpkt	F	Source inter-packet arrival time (mSec)
26	dintpkt	F	Destination inter-packet arrival time (mSec)
27	tcprtt	F	The sum of 'synack' and 'ackdat' of the TCP.
28	synack	F	The time between the SYN and the SYN_ACK packets of the TCP.
29	ackdat	F	The time between the SYN_ACK and the ACK packets of the TCP.

Table 6. ADDITIONAL GENERATED FEATURES [38]

#	Name	T	Description
General purpose features			
30	is_sm_ips_ports	B	If source (1) equals to destination (3) IP addresses and port numbers (2)(4) are equal, this variable takes value 1 else 0
31	ct_state_ttl	I	Number for each state (6) according to specific range of values for source/destination time to live (10) (11).
32	ct_flw_http_mthd	I	Number of flows that has methods such as Get and Post in http service.
33	is_ftp_login	B	If the ftp session is accessed by user and password, then 1 else 0.
34	ct_ftp_cmd	I	Number of flows that has a command in ftp session.
Connection features			
35	ct_srv_src	I	Number of connections that contain the same service (10) and source address in 100 connections according to the record last time.
36	ct_srv_dst	I	Number of connections that contain the same service (10) and destination address in 100 connections according to the record last time.
37	ct_dst_ltm	I	Number of connections of the same destination address in 100 connections according to the record last time.
38	ct_src_ltm	I	Number of connections of the same source address in 100 connections according to the record last time.

39	ct_src_dport_ltm	I	Number of connections of the same source address and the destination port in 100 connections according to the record last time.
40	ct_dst_sport_ltm	I	Number of connections of the same destination address and the source port in 100 connections according to the record last time.
41	ct_dst_src_ltm	I	Number of connections of the same source and the destination address in 100 connections according to the record last time.

Table 7. LABELLED FEATURES [38]

#	Name	T	Description
43	attack_cat	N	The name of each attack category. In this data set, nine categories (e.g., Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms)
44	Label	B	0 for normal and 1 for attack records
Type (T.) N: nominal, I: integer, F: float, and B: binary			

Table 8. DATA SET RECORD DISTRIBUTION [38]

Type	Number of Records	Description
Normal	2,218,761	Natural transaction data.
Fuzzers	24,246	Attempting to cause a program or network suspended by feeding it the randomly generated data.
Analysis	2,677	It contains different attacks of port scan, spam and html files penetrations.
Backdoors	2,329	A technique in which a system security mechanism is bypassed stealthily to access a computer or its data.
DoS	16,353	A malicious attempt to make a server or a network resource unavailable to users, usually by temporarily

		interrupting or suspending the services of a host connected to the Internet.
Exploits	44,525	The attacker knows of a security problem within an operating system or a piece of software and leverages that knowledge by exploiting the vulnerability.
Generic	215,481	A technique works against all block- ciphers (with a given block and key size), without consideration about the structure of the block-cipher.
Reconnaissance	13,987	Contains all Strikes that can simulate attacks that gather information.
Shellcode	1,511	A small piece of code used as the payload in the exploitation of software vulnerability.
Worms	174	Attacker replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it.

Tables 1-8 representing the features along with the descriptions taken from the official literature [38] prepared for the UNSW-NB15 network data set.

The details of the twelve additional features presented in Table 6 are generated with the help of matched features (e.g., Tables 2-4). Authors of the UNSW-NB15 network data set [38] stated that those features are divided into two parts according to the nature and purpose of the additional generated features.

1. First part (features 30-34), are considered to be general purpose features, whereby each feature has its own purpose, according to protect the service of protocols (defence point of view).

2. Second part (features 35-41), are labelled as connection features. that are built from the flow of 100 record connections based on the sequential order of the last time. The attackers might scan hosts in a capricious way. For instance, one scan per minute or one scan per hour [39]. For this reason, those features are intended to sort and identify these attackers accordingly.

Note that in Moustafa and Slay's work [38], they described "last time" feature, also referred to as "record last time" in Table 6. The description and information provided in their paper is limited. For the purpose of this research, it was assumed that every connection feature might have different record section. The experimental setup was captured twice (16 hours and 15 hours), where was more than 100 connections, however for each connection feature they calculated only last 100 sequential connections, indicated as the last experiment record time. For example, packets were captured between 0h and 16h. The last 100 sequential connection for feature "ct_dst_ltm" captured between 12h to 15h, whereas last 100 sequential connection for feature "ct_srv_src" is captured between 14h to 16h. This assumption was based on the experimental setup performed by Moustafa and Slay, but the "last time" feature is not part of their released dataset.

Current thesis utilized the UNSW-NB15 dataset obtained from the original source [5] with total number of 82332 data points, and 42 features. Distribution of the data samples of UNSW-NB15 dataset provided in numbers and percentage in Table 9.

Table 9. Distribution of the data instances among the label class

Type	Number of instances	Total	In percentage
Normal	37,000	37,000	45%
Fuzzers	6,062	45,332	55%
Analysis	677		
Backdoors	583		
DoS	4,089		
Exploits	11,132		
Generic	18,871		
Reconnaissance	3,496		
Shellcode	378		
Worms	44		

3.2 Stage 2. Data Pre-processing

Data pre-processing is a vital step for ML tools that applied in the algorithm. The quality of data which is often incomplete, inconsistent or lacking some trends or overloading, directly influences the performance of the model and ability to derive valuable

information. That is why this stage is crucial before data is going to be trained and feed to the primary algorithm model. This step is proved to transform raw data into cleaner representation by eliminating overloading an unnecessary noise and information from the given data set to achieve better results.

To achieve the most appropriate clean process, data needs to undergo through the few steps before being used in the next stages. The feature selection is a significant step in the whole process, due to its high impact on the performance of the model. This process allows you to automatically find and process the most useful and informative features for the learning pipeline. Not all the features work same as most informative ones, some of the additional features affect negatively to the whole process by decreasing the speed of computation and training, model interpretability and more importantly by reducing the overall performance of the model.

"Feature selection is itself useful, but it mostly acts as a filter, muting out features that aren't useful in addition to your existing features." [40]

Robert Neuhaus

Not all of the features obtained in the data acquisition stage equally influence the selected models. Some of the features might have zero-importance relatively to the label class, some profound importance. Some features might have unique value or correlate to the model. That is why this process is used to eliminate redundant features from the data set with less informative gain relevant for the predicting class and provide better accuracy of the model based on the information stored in the selected variables and attributes.

3.2.1 Feature selection

In order to improve accuracy and efficiency of the classifier methods, three main feature selection methods usually are applied [41].

1) ***Filter method*** **–** this method generally performs selection independently from the classifier and doesn't carry induction algorithms. The method involves statistical measure or distance between the classes to assign a specific score to each feature. The selection or removal of the feature from the further training process decided by the scoring rank of each feature found in the data set and considered individually or respectively to the dependent variable.

Some of the examples of the filter methods listed as Chi-square, Entropy, Fisher's score, information gain, correlation, and one-attribute rule.

2) ***Wrapper method*** **–** in this method classification of the features is performed to obtain better results and performance. The process based on search algorithms, and each combination with the set of features is evaluated to match the best possible outcome for the concrete machine learning algorithm. However, they tend to be slower, since it requires more considerable computational resources and not all these proposed sets are optimal for every other machine learning algorithm.

   The search process mainly divided into three categories: Forward feature selection, Step backward feature selection, and Exhaustive feature selection. An example of the wrapper method can be recursive feature elimination, sequential feature selection algorithms, and genetic algorithms.

3) ***Embedded method*** **–** this method selects the feature based on a learning process in each iteration of the algorithm. Each iteration is equally essential for the process to extract those particular features that contribute the most for the training procedure. Even though the search guided by the learning process might be similar to the wrapper method, it follows with less computational consumption.

   The most common approach for the embedder technique is decision tree classification algorithm and another regularization method in the concept of LASSO.

Since the wrapper method is computationally expensive and embedded model is computationally demanding, where each iteration needs high computational resources, in this research study, the filter method is used for the feature selection process. This method uses independent evaluation criteria for every feature and less computational load which flow-based network traffic would require for each separate feature selection.

The given data set includes both numerical and categorical features. However, the inclusion of the three categorical (namely: proto, service, state) values does not affect the overall accuracy results of detecting the label class (see Appendix 1), compared to the use of only numerical features. Thus, it can be proven with the experiment results executed to measure the performance of the model with all features in comparison to only

numerical features. In the experiment, a dataset with the 82332 data points trained and tested using cross-validation method with 10 folds on the initially predefined five classification models. The results for both cases remained the same both for binary classification (around 94%) and multi-class (approximately 81%) label. For this reason, in this thesis, only numerical features and their selection are considered.

3.2.2 Numerical feature's selection

Binary selection of the features is major characteristics for filter-based feature selection which helps to maximize some performance of the model. Numerous filter-based selection criteria's like information gain [42], Laplacian score [43], ReliefF [44] and others were studied and proposed to be utilized in the last decades, where Fishers Score was the most widely used one, due to its overall favorable performance [45].

The Fisher's score intended for the numeric variables to measure the ratio of the average interclass to the average of the intraclass separation [46]. The discriminatory power of the attribute evaluated by the range of the score, which is calculated by the given formula:

$$\mathrm{F} = \frac{\sum_{j=1}^{k} p_j \left(\mu_j - \mu\right)^2}{\sum_{j=1}^{k} p_j \sigma_j^2}$$

Where μ_j and σ_j are the mean and the standard deviation of the class j, j = 1, 2, 3 … k, corresponding to the n-th feature. The value μ represented as the global mean of the whole data set, and p_j is the fraction of the data points belonging to the class j.

The Fisher's score formula is applied to identify the score for each feature in respective of the two-class label and multi-class label. All features tested with the help of decision tree classification model and performance tested by iterating the feature with the highest score by adding every next feature till the least ranked. No direct threshold dictates the importance of the features to the specific model, and the scoring results might be different for a different type of the datasets and label class. Top features selected for the training stage are those that are higher compared to the others in the same feature set. According to the results shown in Table 10, F score for two-class label value is relatively lower than the multi-class label F score.

Fisher's score was implemented and applied to the two-class label and multi-class label with the distribution records of different attack types, to select potentially the best features with the highest discriminatory value depending on the label class among all the given features set.

The result for the two-class label and multi-class label feature ranking is given below in Table 10. The Fisher's (F) score applied to 39 numerical features.

Table 10. The Fisher's score results

Ranking	Feature name	Two-class label Score	Feature name	Multi-class label Score
1	sttl	0.271061	ct_dst_sport_ltm	1.069932
2	swin	0.174412	ct_srv_dst	0.916540
3	ct_dst_sport_ltm	0.168184	ct_src_dport_ltm	0.853288
4	dwin	0.132030	ct_srv_src	0.850996
5	ct_src_dport_ltm	0.120486	ct_dst_src_ltm	0.714809
6	rate	0.105343	ct_dst_ltm	0.667196
7	ct_state_ttl	0.087106	swin	0.575036
8	ct_srv_dst	0.083646	ct_src_ltm	0.547743
9	ct_srv_src	0.081759	sttl	0.500799
10	ct_dst_src_ltm	0.075949	dwin	0.490655
11	ct_src_ltm	0.074142	dttl	0.310571
12	dtcpb	0.071437	rate	0.262325
13	stcpb	0.070716	stcpb	0.245961
14	ct_dst_ltm	0.064143	dtcpb	0.244908
15	dload	0.063985	ct_state_ttl	0.164801
16	dmean	0.037995	dmean	0.117552
17	synack	0.017699	dload	0.085526
18	tcprtt	0.017568	tcprtt	0.082582
19	sload	0.012956	ackdat	0.081581
20	ackdat	0.011493	synack	0.062480
21	sinpkt	0.011102	smean	0.060272
22	is_sm_ips_ports	0.010466	sload	0.026655
23	dttl	0.008224	dur	0.019671

24	ct_flw_http_mthd	0.004435	ct_flw_http_mthd	0.019027
25	dpkts	0.003312	trans_depth	0.016491
26	smean	0.003114	sinpkt	0.015017
27	dloss	0.001729	is_sm_ips_ports	0.013977
28	dinpkt	0.001102	sjit	0.013635
29	dbytes	0.000929	djit	0.010410
30	spkts	0.000698	dpkts	0.009701
31	sjit	0.000607	is_ftp_login	0.009144
32	trans_depth	0.000591	ct_ftp_cmd	0.008923
33	djit	0.000588	spkts	0.007467
34	sbytes	0.000391	dinpkt	0.007049
35	ct_ftp_cmd	0.000240	dloss	0.005882
36	response_body_len	0.000229	sloss	0.004907
37	is_ftp_login	0.000215	sbytes	0.004137
38	sloss	0.000037	dbytes	0.003909
39	dur	0.000001	response_body_len	0.001099

As can be noted from Table 10, some of the features having the highest score depending on the label class are similar. Namely, those features are ct_dst_sport_ltm and ct_src_dport_ltm that belongs to the group of connection features and having characteristics for evaluating number of connections between source/destination and destination/source ports for the last time among 100 connections.

Both features are holding almost the same characteristics differing only in the source and destination ports represented in Table 11.

Table 11. Matching top features for both scenarios (binary, multi)

ct_src_dport_ltm	No of connections of the same source address and the destination port in 100 connections according to the last time.	Integer
ct_dst_sport_ltm	No of connections of the same destination address and the source port in 100 connections according to the last time.	Integer

According to the multi-class label (Normal, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms) the most discriminative highest ranked 6 features (ct_dst_sport_ltm, ct_srv_dst, ct_src_dport_ltm, ct_srv_src, ct_dst_src_ltm, ct_dst_ltm) are considered and labeled as connection features. Those features are provided in the dataset for defense phase during the attempt depending on the connection scenarios [38]. According to the various scenarios, unpredictable way of scanning the host could be used by attackers, where some of them can take once per minute and others one scan per hour [39]. For this reason, connection features were created and extracted to the database to capture similar characteristics of the connection records for the last 100 connections.

As it was mentioned formerly scoring value of two most discriminative features (ct_dst_sport_ltm, ct_src_dport_ltm) are similar in two-class and multi-class label scenarios. Scoring value is obtained from Fisher's score ranking. However, the other four features named as *sttl*, *swin*, *dwin* and *rate* belong to the content features and basic features. Those features are mostly representing the integrated gathered information from the data packets. Those features are described in Tables 2-6 as Source/Destination TCP window advertisement and source to destination time to live.

Nevertheless, different groups of features are candidates for best discriminatory power, it was also noted that almost all the connection features created to analyze the defense stage during the attempt are getting the higher results compared to the other features. All of the connection features have a threshold of 0,06 and 0,24 for two-class and multi-class label respectively, and thus affecting the better accuracy results on the supervised classification, which described in the section 3.3.

3.3 Stage 3. Classification, Training

After settling with the dataset, data pre-processing, getting the most discriminative features by applying appropriate feature selection criteria and hypothetical testing the obtained input is ready for the next stage. In this stage, an input is ready to be processed and fed to the machine learning classification models. Different models were used to build an appropriate algorithm and compare performance results using classification algorithms with the help of supervised, semi-supervised and active learning approaches.

In the current thesis, the primary approach of botnet detection is achieved by active machine learning algorithms. However, supervised unsupervised and semi-supervised algorithms with the combination of different classification problems are also applied in or order to evaluate the difference and analyze the performance of each separate way. Dataset provides labeled samples to characterize attributes as normal network traffic and anomaly behaviors, whereas nine different attributes for each instance describe anomaly behavior. Malicious and normal traffic are measured with the numerical value 0 for normal and 1 for botnet, while separate instances within anomaly traffic are defined by the categorical value, identifying attack type. Thus, brings the classification problem to discriminate between two classes or label samples, as normal and anomaly (0 and 1), and between multi-class as normal and type of the attack vectors (Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms).

To achieve a classification of assigned one or more predefined categories, some instances from a given dataset should be used to train the model and the rest for testing or validation process. Depending on the problem training data is provided with fully labeled instances for supervised learning or with few labeled instances (dataset has labeled and unlabeled instances) for semi-supervised and active learning, while testing data is used to predict the label class. After the process is completed original labels and predicted instances are analyzed and compared to provide the performance of excellent and lousy classification practice. In supervised learning, dataset with fully labeled instances is mostly split into training with a ratio of 70% to train the model and testing part with the rest of 30% to test the model. For the semi-supervised and active learning labeled samples for training stage might vary from 1% to 99%, depending on the desired performance and number of instances. Following this process, the performance of the evaluation is determined.

Another similar but more complicated approach can be achieved with the help of K-fold Cross Validation, whereas data is divided into folds ensuring that every fold is used both as training and testing sets at some point. K-fold Cross Validation split data into k number of sections/folds and performs k iterations where the current iteration used to test the model and rest to train. The process repeats until every iteration has used given fold as a testing set. This approach provides better performance results compared to splitting the data into 70/30. In the current thesis work, both methods applied in different machine learning techniques.

In the testing experiment using a supervised classification approach, the most competitive five classification models were chosen. After initial analysis and performance of the models in detecting the label class XGBoost, Decision Tree and Random Forest gave overall better and similar results. As a result, the next stages of this thesis, utilized those models in experimental testing scenarios for semi-supervised and active learning algorithms.

3.3.1 Decision Tree algorithms

Decision tree algorithms are the most popular machine learning algorithms that fall under supervised learning and can be used both for solving regression and classification problems. Decision trees are easy to understand for human-level thinking and use the tree representation to solve the problem in which each node represents a feature (attribute), each branch represents a decision, and leaf node corresponds to an outcome (label class). The whole idea lies in creating a tree for the entire given data and process every result in the leaf by minimizing the possible error.

The decision tree has different ways to identify the attribute for the root node in each level. There are two primary attribute selection methods which are:

1. ***Information Gain*** - a measure of the change in the entropy when the root node in a decision tree for every level decided after the partition of the training instances into smaller subsets. Entropy is a standard measurement of the uncertainty of a random variable, and it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.

2. ***Gini Index*** - a metric to measure how often a randomly chosen element would be incorrectly identified. Thus, followed by the usage of the lowest Gini in each split. Sklearn supports both criteria's, however, uses Gini index as a default measure to calculate the root node.

Below, there is a simple visualization of the decision tree, based on the botnet and normal traffic with the help of the top ranked (Fisher's score) three features for binary classification. To visualize small tree only 40 entries from the original data set are used.

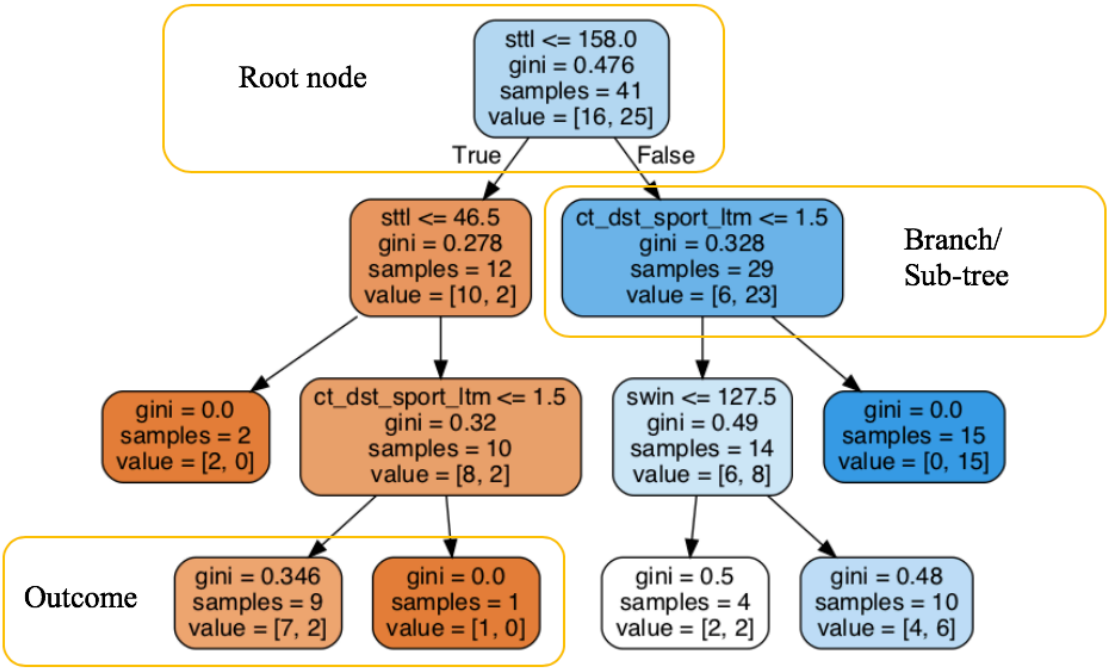


Figure 8. Example of simple Decision Tree based on the botnet and normal network traffic

Like all other models, Decision Trees has advantages and disadvantages for problem-solving as listed below.

Pros:

- Easy to understand, interpret, generate rules, visualize and draw
- Deals well with noise or incomplete data
- Can be used both for regression and classification problems
- Suitable both for categorical and numerical data
- Handles both binary and multi-class output problems
- Validation of the model can be achieved by using quantitative analysis and statistical tests, which makes the reliable decision model leading to good predictive results.

Cons:

- Instability, where small changes in the data lead to a significant difference in the model

- Overfitting, low bias but high variance, performance decreases or fails on testing data (unseen data) even if the score on the training data shown as highly accurate. This happens when the decision tree tries to fit all samples into training data by creating multiple branches with strict rules, which fails on training data. Thus, can be improved with the pruning method, where some branches of the tree tend to be removed.
- Complexity, decision trees are easy to use, however, decision making on the extensive data makes trees complex including many branches and time-consuming
- Cost, where large decision trees require advanced knowledge in quantitative and statistical analysis.

In the experimental setup of this thesis, all required functions to build the decision tree model using the classification problem were archived by using the scikit-learn Python library package.

3.3.2 Random Forest

Random Forest is easy to use and flexible machine learning algorithm that produces good performance result. Random Forest is a supervised learning algorithm used both for classification and regression tasks. Random Forest creates a forest, which is an ensemble of Decision Trees, by adding additional randomness and trains the model by merging decision trees to get more accurate and stable prediction results.

The model offers excellent performance and diversity in the results obtained from training by searching the best feature among the subset for the features, rather than searching for the most essential feature when splitting a node. Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. Trees can become more random, by additionally using arbitrary thresholds for each feature rather than searching for the best possible limits (comparing to the decision tree methods).

Like every other model there are Pros and Cons of the Random Forest as followed below:

Advantages:

- Works correctly for more extensive data items, due to having fewer variances

- Flexible, provides high accuracy results
- Preparation of the input and data scaling are not required
- Accuracy maintained even if massive data proportions are missing
- The process of averaging and combining the results obtained from different decision trees helps to solve the overfitting problem.

Disadvantages:

- Complexity: Harder to follow, not easily interpretable and time-consuming than constructing decision trees
- Requires more computational resources and are less intuitive.
- The prediction process is slow (time-consuming)

3.3.3 XGBOOST

"The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms. Which is the reason why many people use xgboost" [47]

\- *Tianqi Chen*

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable [25]. Gradient Boosting framework is used to implement machine learning algorithms. Many data science problems solved in a fast and accurate way under XGBoost, which provides a parallel tree boosting. In its turn gradient boosting is a machine learning technique for regression and classification problems. It produces a prediction model in the form of weak prediction models, generally decision trees. XGBoost belongs to a broader collection of tools under the Distributed Machine Learning Community or DMLC, the creators of the popular MXNet [48] deep learning library. XGBoost is a software library that can be downloaded and installed on the machine and accessed from a variety of interfaces. In this thesis, XGBoost library is used and implemented under Python interface in scikit-learn.

The main reasons for the utilization XGBoost in the project are the execution speed and model performance. Compared to the other gradient boosting implementations XGBoost is very fast. Regarding to the performance of the model, XGBoost was a winner in the most of Machine learning competitions providing the highest performance results. Even

though the library mainly focuses on computational speed and model performance, it also offers many advanced features that are described as Model Features, System Features, and Algorithm Features.

Model Features

Scikit-learn supports the implementation of the features in the model with some regularization. Three primary forms of gradient boosting features that supported are described below:

1. Gradient Boosting algorithm also known as gradient boosting machine including the learning speed.
2. Stochastic Gradient Boosting includes subsample in a row, column, and a column for separate levels.
3. Regularized Gradient Boosting contains L1 and L2 regularization.

System Features

The library provides the usage to a system in a variety of computing environments:

- All CPU cores uses the parallelization tree construction while training.
- Distributed Computing for training huge models utilizing a cluster of machines.
- Out-of-Core Computing for extensive datasets that do not fit into the memory.
- Optimization of the data structure cache and algorithm for the best use of hardware equipment.

Algorithm Features

The implementation of the algorithm was designed to calculate time and memory resources efficiently. The purpose of the development was to maximize the use of available resources for the training stage of the model. Some key features of the implementation of the algorithm include:

- Sparse Aware deployment with automatic process of missing data values.
- Block Structure supports the parallelization of tree construction.
- Continuous Training for further boost in already fitted model with new data.

- XGBoost is free, open source software available for use under the Apache-2 license.

In the present thesis, XGBClassifier is utilized in different training and testing scenarios under the Python interface in scikit-learn.

3.4 Stage 4. Performance Evaluation Metrics

In order to classify obtained results as poor or satisfying, evaluation of the machine learning algorithms is essential. Several classification metrics are used to ensure expectations for the detection performance and efficiency of the experimental setups.

The fundamental quantities used in the definitions of performance criteria, where N is a number of classifications, are as follows:

Table 12. Confusion Matrix

		Labels returned by the classifier	
		positive	*negative*
True labels	*positive*	N_{TP}	N_{FN}
	negative	N_{FP}	N_{TN}

3.4.1 Correct and Incorrect Classification

When testing a classifier on a defined sample, those four fundamental quantities used:

- The example is positive, and the classifier correctly recognizes it as such (true positive);
- The case is negative, and the classifier correctly identifies it as such (true negative);
- The example is positive, but the classifier labels it as negative (false positive);
- The case is negative, but the classifier labels it as positive (false negative);

The following definitions are originally from [49].

3.4.2 Classification Metrics and Performance Evaluation

3.4.2.1 Error Rate and Classification Accuracy

One of the primary and fundamental detection performance metrics used in supervised learning algorithms is error rate and classification accuracy as stated from the name. It is calculated by diving either correct classification or frequency of the errors by the total number of given samples.

$$E = \frac{N_{FP} + N_{FN}}{N_{FP} + N_{FN} + N_{TP} + N_{TN}}$$

$$Acc = \frac{N_{TP} + N_{TN}}{N_{FP} + N_{FN} + N_{TP} + N_{TN}}$$

$$\text{Acc} = 1 - \text{E}$$

$$|\text{T}| = \text{N}_{\text{FP}} + \text{N}_{\text{FN}} + \text{N}_{\text{TP}} + \text{N}_{\text{TN}} \text{ (size of the set)}$$

Classification accuracy is the ratio of number of correct predictions to the total number of input samples. Accuracy metric works well, when the data set has equal number of samples belonging to each class. Accuracy is considered as the most common and universal performance metric for classification problems measured by using straightforward and intuitive way. However, both accuracy and error rate can give a poor measure results for imbalanced data. Hence, there is an opinion that accuracy is improper scoring rules [50] for imbalanced data (with different numbers of samples belonging to each class, whereas one class considered to be on the much higher side than the others). For this reason, many authors in [7] [9] used these metrics used along with the Precision and Recall giving broader performance overview of the experiment for imbalanced data.

3.4.2.2 Precision and Recall

Some datasets most commonly have a higher amount of negative results than positive. In this sense, calculating the error rate might misdirect the classification performance. For this reason, calculating the ratio of items classified as positive (X) among all examples which are labeled as X is more sensible. Showing the probability that the classifier is right is called Precision (Pr) or Positive Predictive Value (PVV) and achieved by the following equation:

$$Pr = \frac{N_{TP}}{N_{FP} + N_{TP}}$$

Another way of phrasing the Precision is to say, that calculation of the probability classifier is right when labeling an example as positive [49]. Holding this meaning precision calculation can be used for a multi-class problem considering unsupervised and semi-supervised algorithm approaches.

3.4.2.3 Sensitivity and Specificity

Another way to assess the performance of the algorithm is to define the probability of positive examples among all positive cases in the set. But, since the particular choice of the criteria might influence the given application area, sensitivity and specificity are accustomed as Recall measured on positive and negative examples which represented as following formulas:

Sensitivity or Recall measured on positive examples:

$$Se = \frac{N_{TP}}{N_{FN} + N_{TP}}$$

Specificity or Recall measured on negative examples:

$$Sp = \frac{N_{TN}}{N_{FP} + N_{TN}}$$

3.4.2.4 Combination of Precision and Recall

Sometimes when the desired outcome cannot be decided on an exact parameter, neural value combination of precisions and recall is used. Authors of [7] have chosen this method as one of the performance evaluation metrics to measure various algorithms used in their work.

$$F_1 = \frac{2 * Pr * Re}{Pr + Re}$$

3.4.2.5 ROC and AUC Curves

Another performance measurement is AUC ROC Curves. In binary and multi-class classification problems, where specific parameters can modify N_{FP} and N_{FN}, it highly

predicted that the overall classifier behavior could be affected and improved by tweaking particular settings (parameters). This logic is used when a user is sure about specific quantities and which focus is more critical for the training dataset. According to “Towards Data Science” blog, when it comes to label samples with various threshold settings, “AUC (Area Under the Curve) ROC (Receiver Operating Characteristics) curve is the most important evaluation metric for checking any classification model's performance because it tells how much model is capable to distinguishing between classes." [51]. Naturally, it works by distinguishing between classes by predicting positive or negative labeling supported by evidence of nearest neighbors’ classification.

3.4.3 Selected Performance Metrics

Since, the data set utilized in this research has more balanced prediction class with similar amount of normal and botnet samples, *accuracy* evaluation metrics are performed in all experimental setups in this master thesis. In addition, *confusion matrix* was applied in active learning experiments, to be able to see a full range of the positive and negative samples for a correct label returned from the training phase. Nevertheless, some of the evaluation metric described above used in different parts of training, validation and analysis stage. *Accuracy* was decided as primary metric for evaluating the final result and visualize the graphs obtained from the process.

4 Botnet Detection: Practical Implementation

To obtain a proper botnet detection, machine learning (classification) algorithm/model, implementation was performed on the Python programming language as a base. Scikit-learn library [52] and supportive libraries like panda [53] were additionally used for some mathematical equations. According to the official documentation, scikit-learn is an open source, freely accessible for everyone, simple and efficient tool for data mining and data analysis [52]. This library is reusable in various contexts and build on widely used numerical and scientific libraries like NumPy, SciPy, and matplotlib. Scikit-learn has the features and functionality of providing resources for Classification, Regression, Clustering, Pre-processing, Model selection, and Dimensionality Reduction [52]. In this thesis the power of Scikit-learn used as the main library for designing the machine learning algorithm. Data pre-processing and training stages are achieved on the base of Scikit-learn library, with the use of classification models (Decision Tree, Random Forest, XGBoost). Testing and validation that were implemented on the same base and provided systematic performance outputs. The workflow architecture presented in Figure 9.

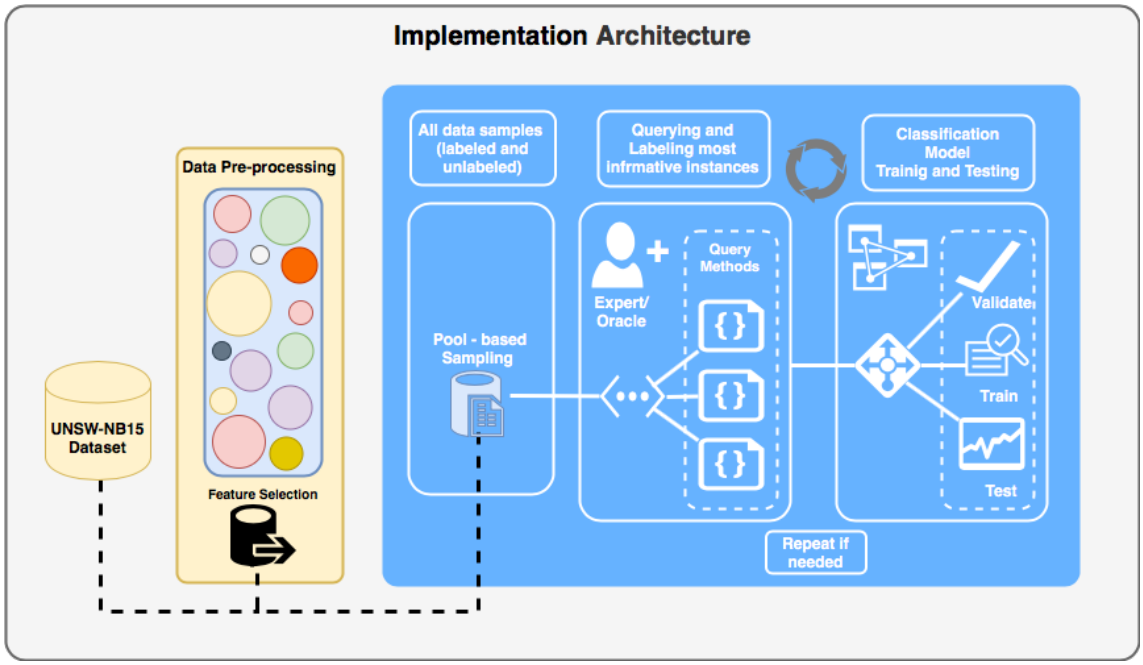


Figure 9. Implementation Architecture

4.1 Selected numerical features for botnet detection

In the previous stages, it was described that Fisher's score applied to the 39 numerical features. 19 (see Figure 10) of them were proven potentially good with discriminative power in respect of binary class and multi-class supervised classification. The dataset with the 82332 samples was used to train and test the Decision Tree model (model chosen arbitrary). Sklearn train_test_split technique with 70% training and 30% testing data and cross-validation method with 10 folds were applied.

All features from the highest rank to the lowest were classified to identify the statistical probability of distribution and effect on the accuracy of the desired label and prediction results. Results are obtained by splitting the data into training 70% and testing 30% sets with the help of train_test_split function and cross-validation method by folding the data 10 times. Results of classification using supervised Decision Tree model are given in Figure 10 both for binary and multi-class. The difference in accuracy for both ways can be seen by mostly 2 %. However, the results are primarily the same, which can be visualized from the graphs represented below.

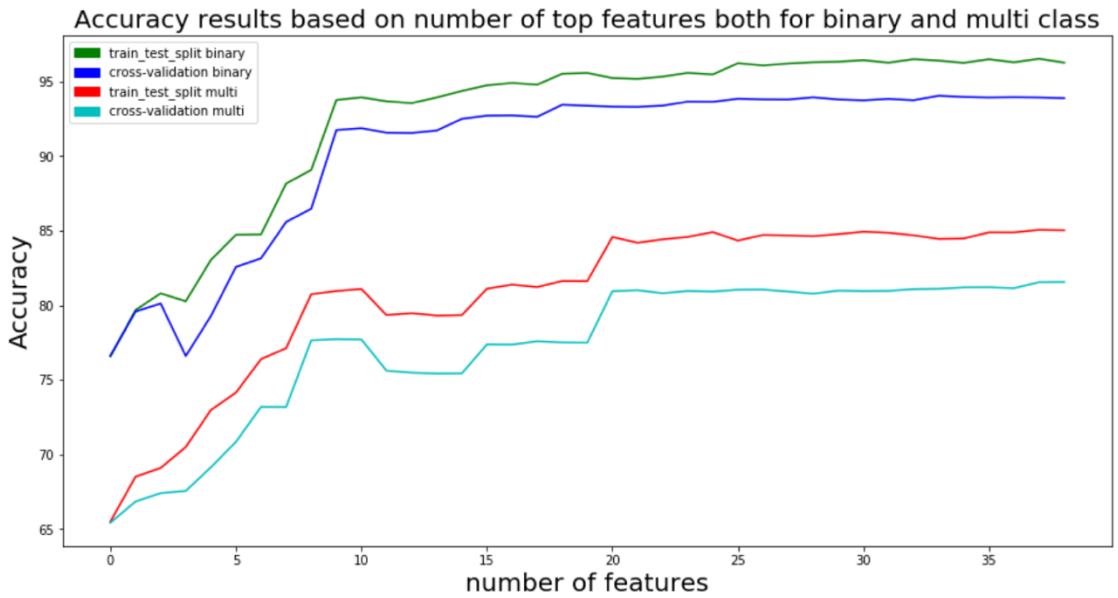


Figure 10. Accuracy performance results for both binary and multi class

As can be seen from the graph above, the accuracy results for the binary class are mostly stable after applying 19 and more features. Those top most informative 19 features are involved in the further stages of training and classification for the binary class task to

reduce overfitting of the selected models. Additional graphs with separate label class are provided in Appendix 2.

For multi-class label accuracy results are fluctuating more on the same scale after applying 21 highest ranked numerical features. In the same way, next stages of the experimental setup utilized those top 21 features according to the Fisher's score ranking.

In both scenarios (binary and multi) the firsts breakout can be seen from the graph after applying 9 top features, reaching the accuracy around 80% for multi-class and 90% for binary class, dominated by the connection features.

Table 13 shows the accuracy metrics for each different range and set of features according to their highest discriminative score to the label class.

Table 13. Accuracy metrics of the model trained with top numerical features

Number of features	**Binary classification (normal, anomaly)**		**Multi-class classification (normal and attack types)**	
	Cross validation	**Train test split**	**Cross validation**	**Train test split**
Best feature	76.63	76.57	65.43	65.51
3 best features	80.12	80.80	67.41	69.10
10 best features	91.74	93.76	77.73	80.96
19 best features	93.44	95.51	77.51	81.62
21 best features	93.31	95.23	80.95	84.57
All features	93.88	96.26	81.56	85.03

Table 13 shows that:

- The highest ranked single feature (which is different for the binary and multi-class) is capable to identify the botnet in the network flow by 76% in the binary classification and 65% in the multi-class. Even though, the performance results are different and relatively low, it is still good indicator for the binary classification.

- Discriminative power gradually increases in both cases with the addition of the next essential features. By training the model with the most three discriminative features the accuracy reached 80% in binary class and 67-69% in the multi-class.

- Visible accuracy improvements for binary class label were noted when top valued 10 features were included in the training stage. Accuracy metrics increased by 10% compared to the 3 feature accuracy metrics. However, for multi-class label the results kept almost on the same scale with 2-3% of fluctuation.

- Gradual improvements for the multi-class label were introduced while training model with the 21 best features proving the accuracy results in 84%. The results are mostly stable after including the rest of the features.

- The best result according to the training stage for two-class label hit the record with 93-95% kept from 19 most important features until the full feature set in the training process.

4.2 Classification algorithms

As stated in the background part, the main idea for machine learning classification algorithms is to categorize given data into desired and distinct number of classes.

Since there are many options to build a classification model, some of the critical elements need to be compared with others in order to choose the most convenient one for the desired problem to be solved. The list of those characteristics are explained below:

- *Prediction Accuracy*, determining the ability to estimate the correct prediction of the testing samples.

- *Speed*, determining the computational cost and time to process the data.

- *Scalability*, determining the amount of training data to estimate the parameters.

- *Interpretability*, determining the level of the interpretation of understanding the model.

- *Simplicity*, addressing the level of complexity of the model. The ability to be less complicated without losing its efficiency on the performance.

For the initial model testing purpose, XGBoost (XGB), Decision Tree (DT), K-Nearest Neighbor (KNN), Random Forest (RF), Linear Regression (LR) selected as competing and most referred [54] [55] [56] models. Those models commonly used for predicting the anomaly and provides excellent feedback. However, to identify the most appropriate one for the specified and selected data set, the results of the given models need to be compared and evaluated. To choose the best for the further training, these five models are trained and tested using cross-validation method with 10 folds both for binary and multi-class label classification. Accuracy was defined as the primary and common performance evaluation metrics in both cases. For this test 44 features and 82332 instances are used. The results presented in Table 14.

Table 14. Performance results for different models

Scenario/Models	**Binary Classification**			**Multi class classification**
	Accuracy	*Recall*	*Precision*	*Accuracy*
XGB	94.76	95.36	95.35	84.04
DT	94.16	95.50	94.25	81.99
KNN	87.79	87.33	90.54	75.47
RF	95.74	95.39	97.24	83.63
LR	86.18	87.85	88.23	74.04

In this testing experiment the most competitive classification models were chosen using a supervised classification approach. As shown in Table 14, XGBoost, Decision Tree and Random Forest gave overall better and similar results. As a result, on the next stages of this thesis, those models will be utilized in experimental testing scenarios with semi-supervised and active learning algorithms.

Next testing scenario was implemented to identify separability of the data by visualizing it in the three-dimensional plot. For this case, three previously selected models and top 3 features based on the Fisher's score raking were used. Different scenarios described in Table 14 along with the performance results and the plot graphs are represented in Figures 12 - 24. The cross-validation approaches training and testing with k=10. In multi-class problems only accuracy metric is used, since multiclass format is not supported in scikit cross-validation calculation technique.

Table 15. Performance results of the modes according to the given scenarios

Scenario/Models	XGB	DT	RF
Scenario 1: Binary class Top three features *(sttl, swin, ct_dst_sport_ltm)*	Accuracy: 81.72 Recall: 79.76 Precision: 84.34	Accuracy: 82.28 Recall: 79.79 Precision: 84.32	Accuracy: 82.28 Recall: 79.76 Precision: 84.36
Scenario 2: Multi class Top three features *(ct_src_dport_ltm,* *ct_dst_sport_ltm,* *ct_srv_dst)*	Accuracy: 67.69	Accuracy: 67.49	Accuracy: 67.61
Scenario 3: Binary class Random three features (from top 19) *(dpkts, rate, sttl)*	Accuracy: 82.25 Recall: 84.46 Precision: 83.53	Accuracy: 81.58 Recall: 88.35 Precision: 82.93	Accuracy: 81.75 Recall: 87.95 Precision: 82.91
Scenario 4: Multi class Random three features (from top 19) *(dpkts, rate, sttl)*	Accuracy: 66.92	Accuracy: 63.07	Accuracy: 63.36

Figure 11 - 14 represents the scatter plot with the set of different three features defined in the Scenarios. Original label class of the data points and classification results of XGBoost, Decision Tree and Random Forest models are provided. All the scatter graphs with different sets of features were attached in Appendix 3.

As can be stated from figures represented below, network traffic with normal (green) and malicious (red) behavior using sttl, rate, and dpkts features, provides a good separability with a minimal collision of the data points by creating layers and rows. For multi-class

scenario shown in Figure 11 and Figure 12 data separability can also be noted. However, some collisions between attack types are kept.

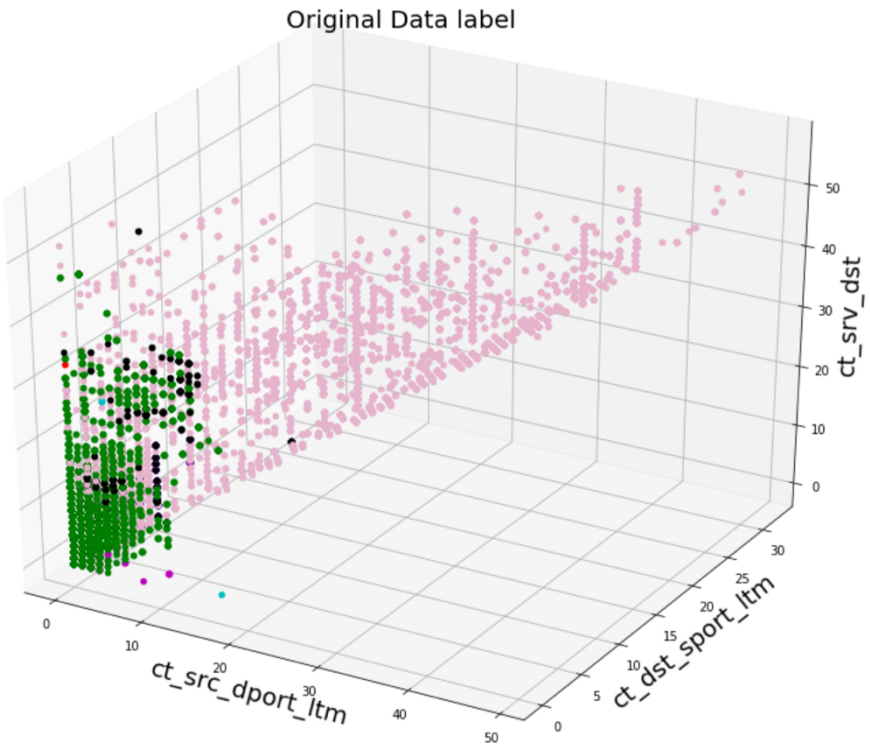


Figure 11. Scenario 2. Original data labels

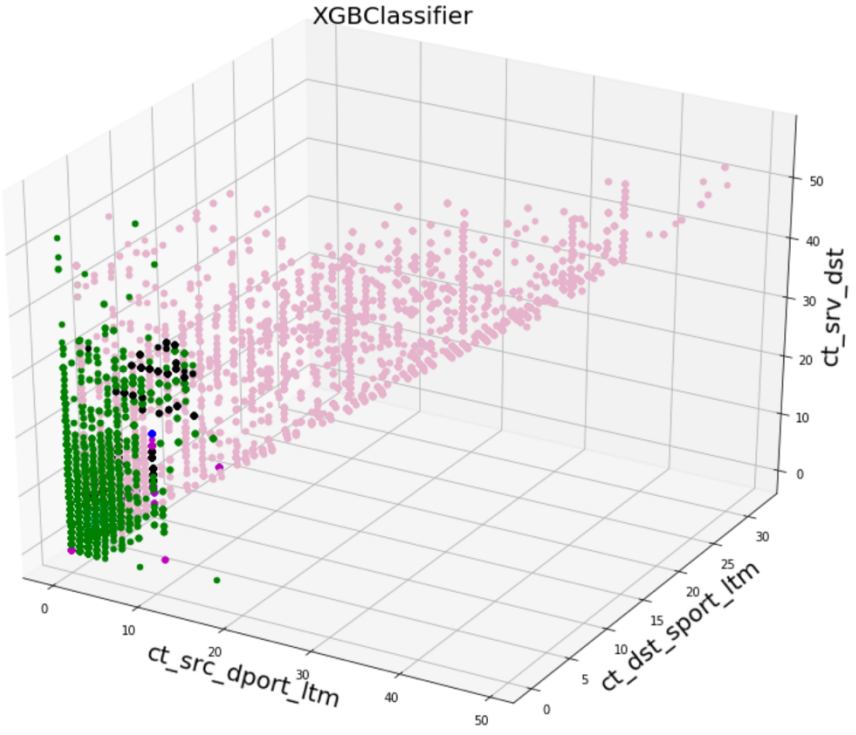


Figure 12. Scenario 2. XGB prediction results

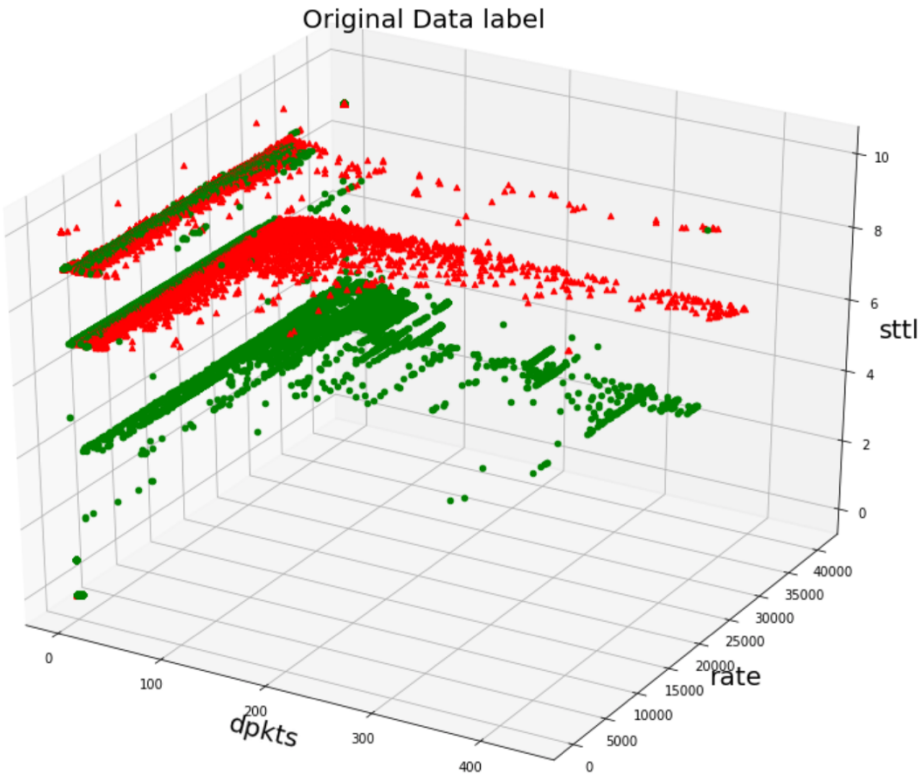


Figure 13. Scenario 3. Original data labels

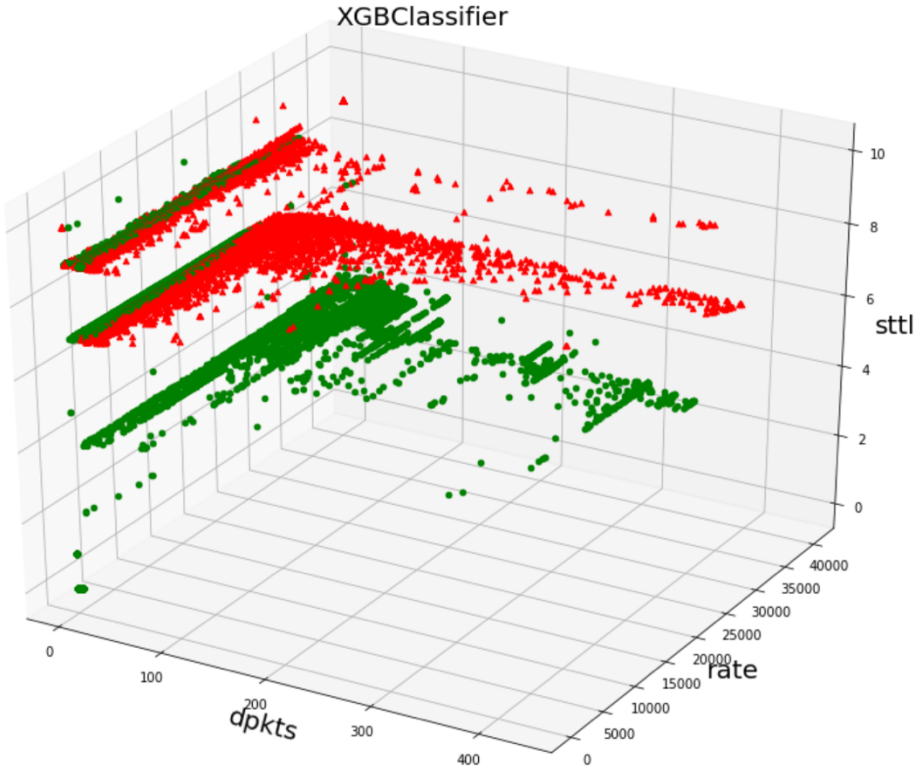


Figure 14. Scenario 3. XGB prediction results

4.3 Unsupervised Learning

At this point, unsupervised learning was applied to identify if there is a natural ability of data points to cluster. For this reason, testing scenario utilizes Clustering method, which allows splitting the dataset into groups according to the most common similarity. Identifying different groups will help to see the separability of the data points and threshold segments. DBSCAN provides excellent results in separating clusters of high density over low density. For this reason, it was utilized to see the possible separability of the data points into clusters/groups. In this thesis, DBSCAN was applied only for graphical demonstration of data points separability.

Experiment with the use of unsupervised learning was performed to understand the separability of the data points regarding the label class and to analyze results for a better understanding of the data set. To approach data point separability results and plot the scatter graphs, only three different features according to the specified scenarios were used.

The application of DBSCAN results are not considered in the active learning algorithm. However, this method can be studied separately as a different technique, which requires different scope of outline analysis. For the future studies, unsupervised clustering can be studied deeply as a separate subject and considered as one of the ways to approach similar study.

4.3.1 DBSCAN Clustering

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular clustering algorithm used to separate clusters of high density from clusters of low density. DBSCAN divides the dataset into n dimensions, and for each data point, it forms an n-dimensional shape around that point. This shape was defined as a cluster and each cluster expands iteratively by counting the number of nearest data points. The density of each region classified into core, border and noise points. The explanation of each point taken from [46] and as follows:

1. *Core point:* A data point is defined as a core point if it contains at least τ data points. Where τ (minimum points within ε (eps) distance) is a threshold for the number of the data points in the same neighborhood.

2. *Border point:* data point is defined as a border point if it contains less than τ points (minimum points within ε (eps) distance), but it also includes at least one core point within a radius Eps. Epsilon (Eps) is the radius given to test the distance between data points.
3. *Noise point:* A data point that is neither a core point nor a border point.

The *silhouette coefficient* achieves cluster validation since it provides a good intuitive feel of the clustering quality. The overall silhouette coefficient is the average of the data point-specific coefficients. It is calculated by the given formula where $Davg_i^{in}$ in is the average distance of X_i to data points within the cluster of X_i. And $Dmin_i^{out}$ represent the minimum of these (average) distances, over the other clusters [46].

$$S_i = \frac{Dmin_i^{out} - Davg_i^{in}}{max\{Dmin_i^{out}, Davg_i^{in}\}}$$

The scikit-learn implementation doesn't require some clusters as an input to run the code. However, it provides a default value of 0.5 for the eps parameter, which can be tuned to get the desired number of clusters.

For this implementation, Eps parameter tuned to get several clusters closer to the original multi-label class given in the labeled dataset. Since the active learning approach does require some labeled data, eps parameter can be tuned easily according to the essential problem. For existing task, Eps value was tuned, to obtain the closest number of clusters that are already given in the data set. Since, this experimental part was not included in the active learning study, binary classification scenario was eliminated from the current experiment. The initial idea of performing this experiment, aimed to see the ability of data points to group into natural clusters. For this reason, only multi-class label experiment was conducted, since all groups of nine synthetic attack types belong to anomaly traffic which is denoted as 1 in the binary label class.

To compare results the same three features from previous scenarios are applied to get the output results and three-dimensional scatter plot. Table 16 describes the results including Silhouette Coefficient for the DBSCAN. All calculations are estimated compared to the multi-class label. In an estimated number of clusters, noise is ignored, if present.

Table 16. DBSCAN results for given scenarios

Scenario	**Sceanrio 1**: Top three features (sttl, swin, ct_dst_sport_ltm) Eps: 0.46	**Scenario 2:** Top three features (ct_src_dport_ltm, ct_dst_sport_ltm, ct_srv_dst) Eps: 0.36	**Scenario 3:** Random three features (from top 19) (dpkts, rate, sttl) Eps: 0.49
Results	Estimated number of clusters: 8 Homogeneity: 0.266 Completeness: 0.331 V-measure: 0.295 Adjusted Rand Index: 0.164 Adjusted Mutual Information: 0.266	Estimated number of clusters: 9 Homogeneity: 0.003 Completeness: 0.160 V-measure: 0.006 Adjusted Rand Index: 0.001 Adjusted Mutual Information: 0.003	Estimated number of clusters: 10 Homogeneity: 0.196 Completeness: 0.311 V-measure: 0.240 Adjusted Rand Index: 0.029 Adjusted Mutual Information: 0.196
	Silhouette Coefficient: 0.464	**Silhouette Coefficient:** 0.518	**Silhouette Coefficient:** 0.231

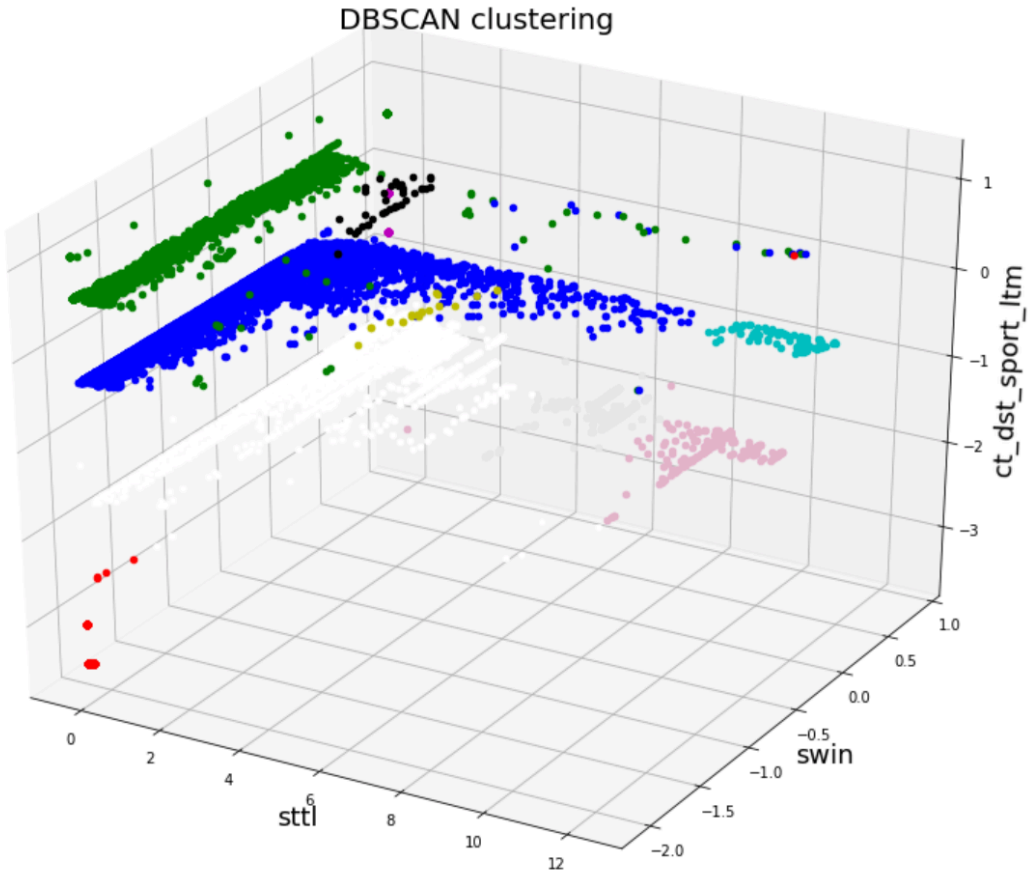


Figure 15. DBSCAN clustering results for Scenario 3

As can be seen from the scatter plot presented in Figure 15, data points separability is quite similar to the original layers. However, the exact label points cannot be identified easily with the help of clustering, which makes the algorithm less effective compared to the supervised, semi-supervised and active learning. Especially in the active learning querying the most informative data points for expert labeling purpose notably improves the detection of the classified labels. Additional Figures with plotted graphs for mentioned scenarios are provided in Appendix 4.

4.4 Semi-supervised Learning

Semi-supervised learning deals with the limited labeled samples in the training data. The combination of supervised and unsupervised methods, semi-supervised learning provides almost the same or even better accuracy results considering the small number of labeled data and a large amount of unlabeled data.

Some of the semi-supervised methods described below:

In semi-supervised classification, several approaches described below have been proposed in addressing the problem:

- *Label Propagation* algorithm assigns labels to previously unlabeled samples based on the labels that the neighboring nodes possess. Obtained labels are propagated to the unlabeled points throughout the algorithm. The disadvantage is that it produces no unique solution, but an aggregate of many solutions [57].

- The *Co-training* algorithm requires two views of the data to provide different, complementary information about the instance by assuming each described example using two different feature sets. For each view, co-training learns a separate classifier by using any labeled instances. Later, the most confident predictions of each classifier on the unlabeled data were used to construct additional labeled training data iteratively [58]. Co-training works only if one of the classifiers correctly labels a chunk of data that the other classifier previously misclassified, which doesn't fit in the logic being performed in the current thesis.

- *Pseudo-Labelling* also referred to as *Self-training* builds a model on the labeled data to estimate labels for the unlabeled pool. Later, the model re-build on the

"pseudo-labeled" unlabeled data and the labeled data. The process can be repeated if necessary. This setting, effectively practicing Entropy Regularization (encourages the classifier to make confident predictions on unlabeled data).

Among all possible methods, Pseudo Labeling [59] was defined as a simple and efficient method to do semi-supervised learning. This method combines almost all neural network and other training models to get Pseudo-Label. The unlabeled data used to predict the pseudo-labels as an output. Performed predictions don't provide entirely correct pseudo-labels. However, they give quite accurate labels, which can be seen from the training results of fully supervised learning validations metrics. Pseudo-Label for each unlabeled sample was picked from the class with the highest predicted probability according to the training model. Then, Pseudo-Labels were identified as the target class for unlabeled data as if they were correct labels.

The algorithm was re-trained in the supervised learning mode with the obtained results in a combination of the real label class. Several pseudo-label samples can be tuned, and validation metrics compared with the entire right labeled samples training, mix according to the predefined ration and fully pseudo-labeled samples training.

4.5 Active learning

"Active Learning is a special case of Machine Learning in which a learning algorithm is able to interactively query the user to obtain the desired outputs at new data points." [60]

The main idea of active learning is to provide better performance with less training data while being allowed to choose data from which it learns. Active learning is also called "query learning" where the learner queries the sample from the unlabeled data pool for labeling purpose. Each queried sample is labeled by the "oracle" (human expert) to solve the assigned task and train the defined model. This approach is well adopted in data mining, artificial intelligence and other modern machine learning algorithms and techniques where labeled data is not readily available, expensive or time-consuming to obtain.

In different scenarios, getting few labels such as defining spam email by marking it, may improve the filtering in applications to avoid unwanted emails. Every flag or mark is considered as a label for the data, which promotes the screening of daily used tools and

software. However, it is not always easy to obtain malicious or anomalous network traffic labels, due to its vague nature and continuous reconstruction.

4.5.1 Active Learning Scenarios

Active learning provides several scenarios, where the active learner requests the labels of the instances by querying the data. The main three methods that are considered in active learning literature are:

- *Membership Query Synthesis*: when learner generates or constructs an instance and, this generated sample is sent to the oracle for labeling. See Figure 16 for the illustration of this scenario.

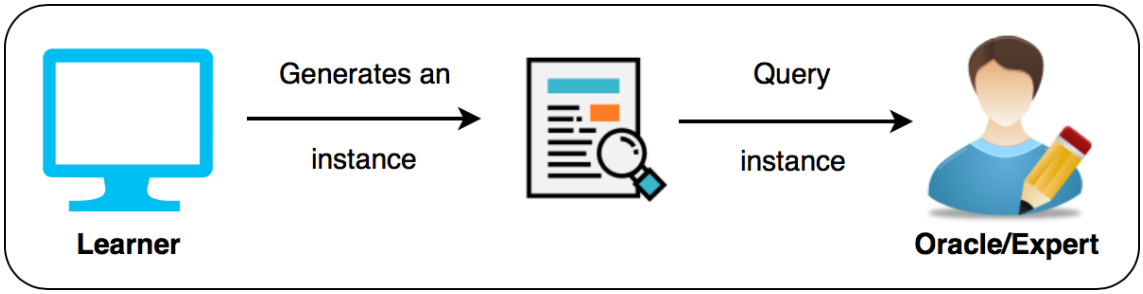


Figure 16. Membership Query Method

- *Stream-Based Selective Sampling*: when each unlabeled instance is considered separately, based on the assumptions that obtaining an unlabeled sample is free. This allows the learner to determine whether every single query coming one at a time needs to be labeled or discarded based on its informativeness. The informativeness of the instance identified with the query strategy is described in the section 4.5.2. For illustration see Figure 17.

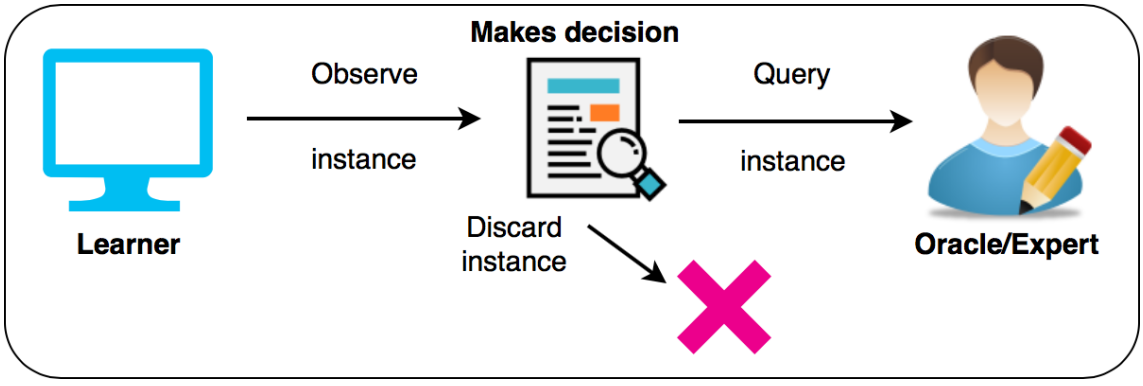


Figure 17. Stream-Based Selection Method

- *Pool-Based sampling*: this scenario assumes that chosen samples are from a large pool of unlabeled data, for the labeling purpose. Instances are then selected from the pool according to the predefined informativeness measure. This measure is applied to all samples in the pool and most informative ones are selected. Following this scenario, all unlabeled samples are ranked and the best (most educational) instances chosen for the label query. This is the most common scenario used in active learning literature. In this case, queries are not discarded or rejected compared to the previous example. For illustration see Figure 18.

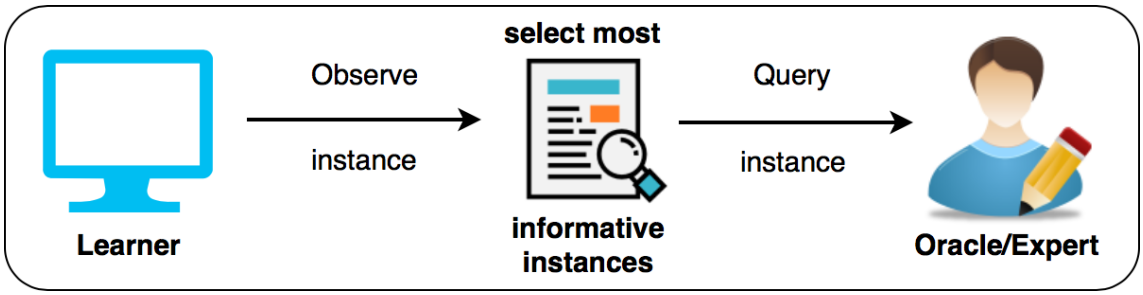


Figure 18. Pool-based sampling scenario

As it can be seen from different scenarios, querying the instances requires informativeness measures of the unlabeled instances. This is the crucial difference between active and passive learning process. In passive learning, the set of instances or the label class are fixed, whereas in active learning it can be decided.

Learning process usually starts with a small number of labeled instances in the training set. In the pool-based active learning cycle, learning algorithm requests for labels for one or more carefully selected instances and then learns from the results of the query. This knowledge then leveraged to select which instances to query next. Alternative ways to query label class, such as *Membership Query Sampling* and *Stream-Based Selective Sampling,* were briefly explained above (see section 4.5.1).

Usually there are no additional assumptions on the part of the learning algorithm once a query has been made. The new labeled instance is simply added to the labeled set, from which the learning proceeds in a standard supervised manner.

Active learning can be also combined with semi-supervised learning to provide alternative way of solving the problem.

According to the Active Literature learning Survey by Burr Settles, pool-based scenario is referenced in many literatures, and is more common for classification problems studied in real-world problems with reasonable outcome [60]. For this reason, in this study, the pool-based scenario is adopted along with several query strategies.

4.5.2 Query Strategy

All active learning scenarios involve evaluating the informativeness of unlabeled instances, which can either be generated or sampled from a given distribution. There are some proposed ways of formulating such query strategies in the literature. An overview for the most known ones is given below:

- *Random sampling*: the data points or samples are randomly selected from the validation set

- *Uncertainty sampling*: the algorithm selects the most uncertain class instances to the label. Different resources also call this method as *Least Confidence* (least confidence in its most likely label)

- *Margin Sampling*: in this scenario, the algorithm selects the instances with the smallest difference between the first and second probable label class.

- *Entropy Sampling*: in this case entropy formula is applied to each instance, and instance with the highest entropy is selected. Entropy formula is given below, where p_i is frequentist probability of an element/class 'i' in the data:

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i$$

Uncertainty sampling is less effective in complex structured instances, such as sequences and trees [61]. Since this work focuses on evaluating Decision Tree, Random Forest and XGBoost, then uncertainty sampling was not considered in the active learning experimental part of this study. All other sampling techniques such as *Random sampling, Margin Sampling* and *Entropy Sampling* are included in the practical experiment under Pool-based scenario.

Furthermore, software companies and large-scale research projects such as CiteSeer, Google, IBM, Microsoft, and Siemens are increasingly using active learning technologies in a variety of real-world applications [60]. However, published results in botnet detection and industry adoption are not sufficient to indicate how active learning methods are used in practice.

In most active learning works, there is an assumption that the quality of labeled data is high. However, when labels come from an empirical experiment (e.g., in biological, chemical, or clinical studies), noise resulted from the instrumentation of experimental setting is mostly expected. Even when labels acquired from human experts, variability in the quality of their annotations might be introduced, for several reasons:

- When instances are implicitly difficult for people and machines
- Distraction
- Misclassification due to fatigue

For this reason, annotations obtained from machines and experts need to "average" different level of noise and show that both true instance labels and individual oracle qualities that can be estimated. Subsequent iterations of active learning can be improved, where these estimates considered to request for only reliable annotators.

There are still many open research questions in the context of these lines. They can be only addressed if more studies will be conducted in active learning including true label and noisy oracle scenarios.

For this reason, this master's thesis includes experiments with true label and different levels of noise in the label class queried from the expert. The study will help analyze the tolerable results, where few exceptions can be made. Besides, it will provide the required level of expertise for the annotator/oracle.

5 Botnet Detection: Model Implementation Analysis

This section is dedicated to present the testing results and analysis for supervised, semi-supervised and active learning algorithm implantations: both binary and multi-class classifiers included in the implantation stage. Testing results are evaluated and presented in comparison with the different models and algorithms applied.

5.1 Semi-supervised algorithm results for botnet detection

As it was stated above, semi-supervised learning is the part of machine learning where a small amount of labeled data with a large number of unlabeled data are used to train the models. Due to the lack of labeled data, high cost and time resources the input dataset for the machine learning algorithms can consist of many unlabeled data. Those datasets cannot be trained with the supervised learning algorithms which make the task harder to achieve. However, semi-supervised learning can train the model with the combination of unlabeled data.

As it was already described, Pseudo Labeling defined as a simple and efficient method to do semi-supervised learning while having less labeled instances, in this thesis Pseudo Labeling approach was utilized. Also, this method earned the second prize in the ICML 2013 Workshop in Challenges in Representation Learning: The Black Box Learning Challenge [59].

The main idea approached in this step is to train the model with the available number of labeled instances to predict the label class for unlabeled data, which are called pseudo-labels. Further, a combination of labeled data and newly obtained pseudo-labeled data were used as a new dataset input to train predefined three classification models. Figure 19 shows a brief explanation for this algorithm's architecture [62].

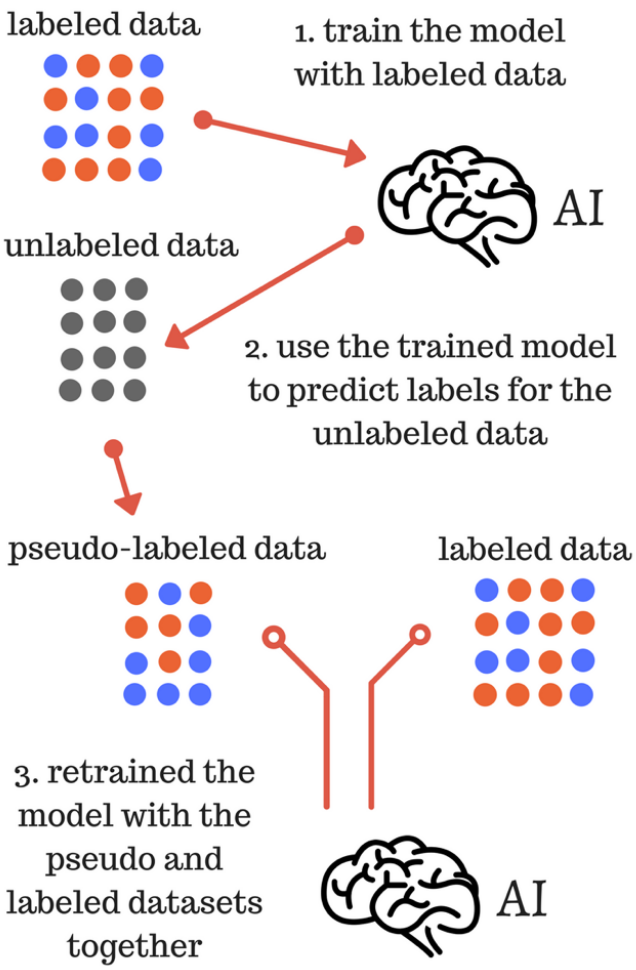


Figure 19. Pseudo Labelling Technique [62]

Usually, this method mentioned in the deep learning (online algorithms); however, in this thesis, it was approached with traditional machine learning classification models, namely XGBoost, Decision Tree and Random Forest. Those models were chosen from previous tests, due to higher performance results.

At this stage, the dataset with labeled instances including 82332 samples was split into training and for testing where unlabeled instances considered as 57632 instances. The ratio of labeled to unlabeled data was 0.3:0.7. Performance evaluation explored on XGBoost, Decision Tree and Random Forest classification models achieved by the accuracy metric performance for each model, using 5-fold cross-validation technique.

The PseudoLabler class achieved Pseudo-Labels for 70% of unlabeled data, written in sklearn estimator. Naturally, it wraps a sklearn classification while passing the test along with the list of features and target column. In this class, a function that creates “augmented

training set" consist of pseudo-labeled and labeled data. Function takes such arguments like the name of the model, training set, testing set, data input, features (since 19 for binary and 21 for multi-class most discriminative features have been selected, in this part all models are trained and tested respectively with those top features) and the parameter sample_rate. The parameter sample_rate was designed to control the percentage ratio of the pseudo-labeled data mixed with the valid labeled data. If sample_rate is defined as 0.0, that means a model is being trained exclusively with the labeled data. And the same logic applies for sample_rate parameter holding the value of 0.5 where proportion for both labeled and pseudo-labeled data are kept the same. In both cases resulting models used all labeled data for training and testing purpose. Obtained data was used to fit the model and tested with true labels to compare results of training with pseudo-labels.

As an example, Table 17 and Table 18 provide the accuracy results for PseudoLable (for all selected models) compared to original labeled data, using 5-fold cross-validation technique. Visual representation for all sample_rate values in range between 0.0 and 1.0 with all three models are given in Figure 20 and Figure 21. But for the purpose of showing the difference between classification results using originally labeled and Pseudo-Labeled data, sample_rate was set to 0.3. As it can be seen from Tables 17-18 below, accuracy result for both scenarios is similar, with a very small change where PseudoLabel accuracy result are lower by around 1/10.

Table 17. Accuracy results for binary class

	Accuracy	
	Model Classifier	**PseudoLabel**
Decision Tree	94.2022	94.0969
Random Forest	95.6111	95.5382
XGBoost	93.8216	93.408

Table 18. Accuracy results for multi class

	Accuracy	
	Model Classifier	**PseudoLabeler**
Decision Tree	83.5742	83.2057
Random Forest	85.1898	85.3599
XGBoost	85.4084	84.8861

Performance of pseudo-labeling based on different sample rates and machine learning models, using 5-fold cross-validation technique is presented in Figure 20 and 21. The fluctuation of the accuracy can be seen in all models for 1-2% at a different ratio of the sample_rate. However, 5-fold cross validation provided better accuracy results for about 4%, compared to the same models evaluated with 10-fold cross-validation method in the previous stages.

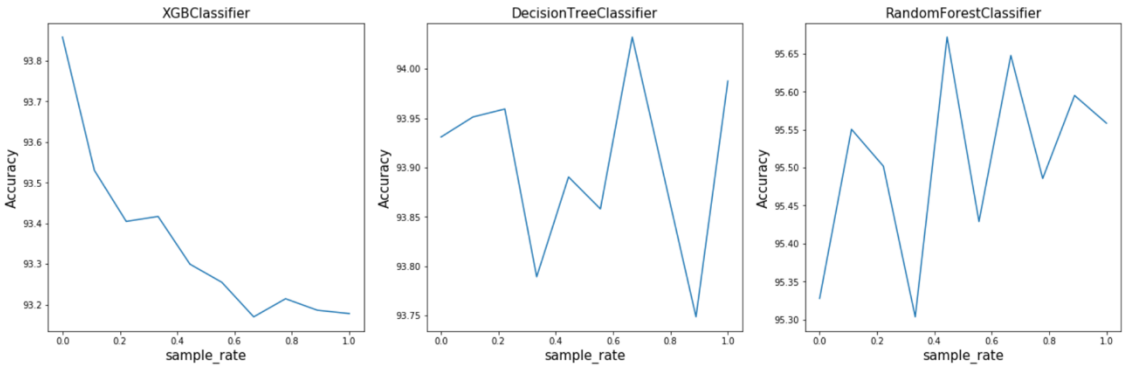


Figure 20. Semi-supervised learning results for binary label class

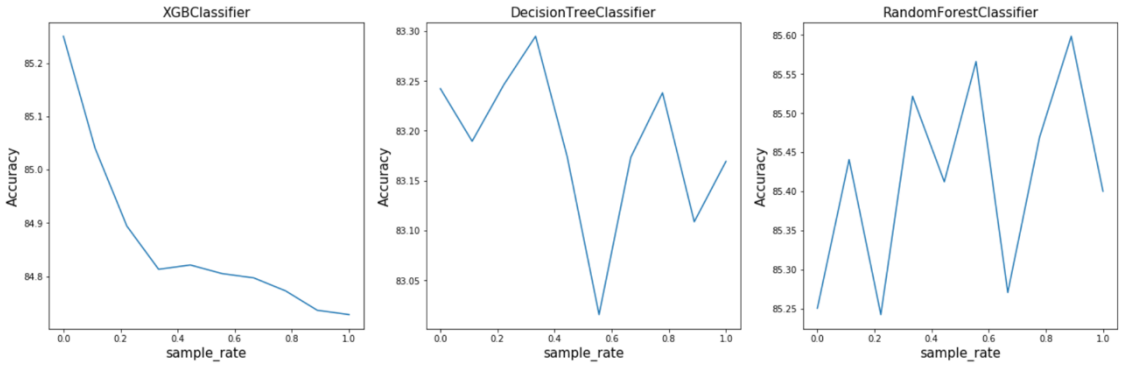


Figure 21. Semi-supervised learning results for multi label class

Appendix 5 includes the representation of each model separately in respect to different class type scenarios.

Pseudo-labeling is a powerful technique which allows utilizing large pool of unlabeled data while training the models. As can be analyzed from the results, this technique provided a slight performance boost/improvement while having a smaller number of labeled samples. Best model performance results in a two-class label can be noted in Random Forest. However, the performance of the other models was almost the same and competitive. Although all the models provide good accuracy results in both multi-class

and two-class label scenarios, XGBClassifer accuracy metric results are continuously decreasing in both scenarios while pseudo-labeled instances are increased.

5.2 Active learning botnet detection results

In this part, an experiment was designed in two stages. The first scenario assumes that the queried label was given correctly and in the second scenario; it was considered that expert might give wrong labeled samples, while the instance was queried. Nevertheless, the algorithm design was kept the same for both cases.

The algorithm for the active learning approach was designed in scikit-learn machine learning library for Python, using the pool-based sampling. The central part is described in the pseudo-algorithm below:

1. Initial data is divided into a pool and test-set
2. k samples in the set of numbers [10, 25, 50] are selected from the pool for the training and labeling purpose and remaining data is used for validation.
3. All defined sets (training, testing, and validation) are normalized using MinMax scaler in a given range (0, 1).
4. Base models are defined as XGBoost, Decision Tree and Random Forest Classifiers are trained with the training set with balanced weights.
5. Trained model was validated by the validation-set separated in step 2, to get the probabilities per sample.
6. Trained model is tested with testing-set to get the performance results.
7. k most informative samples are selected using one of the three methods in each experiment, based on each sample probabilities. (the most uncertain)
    - Random selection, k samples are randomly chosen from the validation set.
    - Entropy selection, k samples with the highest entropy selected from the validation set.
    - Margin selection, k samples with the lowest difference between the two highest class probabilities are selected from the validation set.
8. Selected k samples are moved from the validation set to the train set with queried label class. (since data set already have fully labeled instances, labeling part is processed automatically)
9. Normalization of all data sets are inversed

10. If all experiments are completed, the algorithm is completing work, otherwise continuing from step 3.

Things to note regarding the normalization steps (3 and 9):

- Normalization for all sets was inversed and normalized again after samples from the validation set was removed, because sample distribution changed in both the new validation and new train-sets.

The simple formula calculates a total number of the experiments, where the number of base models (3) multiplied to the number of selection methods (3) and several k set samples (3) to the number of desired repeats. In current work, a number of repeats are defined as 1 to see the initial results. Total 27 experiments are approached and results for each selection method and number of k samples are visualized in the graphs. Figures 41-46 are the representation of the accuracy results depending on the base model and different sampling scenarios separated by different colors.

The critical area of applying active learning method is influenced by the sample selection method, which is described in step 7. There are literature such as [60], [63], [64] where query selection strategies such as Random selection, Entropy selection and Margin selection are discussed. For this reason, this study included them to compare the results and analyze the benefit of using them in the data set and models assigned for the experiment.

The Training Model accepts one of the selection learning algorithms with predefined Base Model (XGBoost, Decision Tree, Random Forest). Model is trained using the training set and evaluation performance metrics are calculated using testing set.

5.2.1 Queried Labels are True

Initially, data of 82332 samples were split into 3% for the training set with 2469 instances and rest 97% for the testing set. This scenario of using only a few labeled instances is adapted to be closer to real-life enterprise situations. Later the train set was split to train and validation sets. For the purpose of acquiring label class marked by oracle, a total number of 500 instances was queried with the predefined methods described above. As it was already stated, 19 top features were selected for the binary class problem and 21 best for the multi-class task. Upper bound for each model is identified by a straight blue line,

calculated using supervised learning with 2469 instances as training set and the rest number of instances for testing purpose. This scenario presents results for the classification, assuming that the expert has correctly labeled data points. The results are presented in graphical visualization for each selected model and label class type in Figures 22-27.

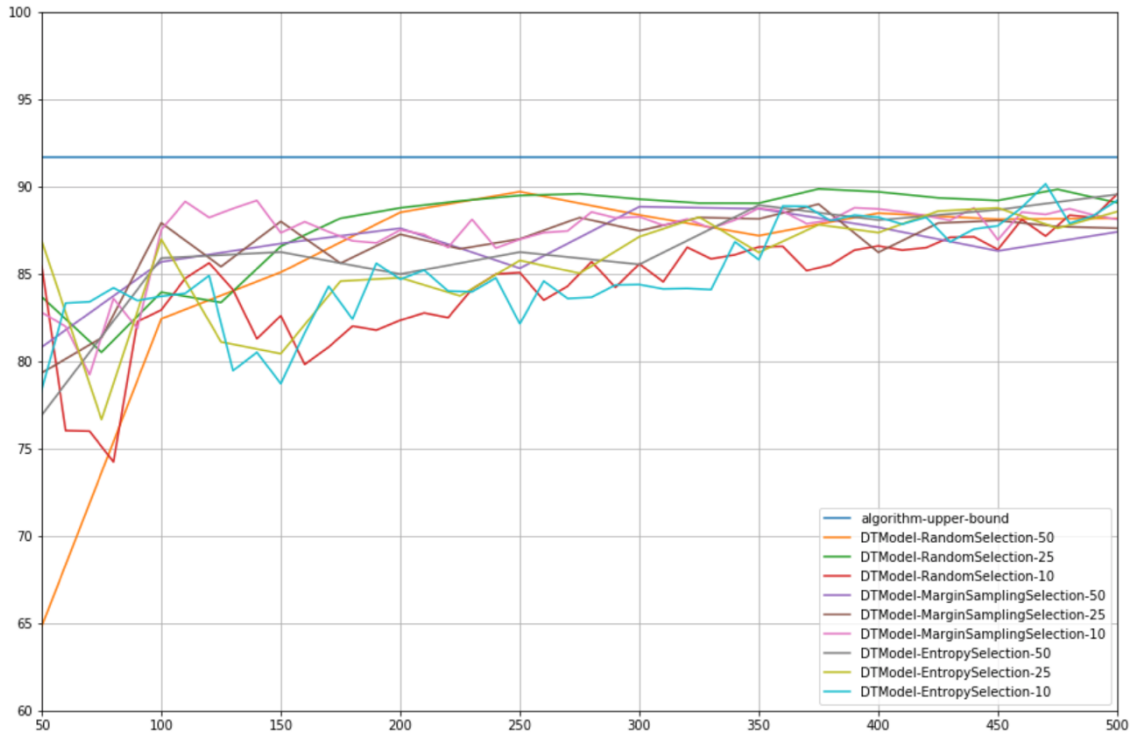


Figure 22. Binary class: Active learning performance results for Decision Tree

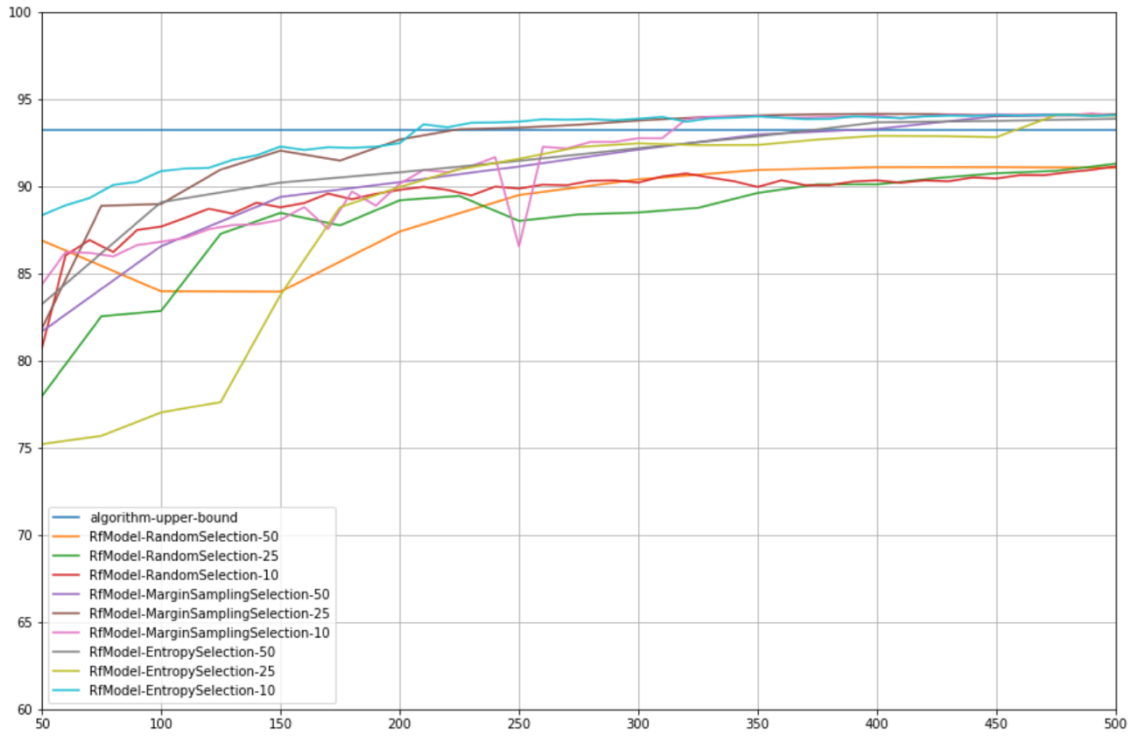


Figure 23. Binary class: Active learning performance results for Random Forest

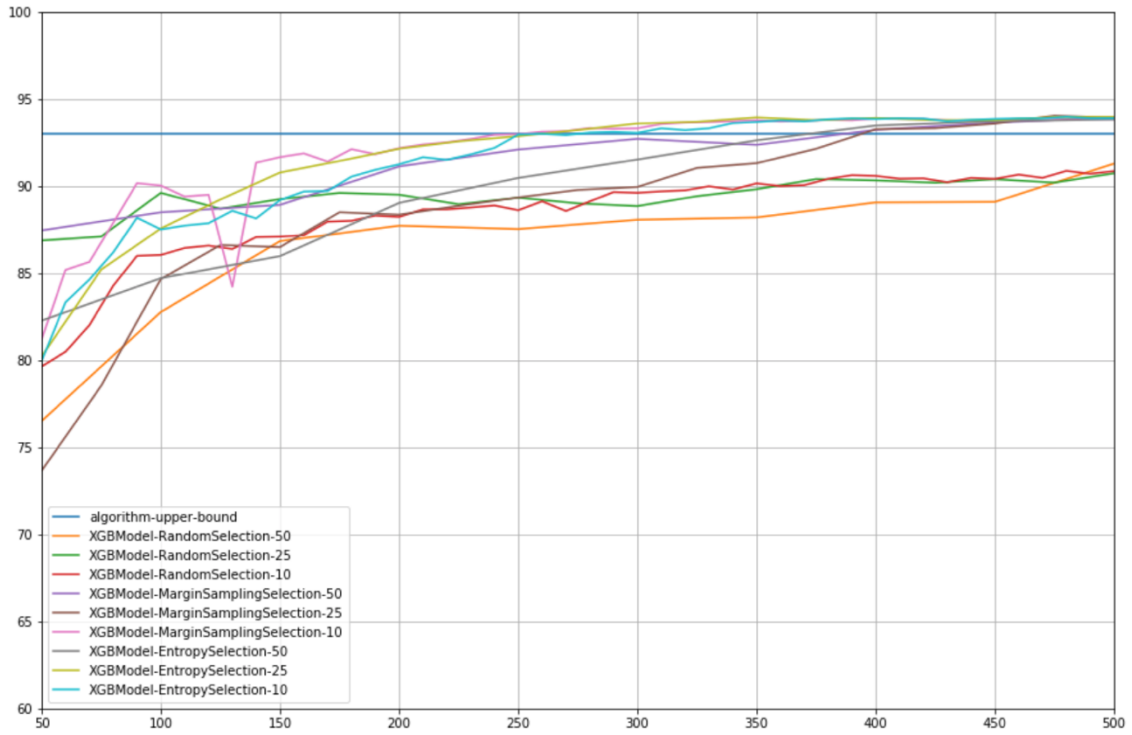


Figure 24. Binary class: Active learning performance results for XGBoost

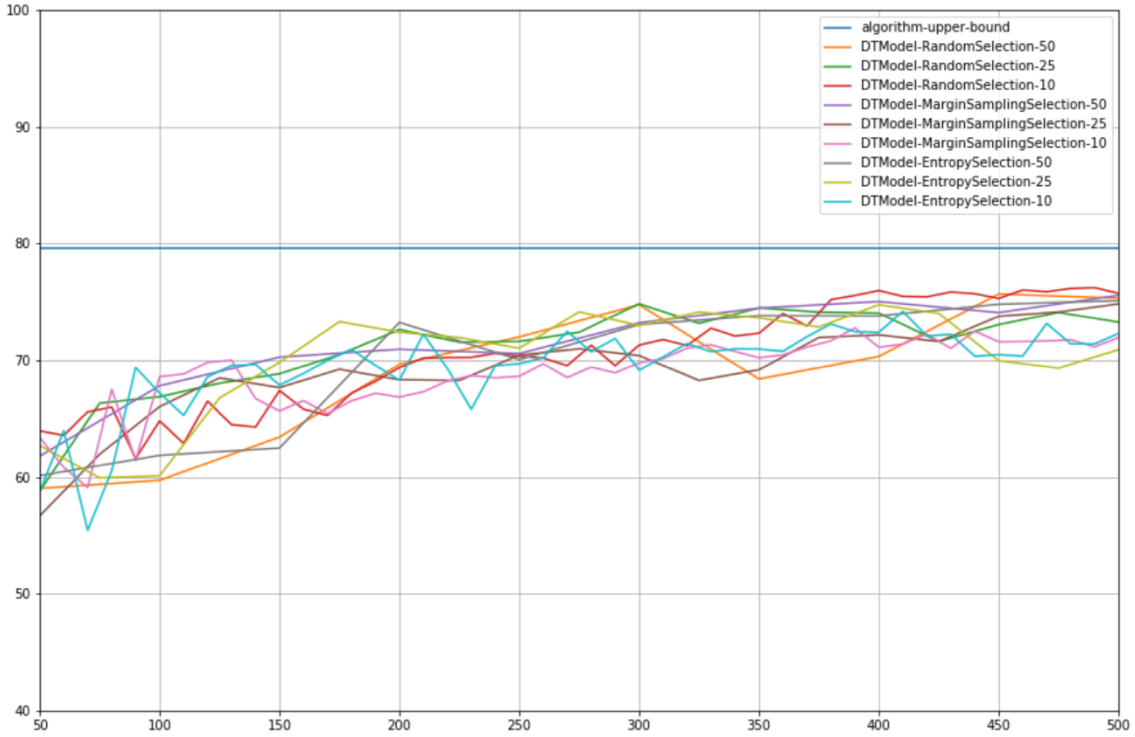


Figure 25. Multi class: Active learning performance results for Decision Tree

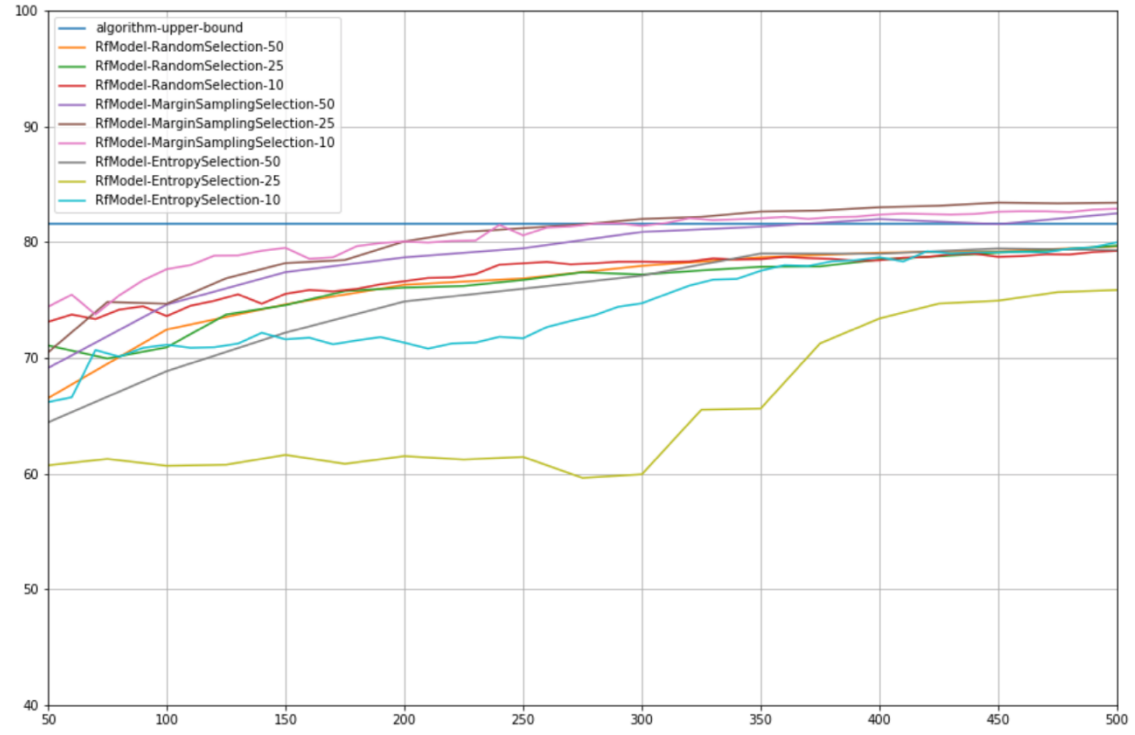


Figure 26. Multi class: Active learning performance results for Random Forest

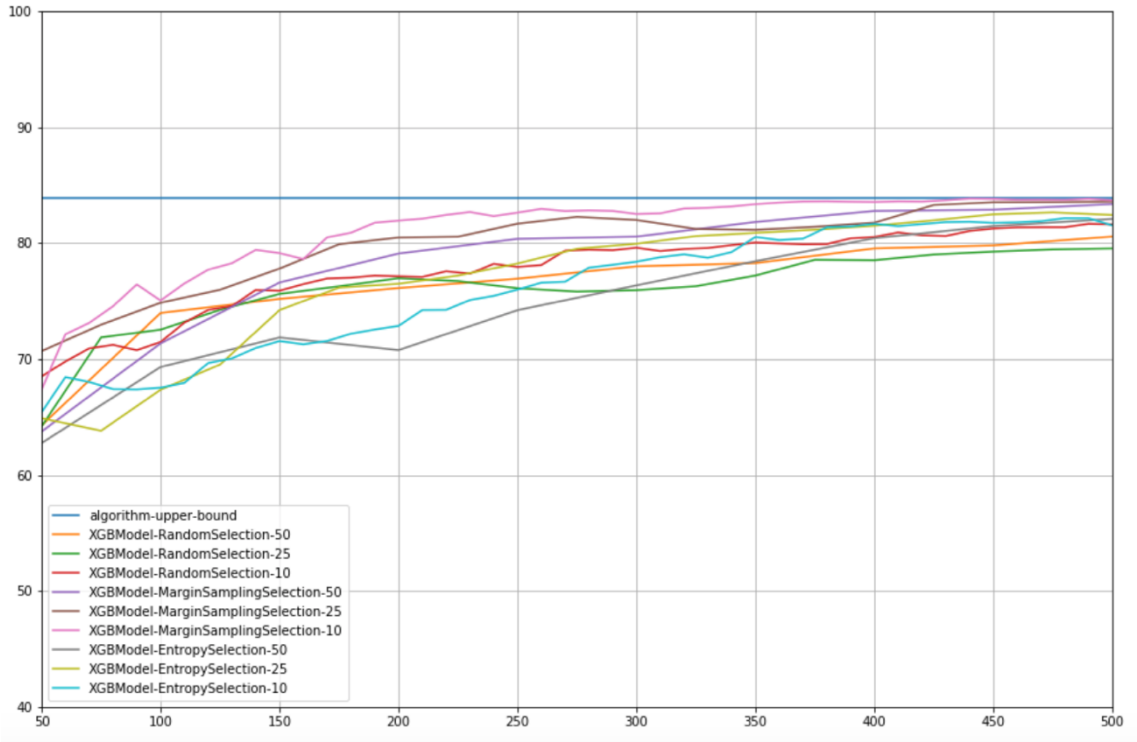


Figure 27. Multi class: Active learning performance results for XGBoost

On the basis of the above, it can be seen that XGBoost and Random Forest models performed better than Decision Tree both in binary and multi-class classification problem. Moreover, querying most informative 500 data points under active learning approach provided competitive results compared to the fully labeled (upper bound) accuracy results trained on 2469 instances.

In the binary class label scenario, the most significant and competitive samples selection methods for all models was Margin Selection with k=[10,25] instances and Entropy selection with k=10. Even though, Margin Selection gave most discriminative results both for binary class and multi class experiment scenarios, Random Selection contributed higher accuracy results in multi-class scenario for Decision Tree model. Nevertheless, Margin Selection method with k=[10,25,50] proved as the most competitive keeping better accuracy results for multi class label scenario. Also, it is concluded that, for all models in binary class label, the least competitive selection method was Random Selection with all tested numbers of k samples.

For the purpose of analysis and comparison between active learning and supervised learning, visual representation of Margin Selection and Entropy Selection methods for

binary class scenario was given in Figures 28-31. Since Decision Tree was the least confident model in this particular experiment, graphs are given only for XGBoost and Random Forest models.

Additionally, as can be concluded from Figure 25, in the multi-class label scenario, Decision Tree model did not reach upper bound results in the first iteration of label querying process, for this reason in Figures 32-33, the most competitive Margin Selection method with Random Forest and XGBoost models was presented to provide better visual representation of the multi class label learning process.

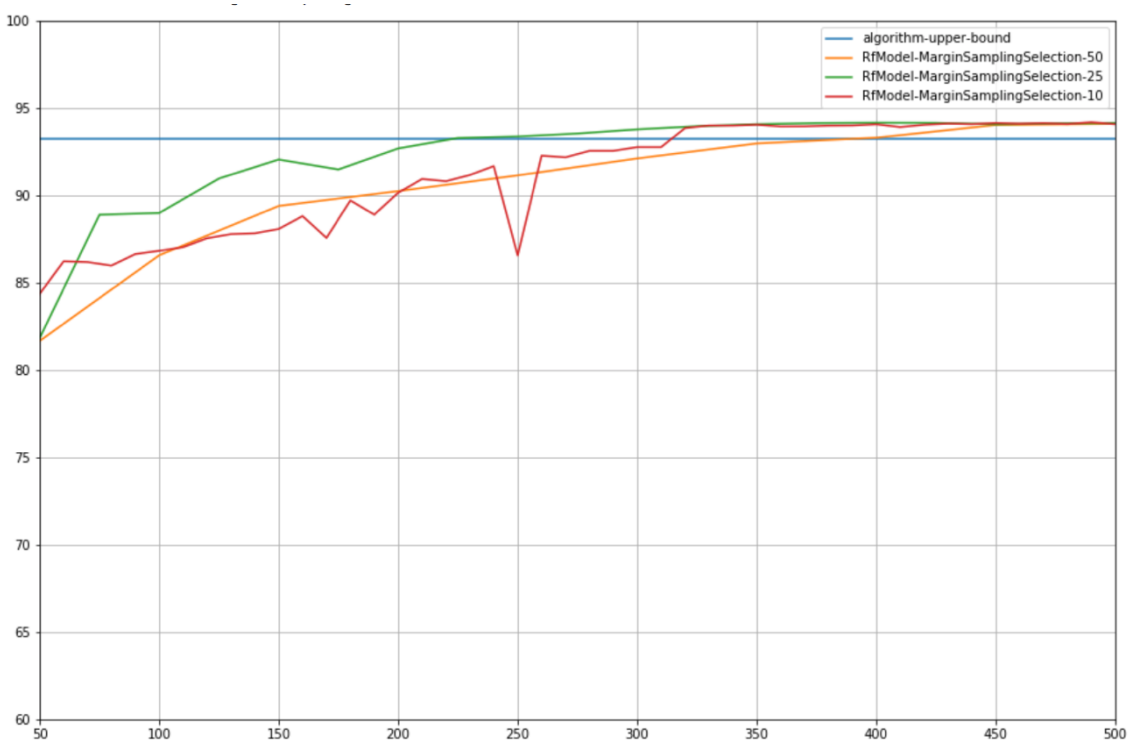


Figure 28. Binary class: Active learning performance results for Random Forest with Margin Sample Selection

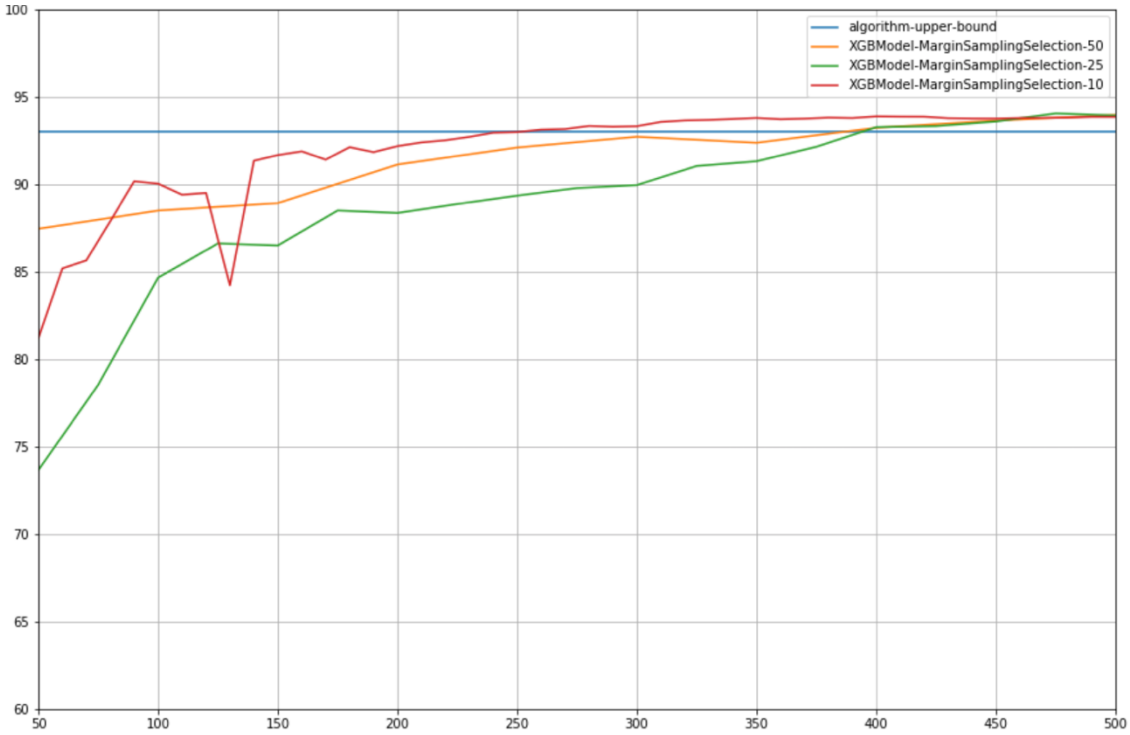


Figure 29. Binary class: Active learning performance results for XGBoost with Margin Sample Selection

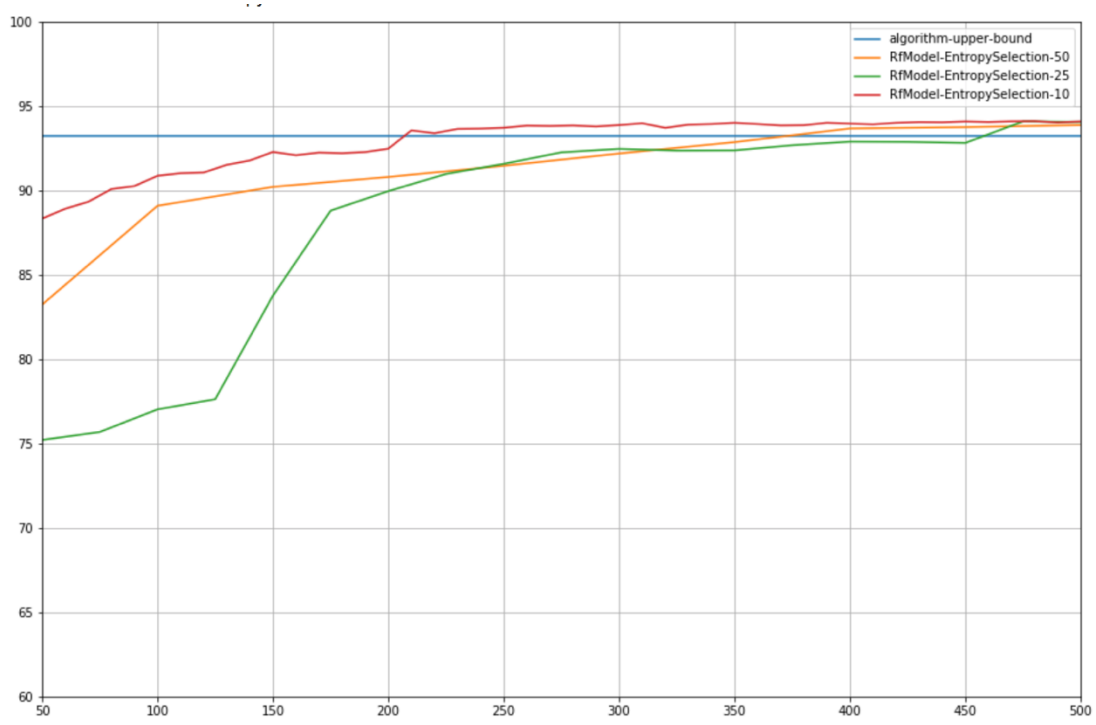


Figure 30. Binary class: Active learning performance results for Random Forest with Entropy Selection

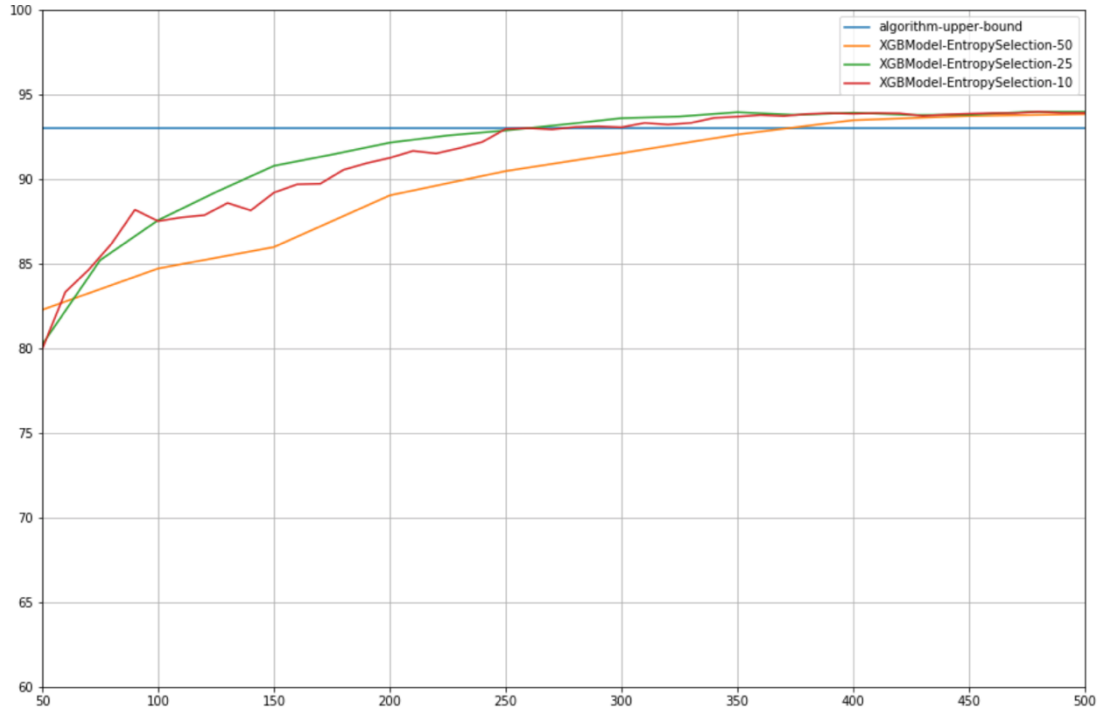


Figure 31. Binary class: Active learning performance results for XGBoost with Entropy Selection

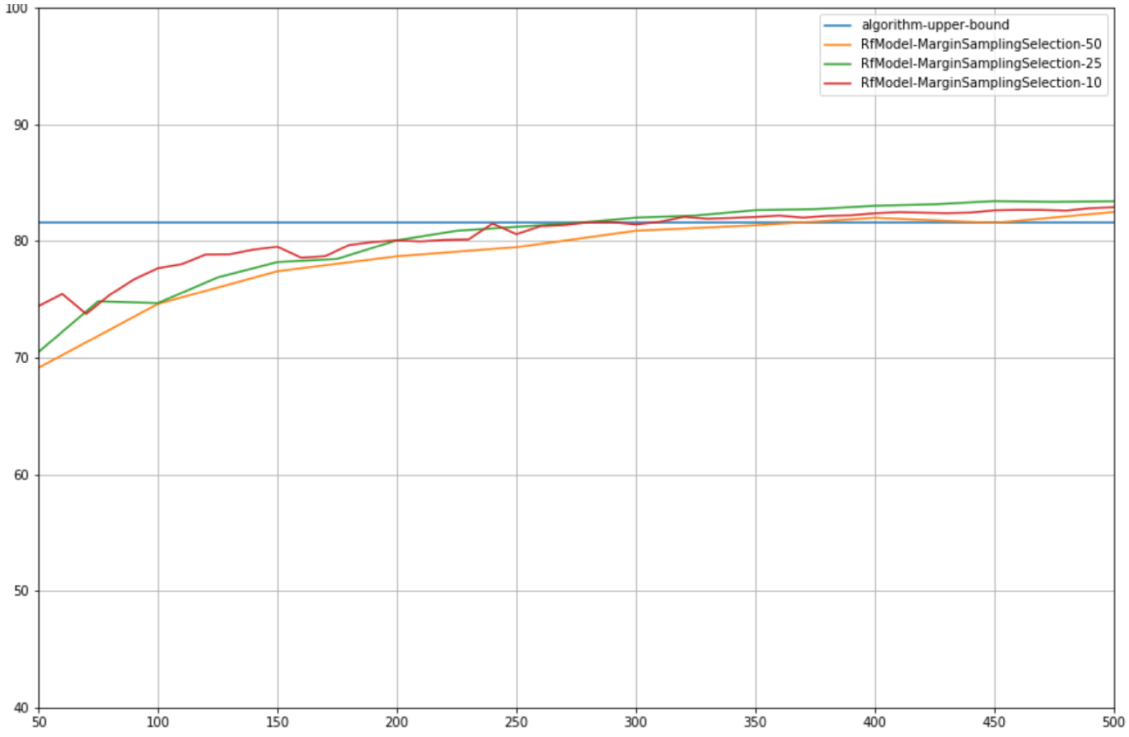


Figure 32. Multi class: Active learning performance results for Random Forest with Margin Sample Selection

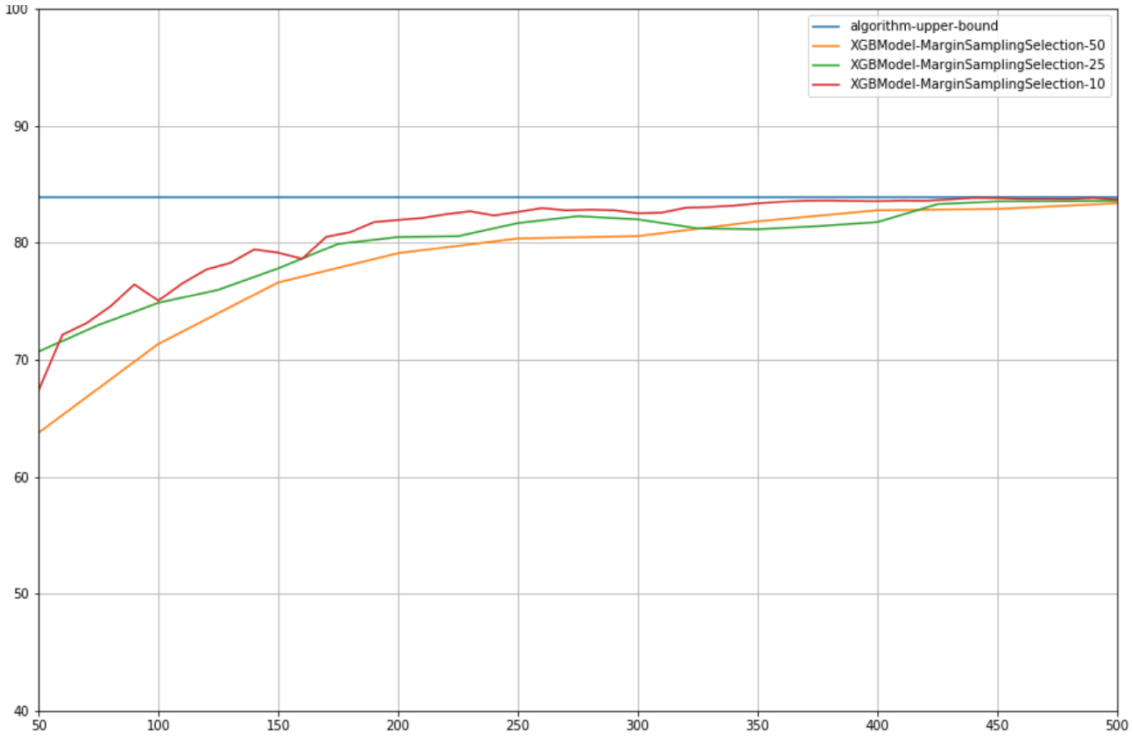


Figure 33. Multi class: Active learning performance results for Random Forest with Margin Sample Selection

As can be concluded from the above given figures, learning process from the most informative instances can provide higher accuracy results while having small pool of labeled data. In most of the cases, active learning accuracy results with 350 and 400 most informative labeled samples exceeded supervised learning accuracy results with 2469 labeled instances. Different data sets and models tend to have different most appropriate selection method and number of k queried samples. However, based on the experiment results, intuitively it can be concluded that it is easier and more appropriate for expert to label less number or queried samples with k=10 and k=25.

5.2.2 Queried Labels are Wrong

In real-case implementations, to get realistic outcome, labeling with misclassification should be considered. Complete results should take into account both true labeling and misclassification approaches. For this reason, it was assumed that the obtained label class has wrong labels both for binary and multi-class problem statements. Different experiments were conducted based on the assumption if the queried instances labeled with 10%, 20%, 30%, 40% and 50% wrong label class (noise). All the results obtained from this experiment are visualized in graphs and are attached in Appendix 6.

Classification accuracy results with accurate and wrong label proportions are shown in Tables 19-20. This information is provided to visualize the difference in supervised learning between different wrong acquired label class and accurate labeled classification scenario.

Table 19. Binary class: Supervised learning results for accurate and wrong labels

	Accurate Labels	**10% wrong Labels**	**20% wrong Labels**	**30% wrong Labels**	**40% wrong Labels**	**50% wrong Labels**
XGBoost	92.74	91.94	91.00	85.24	55.85	53.33
Decision Tree	91.80	82.64	72.71	61.25	57.19	47.63
Random Forest	93.42	90.37	84.43	70.86	59.11	49.62

Next scenario shown in Table 20, was performed on the multi-class problem with one normal traffic and nine attack type labels.

Table 20. Multi class: Supervised learning results for right label and wrong label

	Accurate Labels	**10% wrong Labels**	**20% wrong Labels**	**30% wrong Labels**	**40% wrong Labels**	**50% wrong Labels**
XGBoost	84.1	82.57	80.14	73.79	60.03	42.16
Decision Tree	79.56	70.46	62.30	51.27	43.97	40.37
Random Forest	82.09	78.71	73.86	62.37	51.61	42.23

From Tables 19-20 presented above, it can be concluded that, testing results under supervised learning algorithm with 2469 trained instances tends to have different behavior of decreasing accuracy results. XGBoost model had tolerable accuracy decline in 10%, 20% and 30% from the classification with accurate label. However, Decision Tree model decreased accuracy results similarly to the percentage of the wrong labeled instances. In every scenario around 8-10% decrease in accuracy was noted for Decision

Tree model. Also, it was noted that acquiring 50% wrong labeled instances leads to decrease accuracy results for the same amount. And results after 30% wrong labeled instances are less tolerable for all models.

According to the preliminary analysis, Decision Tree model has more fluctuation and unstable results. But, Random Forest and XGBoost models provided similar classification performance and even tolerable results with 10% and 20% inaccurate label class. Since there were not much difference between two models, Random Forest model selected to provide mode detailed information regarding the classification performance results for binary class label scenario, whereas XGBoost for multi-class label scenario.

Taking into account that wrong labeled instances provided for the training stage, Random Forest with Entropy selection method visualized in the graph to compare the results obtained from all scenarios conducted for the experiment in this section. Entropy Selection with k=10 samples were selected, due to the better performance compared to other query selection methods.

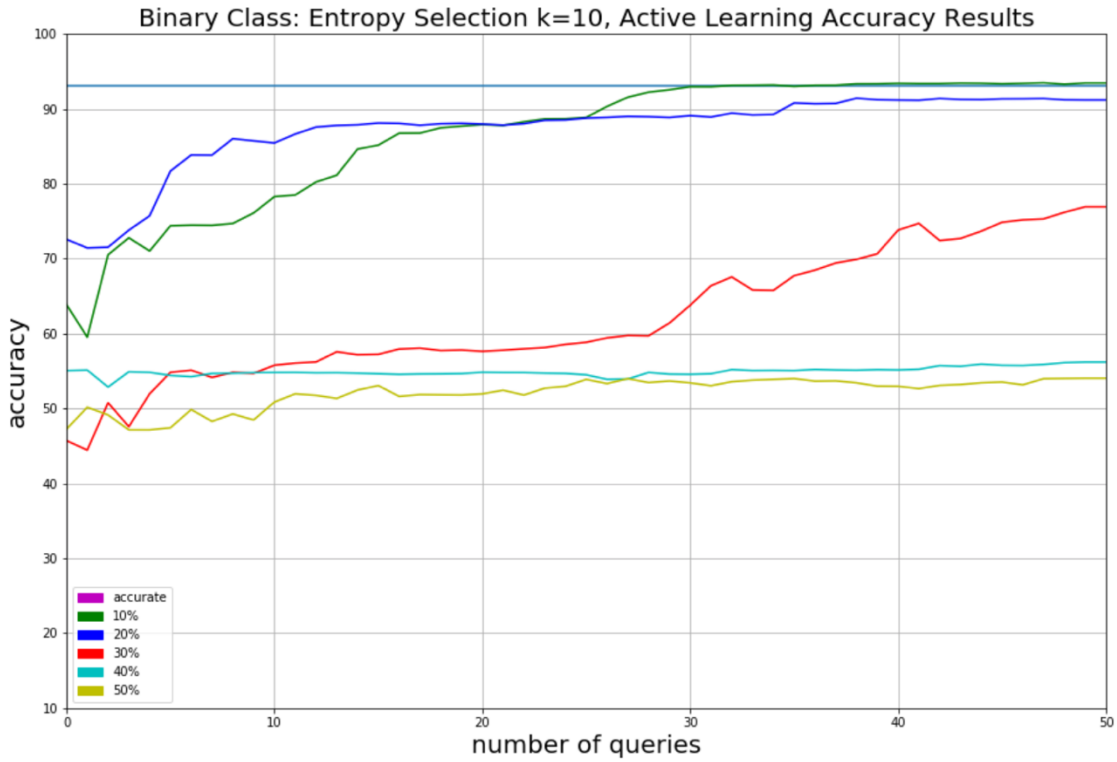


Figure 34. Binary Class: Active learning accuracy results for Random Forest with Entropy Selection k=10

In Figure 34 x-axis represents the total number of queries using the Entropy selection method with k=10 samples. In this case, total number of queries instances equals to 500.

Analysis of the graph provided reveals that 10% and 20% wrong labeled instances can be tolerated. This assumption was based on the accuracy performance results given for Random Forest with Entropy sample selection method. However, 40% and 50% inaccurate label class tend to keep low accuracy results similar to supervised learning results shown in Table 19. In addition, it can be seen that 30% wrong labeled instances still tend to increase the accuracy results when number of queried instances for testing purpose are increased.

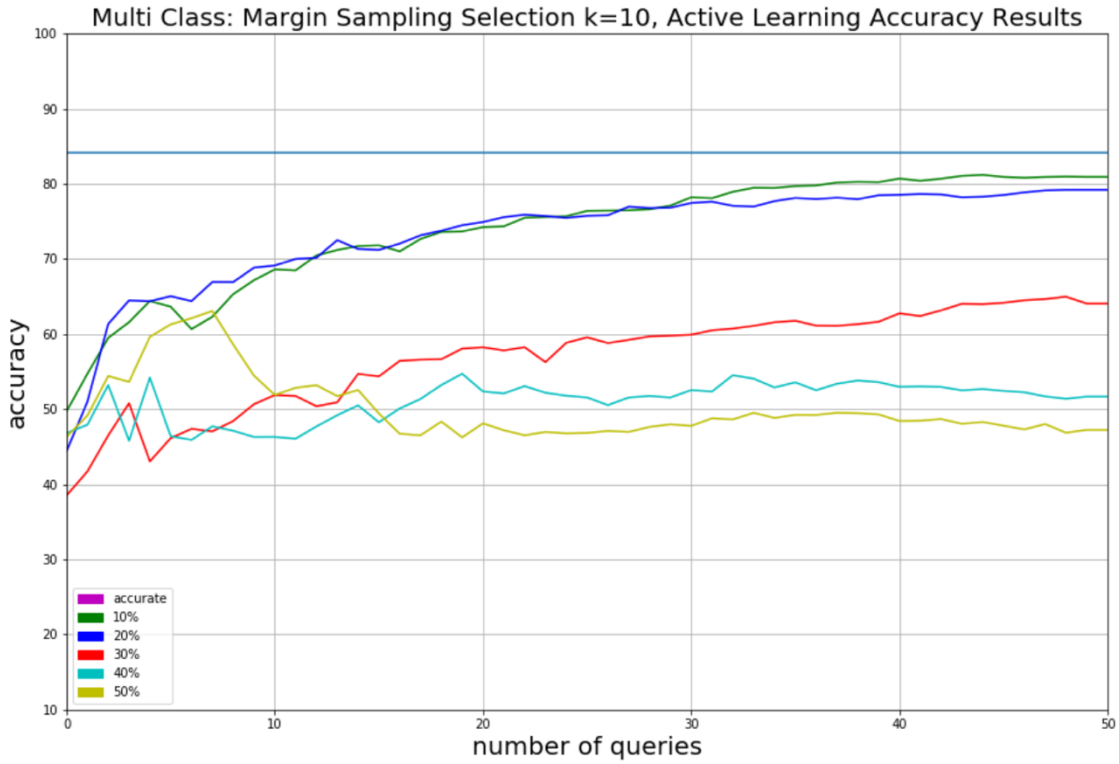


Figure 35. Multi Class: Active learning accuracy results for XGBoost with Margin Sampling Selection k=10

As presented in the graph above, for multi-class label classification problem, performance reduction equals to the wrong label acquisition percentage. Comparatively to the supervised learning upper bound overall reduction cost varies from 30% to 40%. However, the results are similar to the binary class scenario, where 10% and 20% wrong labeled instances can be tolerated, while 40% and 50% are not likely for network intrusion detection systems. Also, similar behaviors noted from the case where inaccurate labeled

instances reach 30%, where accuracy still tend to increase and learn from accurate label class.

All models and sample selection methods used in this particular experimental setup are shown in Appendix 6. Each selection method was denoted with specific color and was named accordingly. The most significant reduction cost in accuracy results can be seen from Decision Tree, whereas some scenario cases can be tolerated for XGBoost and Random Forest models. However, unlike the scenario where all queried samples are obtained with accurate label class, in this experiment it can be noted that most appropriate selection method varies from depending on the model and case with the wrong label percentage.

According to the results provided in this section, it can be concluded that active learning algorithms can drastically improve the detection problems while having small pool of labeled instances and large pool of unlabeled instances. Learning process from the most informative samples provides higher accuracy (and other performance metrics, see Appendix 7) results in identifying the normal traffic and attack types. Nevertheless, experiments where models were trained with wrong labeled instances proved that small mistakes (10%-20%) in the label class can be tolerated, since the accuracy results did not have significant changes. However, obtaining the 40% to 50% wrong label class cost same amount of accuracy reduction compared to the right class, which makes the detection systems lousier in production usage. This also concludes that proficiency level of the expert/oracle should be higher than average with minimal cost of making mistakes in providing label class for each instance.

6 Conclusion

Active learning is a growing area of research in machine learning. Active learning provided a lot of evidence that the number of labeled instances necessary to train accurate models can be dramatically reduced in a variety of applications. Taking advantage of these foundations, different cyber-attack and malicious network traffic can be detected by applying the knowledge derived from the active learning concepts. This research aimed at using active learning methods in practice on botnet detection. Results of the study introduced many essential problem variants and practical concerns.

In this thesis, the application of classification models for active learning was investigated concerning the binary and multi-class label for network botnet detection. Different querying methods are applied and studied to identify their effect on the classification of normal and malicious network traffic. The learning process with most informative data points was considered and concluded as an effective way for botnet detection performance. Feature selection methods provided a reduced and the most discriminative number of features for botnet detection, which is used to train and test classification models along with the active learning methods. Analysis of feature selection states that best predictive features belong to the connection features in the network traffic.

In the active learning section, pool-based sampling method along with the three querying scenarios was implemented and analyzed. Analysis of each separate way provided different results where Margin Sample Selection scenario with the XGB and Random Forest classification model executed the most competitive results. Learning process states that from a labeled and unlabeled pool of instances, most informative samples can be chosen and increase the prediction accuracy than classical supervised algorithms. Obtained results are an essential addition for the research gap in the active machine learning methodology, for the botnet detection in the network intrusion detection systems.

More specifically, results show that if a queried sample was correctly labeled, training of the model provides better results than incorrectly labeled data in detecting both normal traffic and attack types within a malicious set. Moreover, classification of normal and

botnet traffic provided tolerable performance results, where acquired sample had 10% to 20% of inaccurate label class. Different experiments with wrongly labeled instances are conducted and concluded, to predict the reduction cost for the network intrusion systems. Based on the results, suggestions on the proficiency of expert/oracle were given.

The critical point in utilizing active learning for botnet detection, allowing to minimize the cost of obtaining the label class, while learning on the accurately chosen ones from the whole pool. This fact may lead to improve network intrusion detection systems and build accurate classifiers that focus on main predictors. However, the procedure for different datasets and desired detection may vary. This research shows that depending on detection objectives and requirements, different datasets might require investigation of several query methods and different models, in order to accomplish experimental results with optimal botnet detection performance.

This work can be extended considering combining learning algorithms. By following the results and analysis presented in this study, semi-supervised pseudo labeling can be applied to the unlabeled pool, where label class of the instance obtained from the active learning phase. In addition, the clustering technique can be studied more broadly, in order to identify the areas of its application in the active learning scenarios.

References

[1] Sergio SC Silva, Rodrigo MP Silva, Raquel CG Pinto and Ronaldo M Salles, "Botnets: A survey," *Computer Networks,* vol. Volume 57, no. Issue 2, pp. 378-40, 4 February 2013.

[2] I. Ilascu, "BitdefenderBOX," 9 October 2018. [Online]. Available: https://bit.ly/2KDuvZw.

[3] Guofei Gu, Roberto Perdisci, Junjie Zhang and Wenke Lee, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," *The 17th USEN1X Security Symposium,* pp. 139-154, July 2008.

[4] J. Goebel and T. Holz, "ishi: Identify bot contaminated hosts by IRC nickname evaluation," in *Proceedings of USENIX HotBots'07*, 2007.

[5] "The UNSW-NB15 Dataset Description," Australian Centre for Cyber Security (ACCS), 2015. [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/?fbclid=IwAR3LKJNjf34sCAP-bAs99W_64jITks04KBdzNkX0iZlrbHbe2YUawwOMc9Y.

[6] Vaibhav Nivargi, Mayukh Bhaowal and Teddy Lee, "Machine Learning Based Botnet Detection," Stanford.

[7] Xuan Dan Hoang and Quynh Chi Nguyen, "Botnet Detection Based On Machine Learning Techniques Using DNS Query Data," *Future Internet,* 18 May 2018.

[8] García, A. Zunino and M. Campo, "Survey on network-based botnet detection methods," *Security and Communication Networks,* vol. 7, no. 5, pp. 878-903, May 2014 .

[9] Bahşi Hayretdin and Nõmm Sven, "Unsupervised anomaly based botnet detection in IoT networks," in *17th IEEE International Conference on Machine Learning and Applications*, Orlando, Florida, USA, 17-20 December 2018.

[10] Zhicong Qiu, David J. Miller and George Kesidi, "Flow-Based Botnet Detection through Semi-supervised Active Learning," CSE Dept, PSU, Technical Report CSE-16-010, September 13, 2016.

[11] Zhicong Qiu, David J. Miller and George Kesidis, "Flow based botnet detection through semi-supervised active learning," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, 5-9 March 2017.

[12] Khalid Huseynov, Kwangjo Kim and Paul D. Yoo, "Semi-supervised Botnet Detection Using Ant Colony Clustering," in *The 31th Symposium on Cryptography and Information Security*, Kagoshima, Japan, Janury 21-24, 2014.

[13] C. M. Jane, "Atlanta Business Chronicle," 22 July 2002. [Online]. Available: http://bit.ly/2Dp95gi.

[14] A. Zaharia, "HEIMDAL Security," 25 April 2016. [Online]. Available: http://bit.ly/2VZKXbs.

[15] D. Etherington and K. Conger, "TechCrunch," 2016. [Online]. Available: https://tcrn.ch/2Gy9S0k.

[16] T. I. Team, "Avast," 25 October 2018. [Online]. Available: http://bit.ly/2XwAeFN.

[17] M. Weinberger, "Business Insider," 22 April 2015. [Online]. Available: http://bit.ly/2UTkKyZ.

[18] V. Paxson, "LBNL/ICSI Enterprise Tracing Project," Lawrence Berkeley National Laboratory and ICSI, 2005. [Online]. Available: https://www.icir.org/enterprise-tracing/download.html.

[19] "VRT Labs - Zeus Trojan Analysis," [Online]. Available: https://labs.snort.org/papers/zeus.html.

[20] "ISOT Dataset Overview," [Online]. Available: https://www.uvic.ca/engineering/ece/isot/assets/docs/isot-datase.pdf.

[21] Wei Li, Marco Canini, Andrew W Moore and Raffaele Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Computer Networks: The International Journal of Computer and Telecommunications Networking,* vol. 53, no. 6, pp. 790-809 , 2009.

[22] Andrew Moore, Denis Zuev and Michael Crogan, "Discriminators for use in flow-based classification," Queen Mary: University of London , London, 2005.

[23] Z Berkay Celik, Jayaram Raghuram, George Kesidis and David J Miller, "Salting public traces with attack traffic to test flow classifiers," in *Proceedings of the 4th conference on Cyber security experimentation and test*, 2011.

[24] Kai Yang, Jie Ren, Yanqiao Zhu and Weiyi Zhang, "Active Learning for Wireless IoT Intrusion Detection," *IEEE Wireless Communications,* vol. 25, no. 6, pp. 19 - 25, December 2018.

[25] "XGBoost Documentation," [Online]. Available: https://xgboost.readthedocs.io/en/latest/.

[26] L. Bull, N. Dervilis, K. Worden and G. Manson, "Active learning for semi-supervised structural health monitoring," *Journal of Biomedical Informatics,* vol. 64, pp. 168-178, December 2016.

[27] Moheeb Abu, Jay Zarfoss, Fabian Monrosereas and Andreas Terzis, "A Multifaceted approach to understanding the botnet phenonmenon," 2006.

[28] Y. Chen, "Towards wireless overlay network architectures".

[29] G. Kaur, " Novel Distributed Machine Learning Framework for Semi-Supervised Detection of Botnet Attacks," in *Eleventh International Conference on Contemporary Computing (IC3)*, August 2018.

[30] "Botnet dataset," Canadian Institute for Cybersecurity, [Online]. Available: https://www.unb.ca/cic/datasets/botnet.html.

[31] "KDD98," [Online]. Available: https://www.openml.org/d/23513.

[32] "KDD Cup 1999 Data," 2007. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[33] "NSL-KDD," 2009. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html.

[34] P. e. al, "Packet and flow based network intrusion dataset. Contemporary Computing," *Springer Berlin Heidelberg,* pp. 322-334, 2012.

[35] John McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln

Laboratory," *ACM transactions on Information and system Security,* vol. 3, pp. 262-294, 2000.

[36] V.Mahoney and K.Philip, "An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection."Recent Advances in Intrusion Detection," *Springer Berlin Heidelberg,* 2003.

[37] A.Vasudevan, E. Harshini and S. Selvakumar, "SSENet-2011: a network intrusion detection system dataset and its comparison with KDD CUP 99 dataset," in *Second Asian Himalayas International Conference*, 2011.

[38] N. Moustafa and Jill Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).," in *Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, November 2015.

[39] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu and Ali, "A Detailed Analysis of the KDD CUP 99 Data Set," in *CISDA'09 Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications*.

[40] "How valuable do you think feature selection is in machine learning? Which do you think improves accuracy more, feature selection or feature engineering? - Quora," [Online]. Available: http://bit.ly/2UM1UcI.

[41] "An Introduction to Feature Selection," [Online]. Available: http://bit.ly/2UU8OwU.

[42] B.Azhagusundari and Antony Selvadoss Thanamani, "Feature Selection based on Information Gain," *International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN,* vol. 2, no. 2, pp. 2278-3075, January 2013.

[43] Xiaofei He, Deng Cai and Partha Niyogi, "Laplacian Score for Feature Selection".

[44] Igor Kononenko and Marko Robnik-Sˇikonja, "Theoretical and Empirical Analysis of ReliefF and RReliefF," University of Ljubljana, Faculty of Computer and Information Science.

[45] Quanquan Gu, Zhenhui Li and Jiawei Han, "Generalized Fisher Score for Feature Selection".

[46] A. C.C., Data Mining: The textbook, New York: Springer, 2015, p. 290.

[47] T. Chen, "Quora," [Online]. Available: http://bit.ly/2XzcAsk.

[48] "MXNet," [Online]. Available: https://mxnet.apache.org.

[49] M. Kubat, in *An Introduction to Machine learning*, 2017, p. 212.

[50] "Damage Caused by Classification Accuracy and Other Discontinuous Improper Accuracy Scoring Rules," Statistical Thinking, 2017. [Online]. Available: http://www.fharrell.com/post/class-damage/.

[51] S. Narkhede, "Understanding AUC - ROC Curve," Towards Data Scienece , [Online]. Available: https://bit.ly/2E0YdqU.

[52] "Scikit-learn," [Online]. Available: https://scikit-learn.org/stable/.

[53] "Pandas: Python Data Analysis Library," [Online]. Available: https://pandas.pydata.org.

[54] "Classification Algorithms in Machine Learning – Medium," [Online]. Available: http://bit.ly/2ZpSlPw.

[55] "Types of classification algorithms in Machine Learning," [Online]. Available: http://bit.ly/2IAmDcY.

[56] "Machine Learning Algorithms: Which One to Choose for Your Problem," [Online]. Available: http://bit.ly/2XxaEAo.

[57] "Label Propagation Algorithm," [Online]. Available: https://en.wikipedia.org/wiki/Label_Propagation_Algorithm.

[58] "Semisupervised Learning Approaches," [Online]. Available: http://videolectures.net/mlas06_mitchell_sla/.

[59] D.-H. Lee, "Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks," in *ICML Workshop: Challenges in Representation Learning (WREPL)*, Atlanta, Georgia, USA, 2013.

[60] B. Settles, "Active Learning Literature Survey," Computer Sciences Technical Report 1648 University of Wisconsin–Madison, January 26, 2010.

[61] S. Hosein, "DataCamp: Active Learning: Curious AI Algorithms," 9 February 2018. [Online]. Available: https://www.datacamp.com/community/tutorials/active-learning.

[62] V. Kodžoman, "Pseudo-labeling a simple semi-supervised learning method," 6 September 2017. [Online]. Available: http://bit.ly/2GxZxBI.

[63] Sheng-Jun Huang, Rong Jin and Zhi-Hua Zhou, "Active Learning by Querying Informative and Representative Examples," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 36, no. 10, pp. 1936 - 1949, October 2014.

[64] S. Hosein, "Active Learning: Curious AI Algorithms," DataCamp , 9 February 2018. [Online]. Available: https://www.datacamp.com/community/tutorials/active-learning.

[65] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

Appendix 1 – Classification performance comparison for different set of features

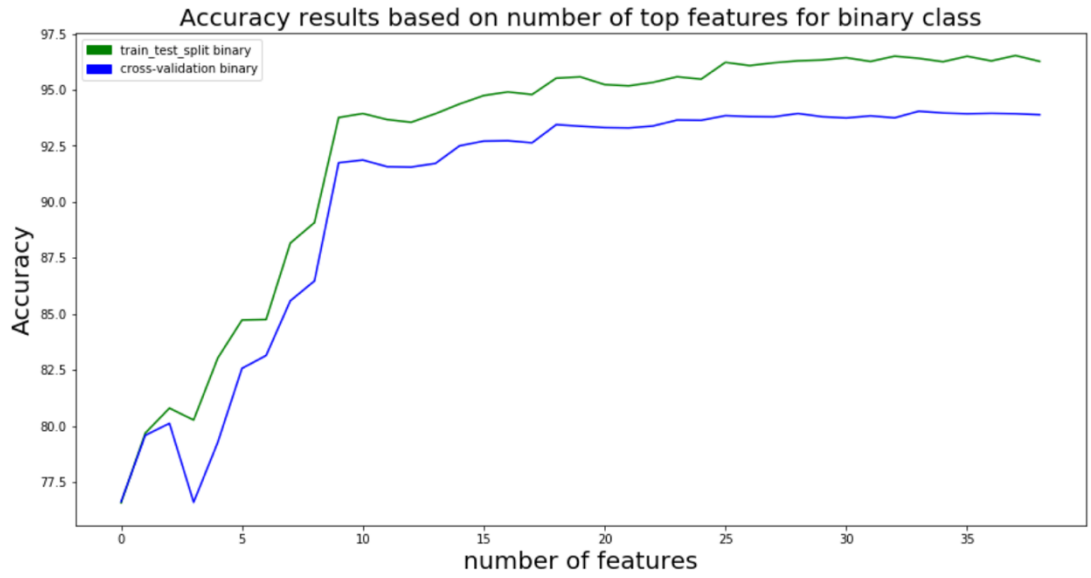
The results presented in table below, provided to support the statement that elimination of the three categorical features (proto, state and service) does not influence the classification performance. Result provided both for binary class and multi class problems and different evaluation performance metrics are used. First table has all 42 features, second table has only 39 numerical features.

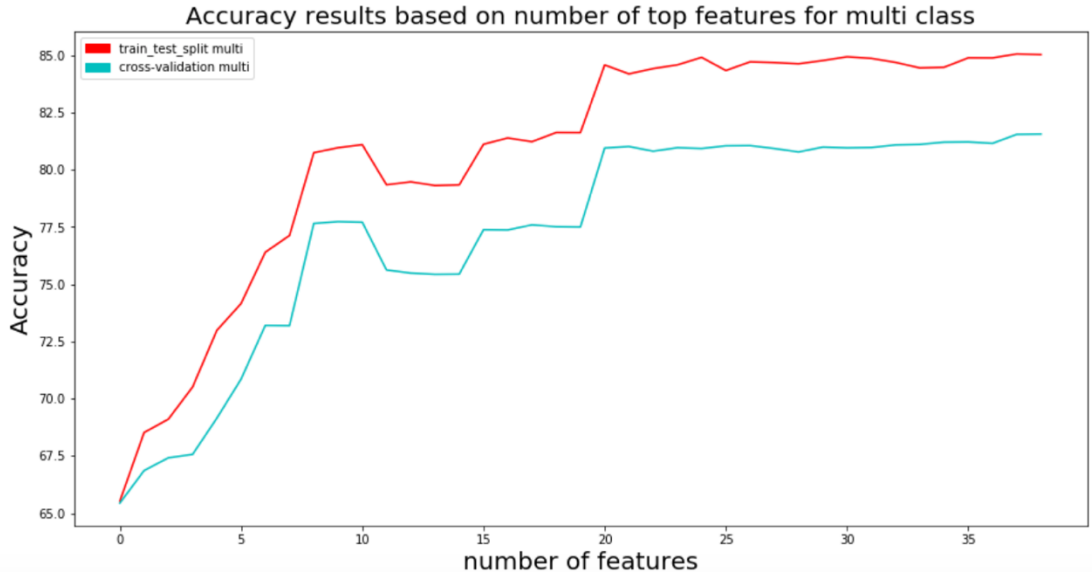
Scenario/Models	**Binary Classification**			**Multi class classification**
	Accuracy	*Recall*	*Precision*	*Accuracy*
XGB	94.76	95.36	95.35	84.04
DT	94.16	95.50	94.25	81.99
KNN	87.79	87.33	90.54	75.47
RF	95.74	95.39	97.24	83.63
LR	86.18	87.85	88.23	74.04

Scenario/Models	**Binary Classification**			**Multi class classification**
	Accuracy	*Recall*	*Precision*	*Accuracy*
XGB	94.52	95.33	95.01	83.93
DT	93.91	95.29	94.24	81.58
KNN	87.76	87.27	90.53	75.51
RF	95.50	95.22	96.77	83.74
LR	84.41	87.54	86.25	73.06

Appendix 2 – Classification accuracy with top ranked features

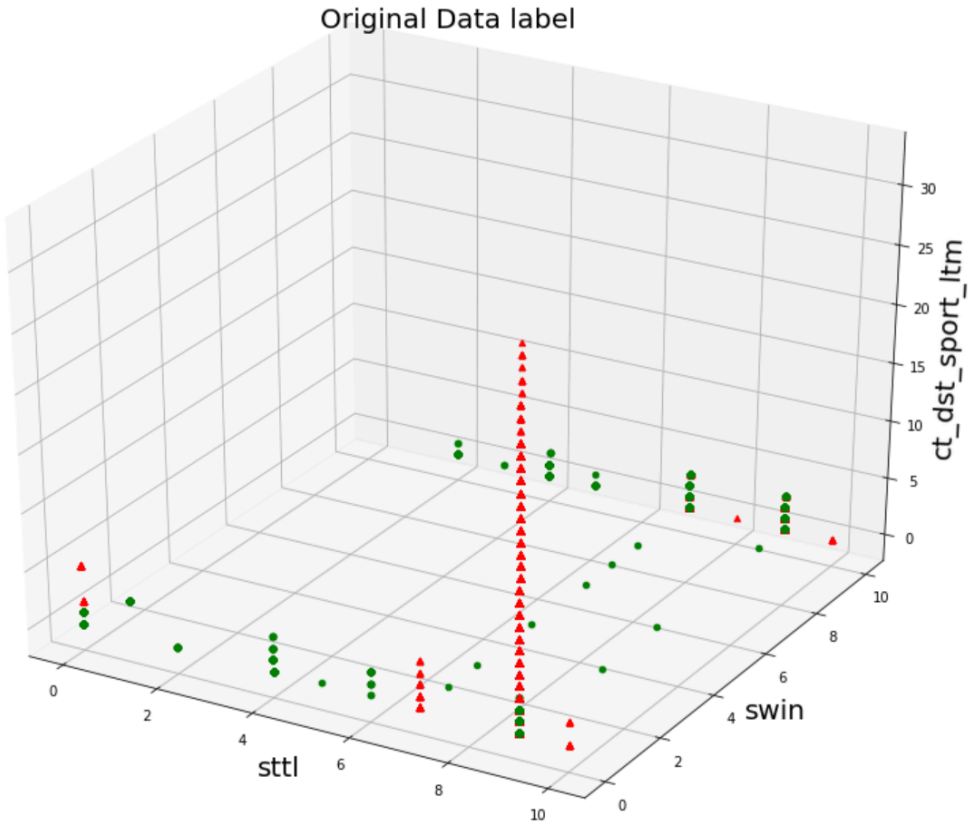
Accuracy result based on the list of score obtained from Fisher's score feature selection, trained with Decision Tree Classification.

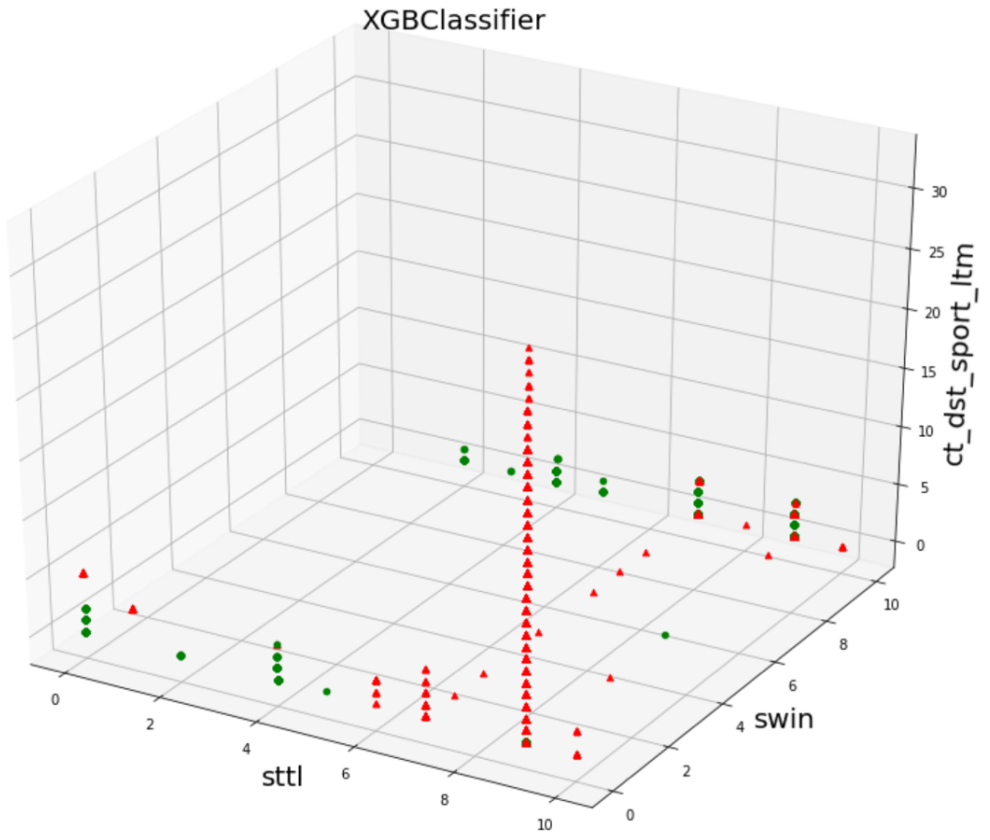


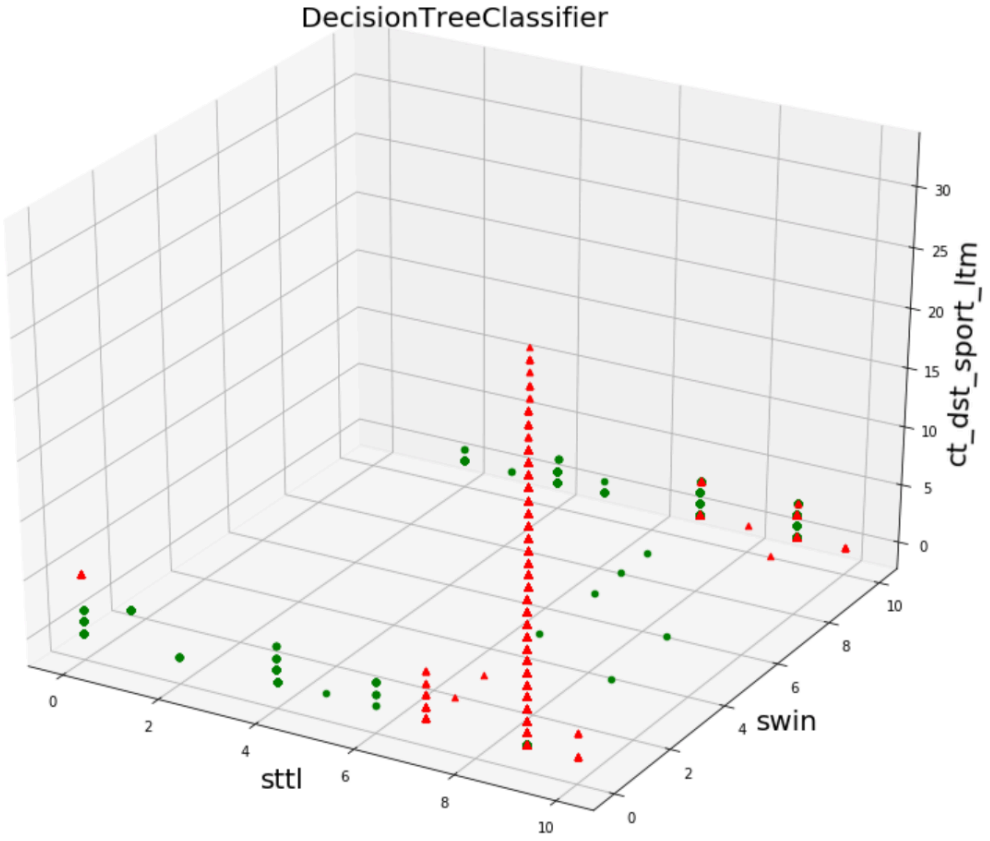


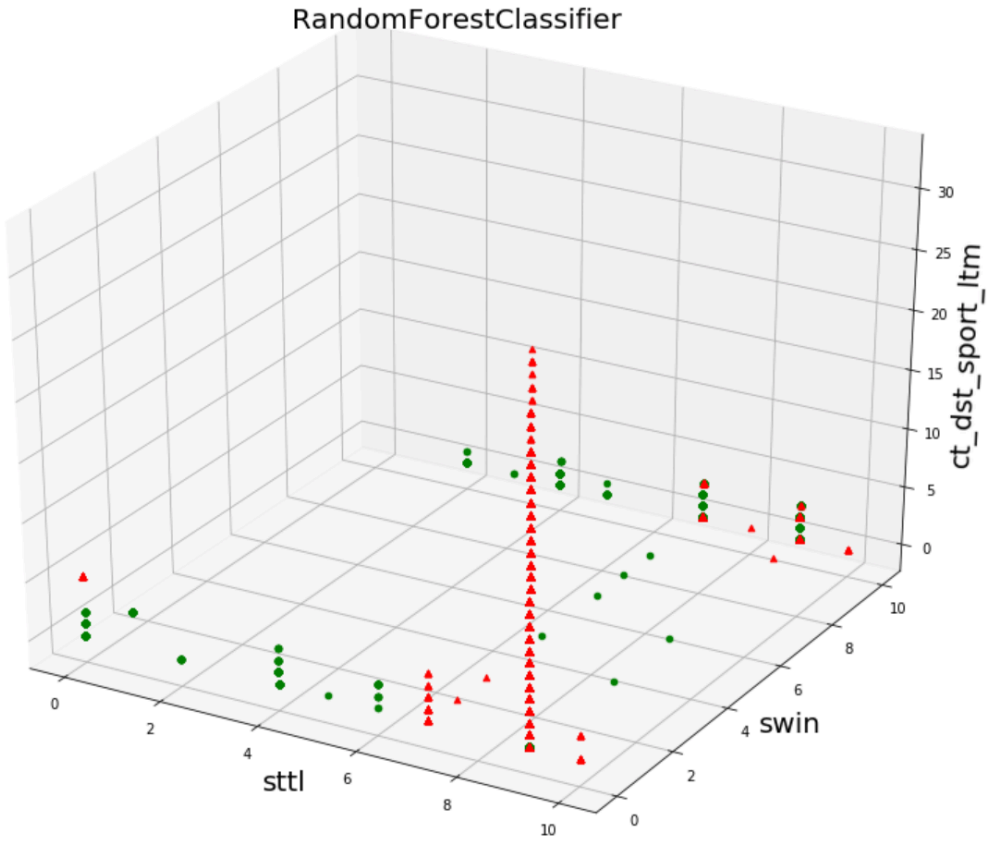
Appendix 3 – Scatter Plot with three features

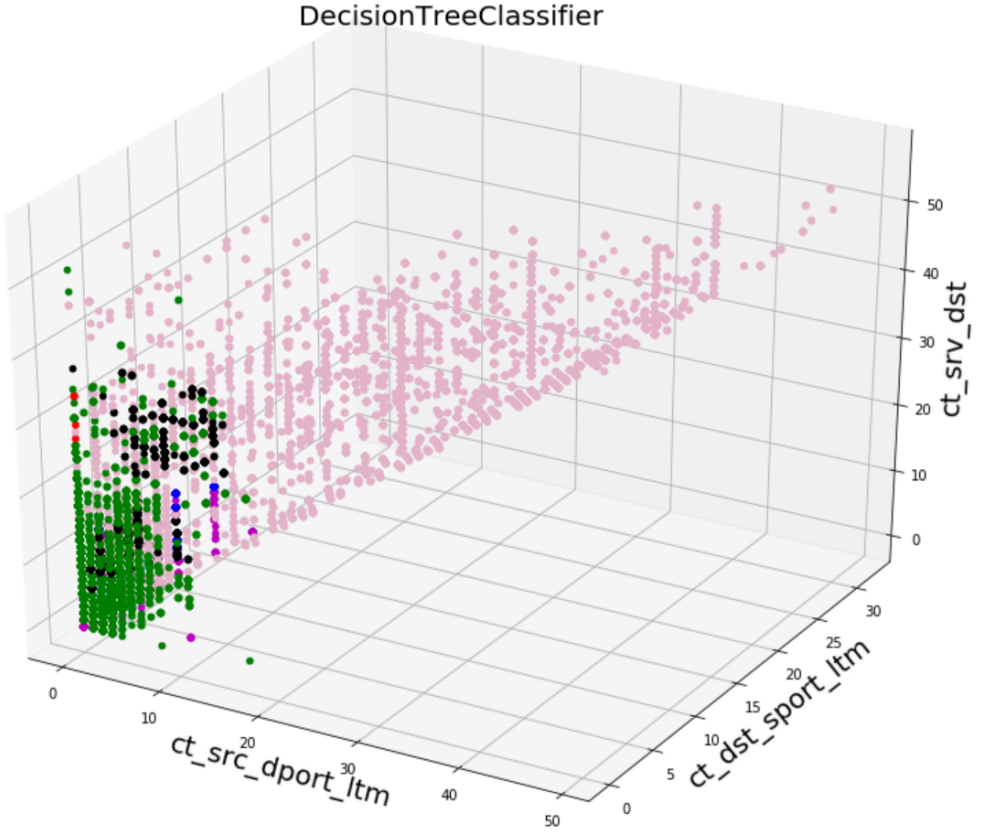
Here different scatter plots are presented with the binary class and multi-class original labeled data points and prediction results from classification models (XGB, Decision Tree, Random Forest).

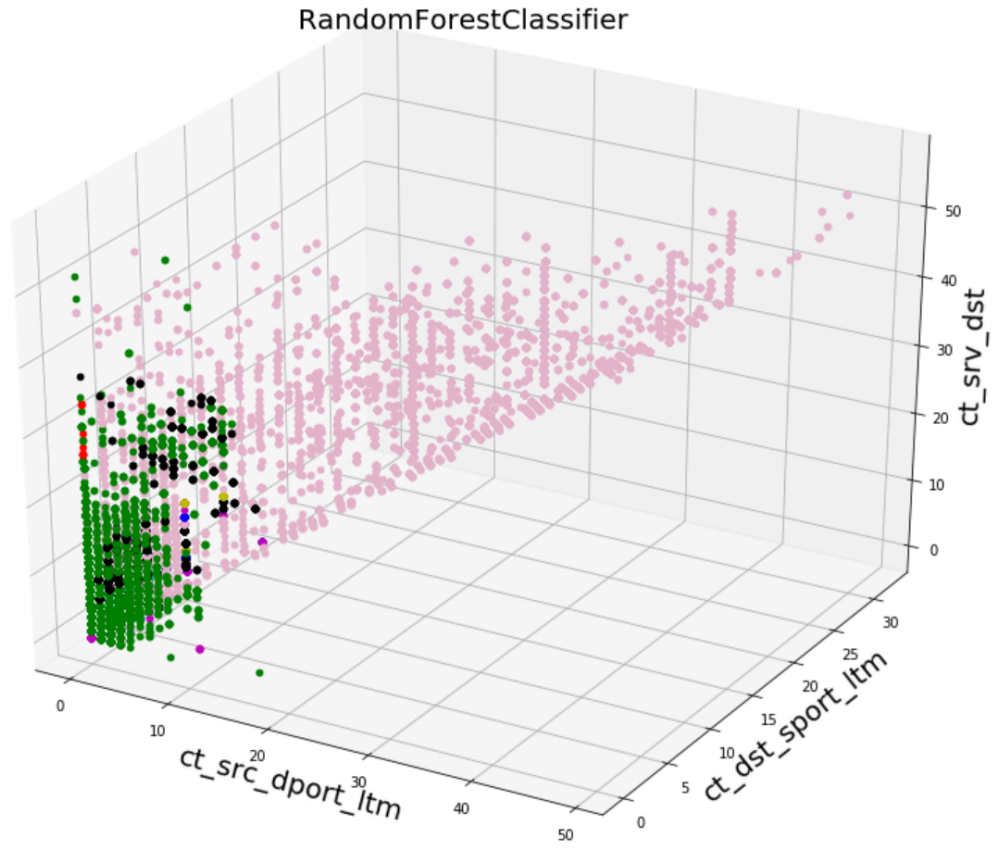


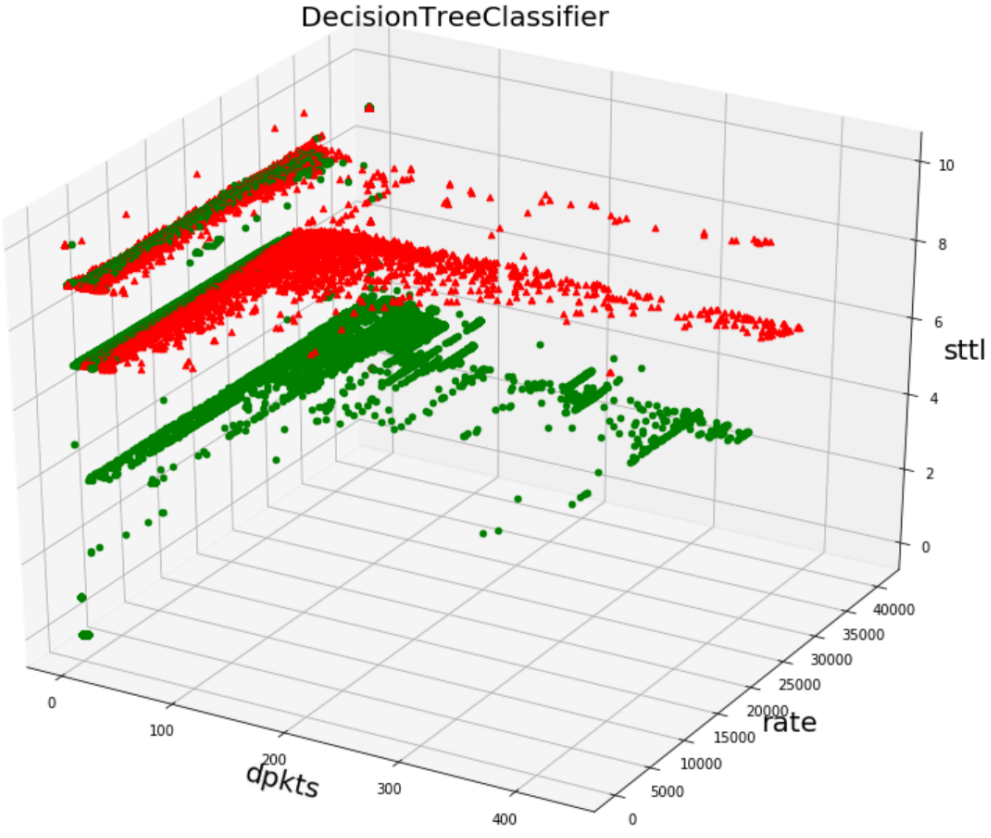

XGBClassifier
sttl
swin
ct_dst_sport_ltm

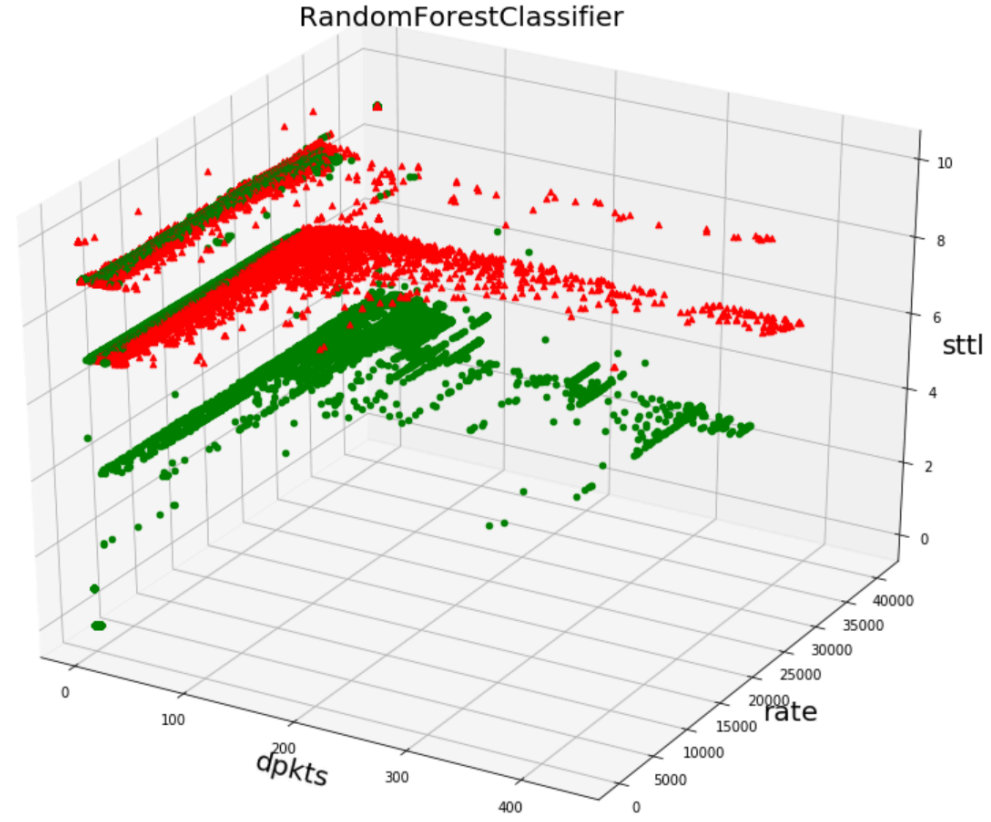

DecisionTreeClassifier
sttl
swin
ct_dst_sport_ltm

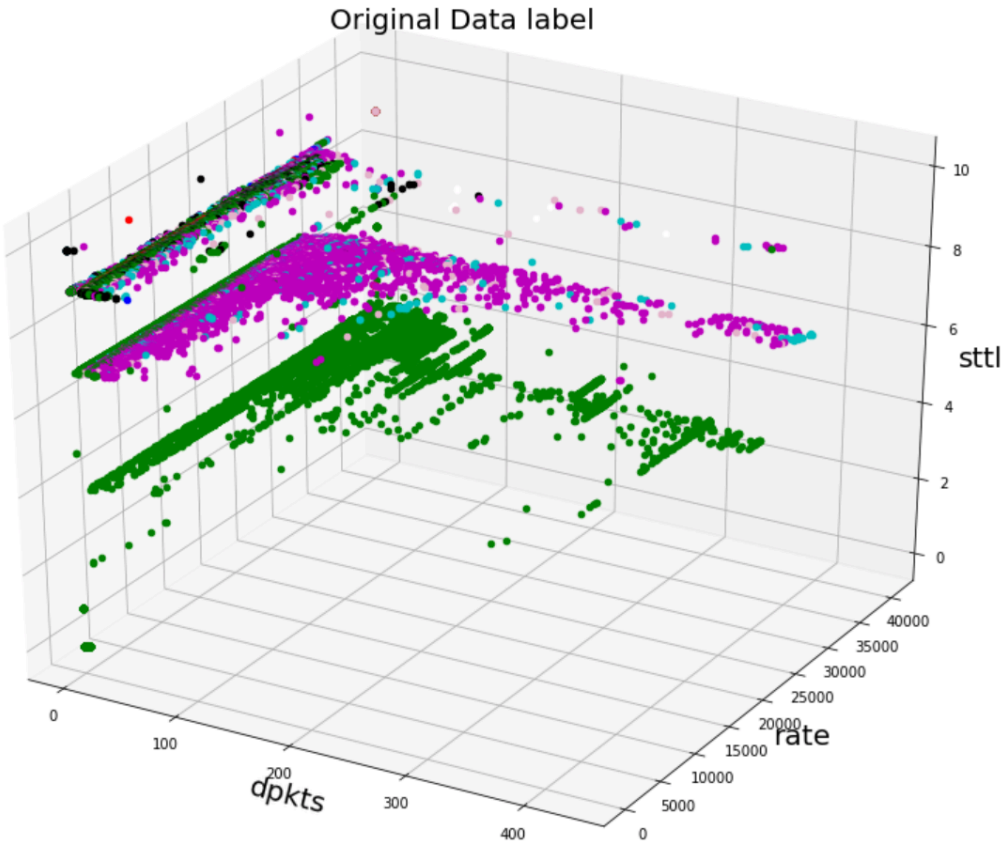

RandomForestClassifier
sttl
swin
ct_dst_sport_ltm

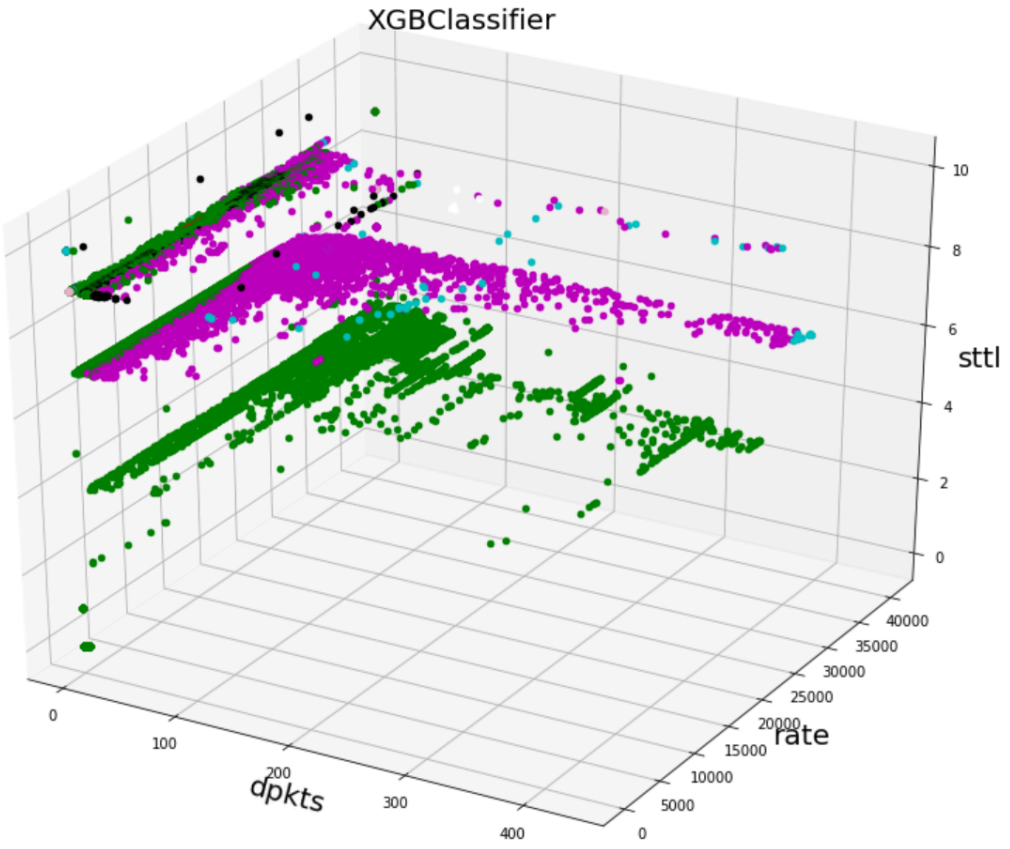

DecisionTreeClassifier
ct_src_dport_ltm
ct_dst_sport_ltm
ct_srv_dst

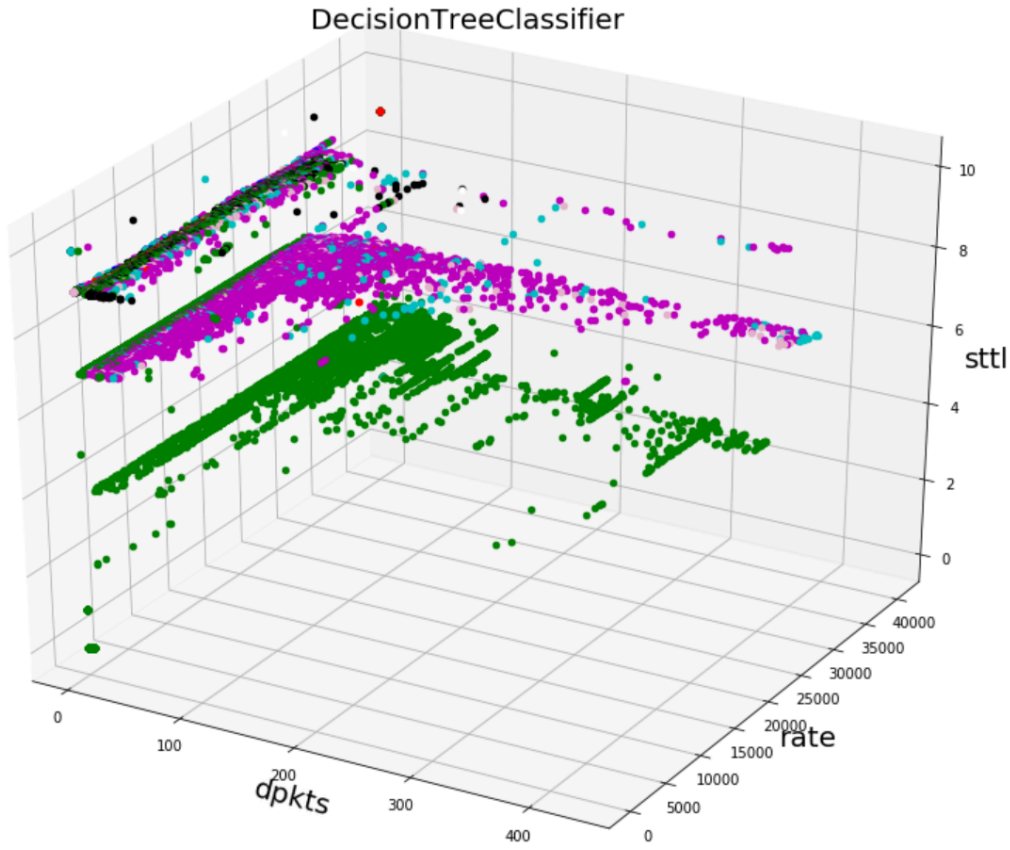

RandomForestClassifier
ct_src_dport_ltm
ct_dst_sport_ltm
ct_srv_dst


DecisionTreeClassifier
dpkts
rate
sttl


RandomForestClassifier
dpkts
rate
sttl

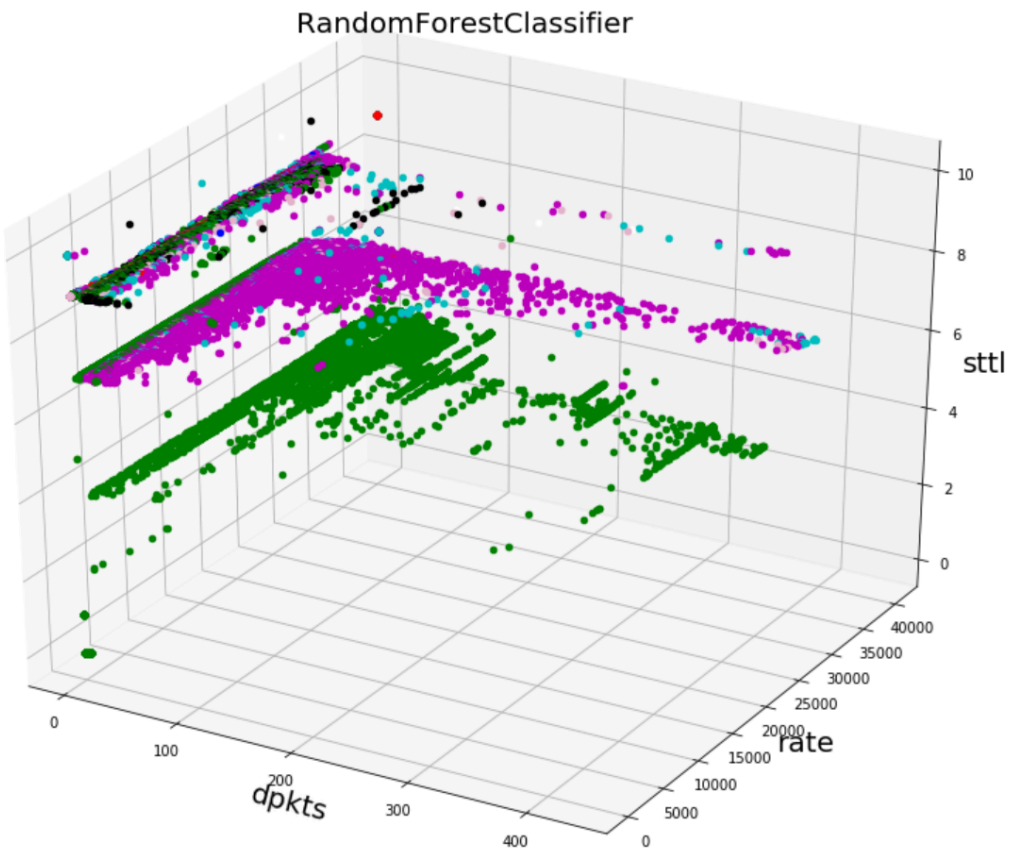

Original Data label
dpkts
rate
sttl

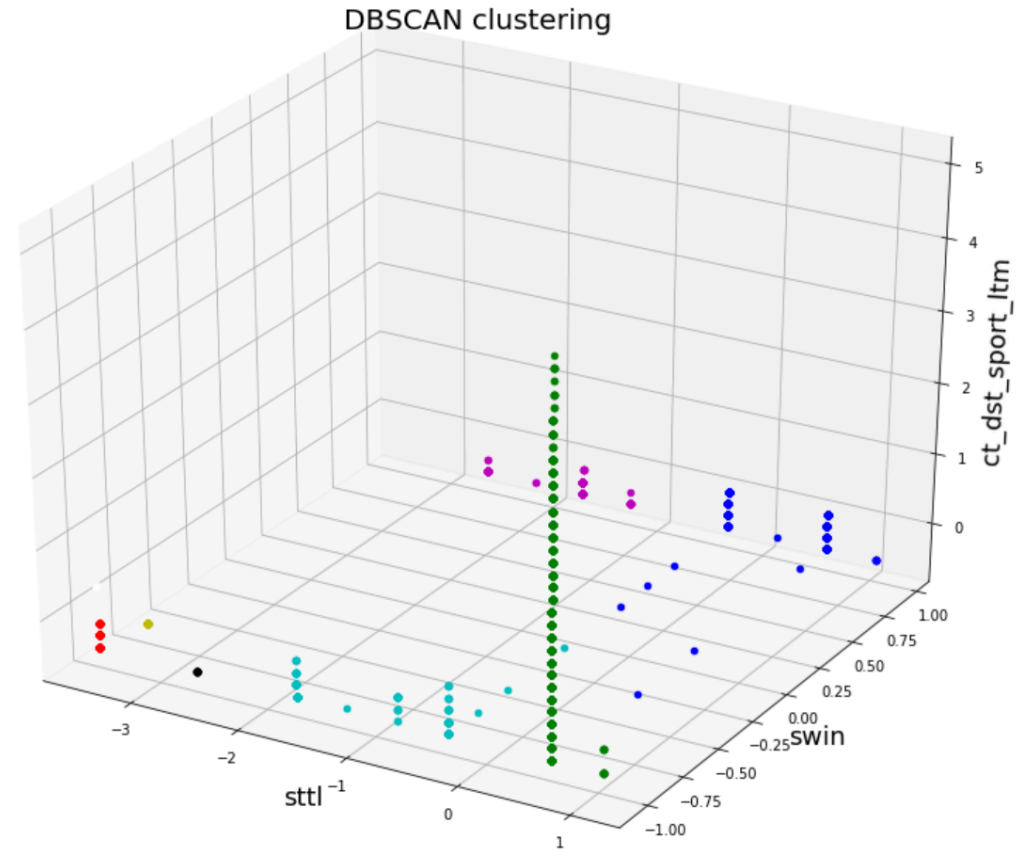

XGBClassifier
dpkts
rate
sttl

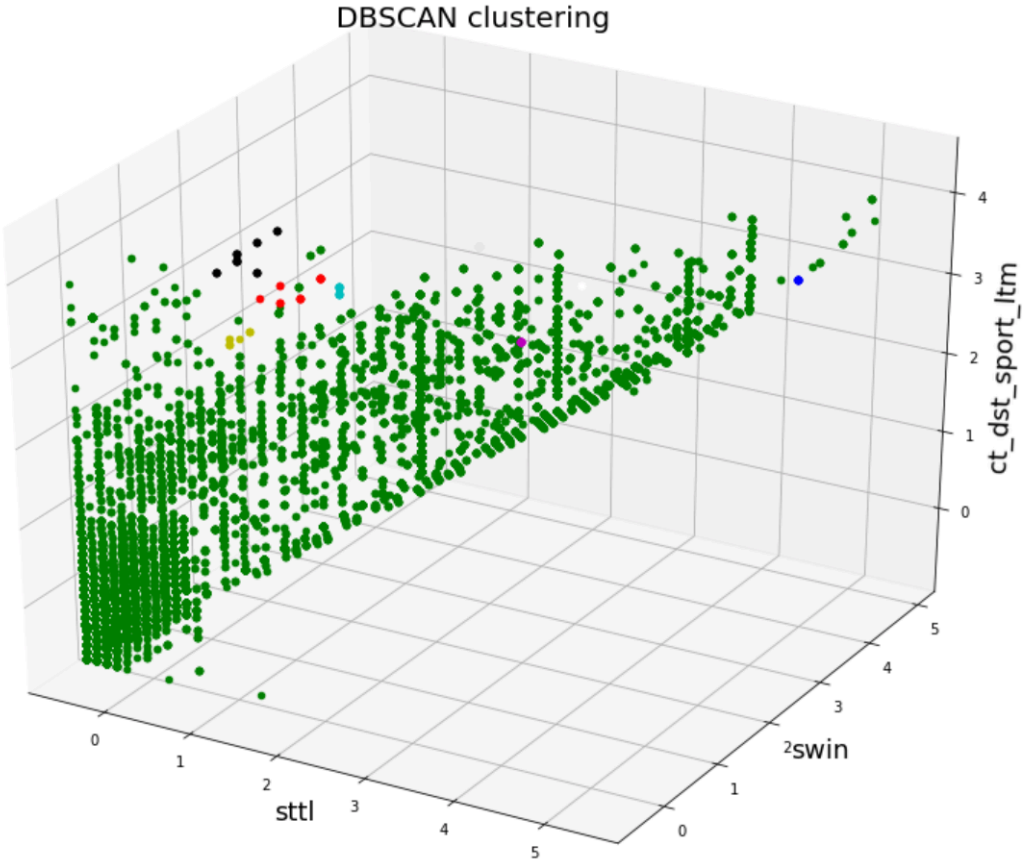

DecisionTreeClassifier
dpkts
rate
sttl

Appendix 4 – DBSCAN results

DBSCAN clustering results with the different sets of three features (most discriminative based on the Fisher's score) and classification models, represented in the scatter plots.

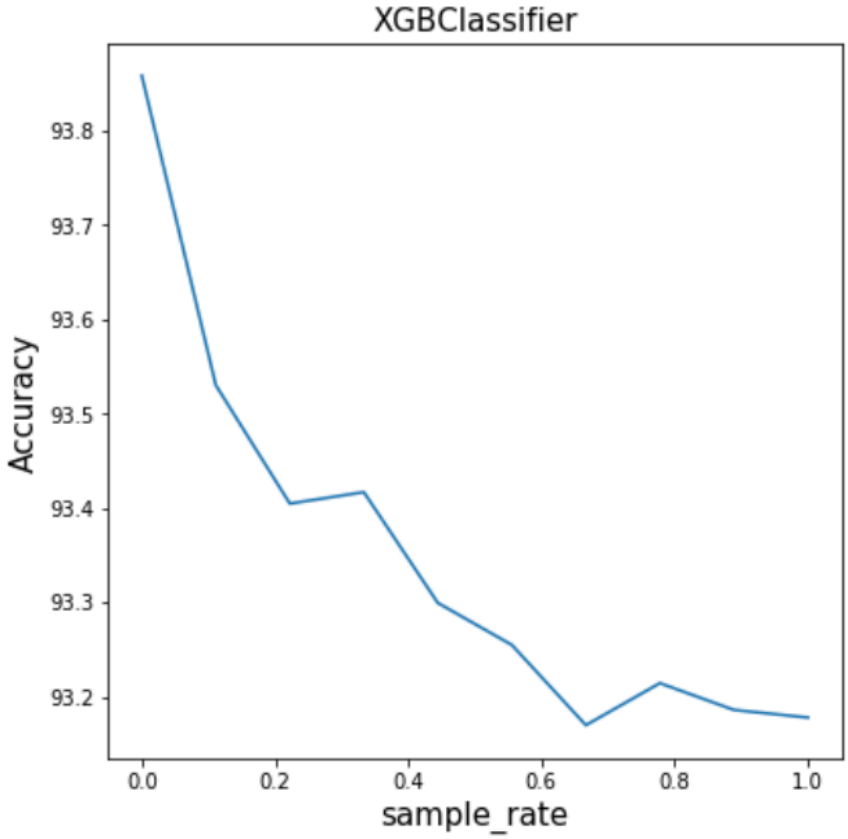


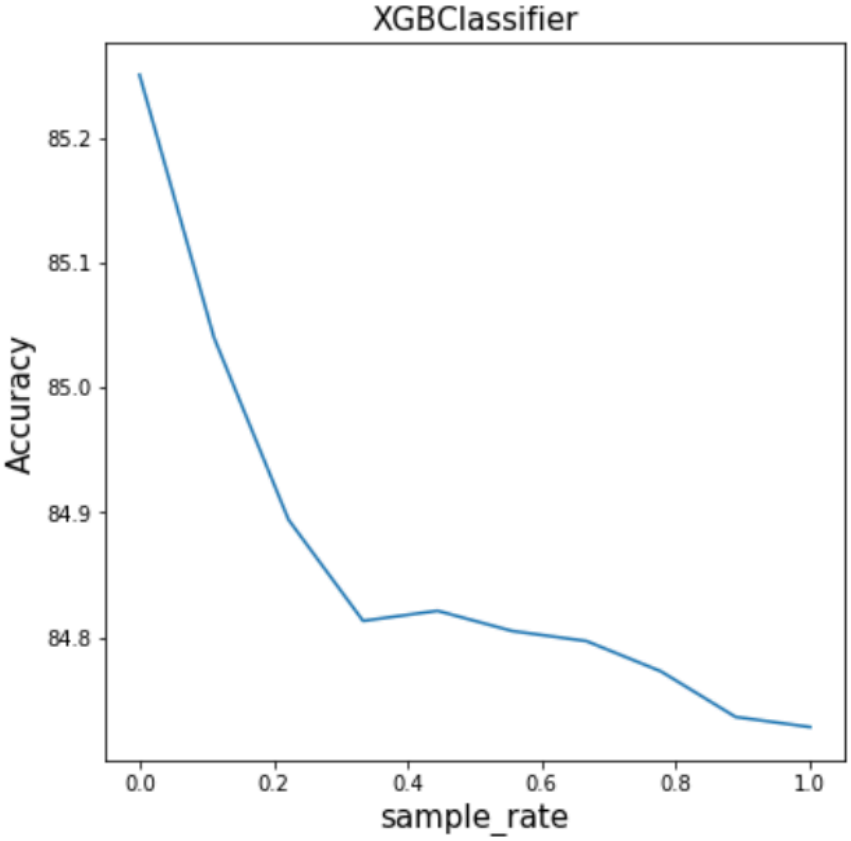

DBSCAN clustering
sttl
swin
ct_dst_sport_ltm

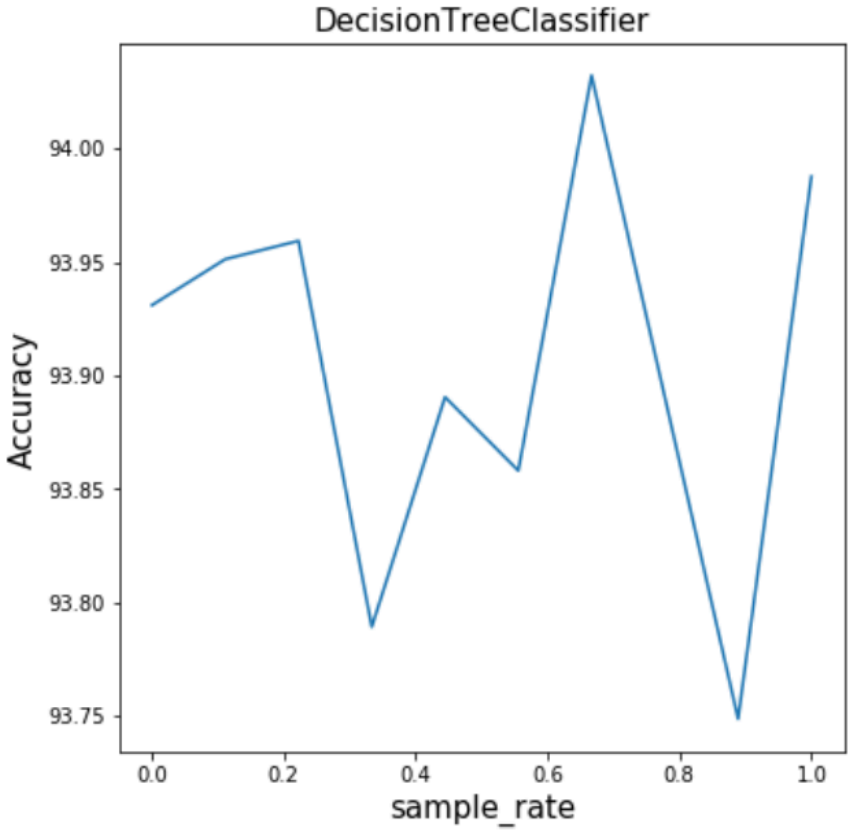

DBSCAN clustering
sttl
²swin
ct_dst_sport_ltm

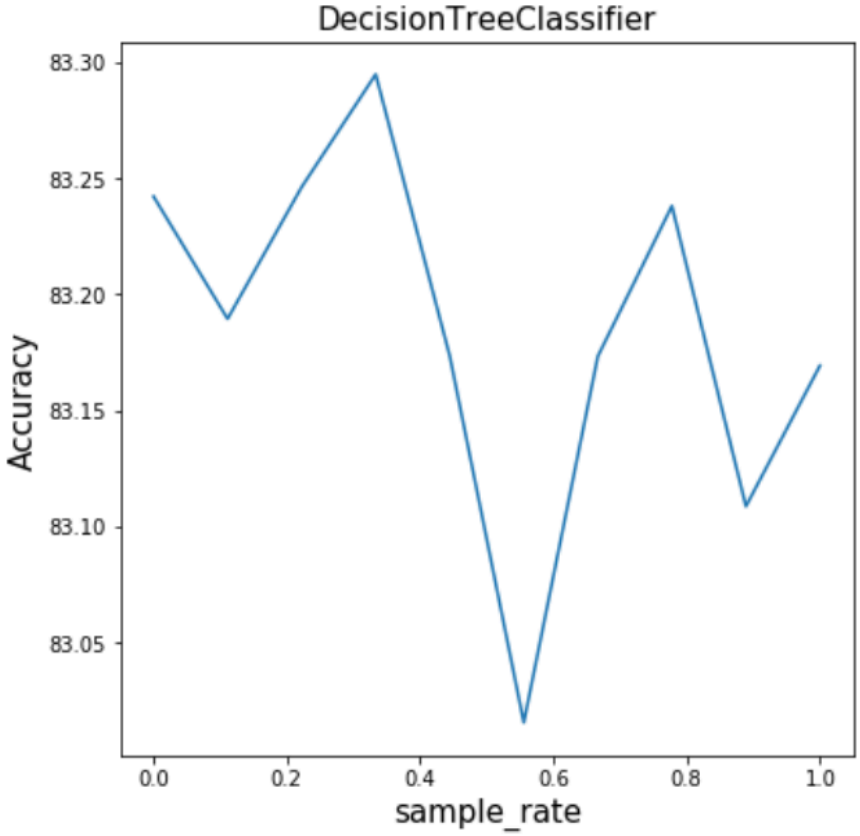
Appendix 5 – Semi-Supervised Pseudo-Labeling Results

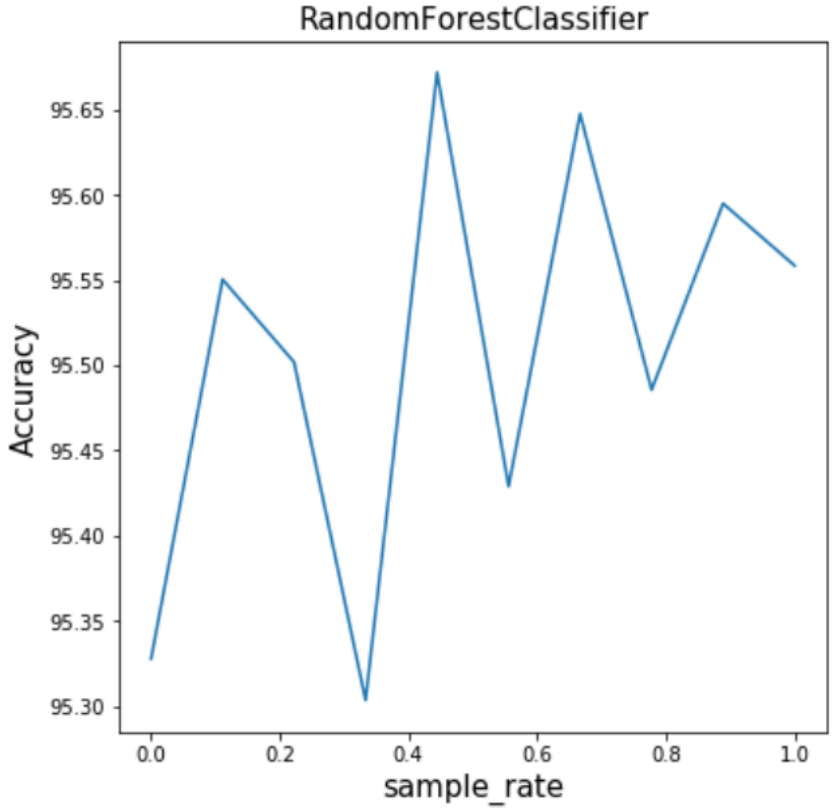
In this section, Semi-Supervised Pseudo-Labelling botnet prediction accuracy results are presented based on the sample rate and classification model. The first graph shows the results for binary classification and the following chart for the same model presents multi-class classification results.

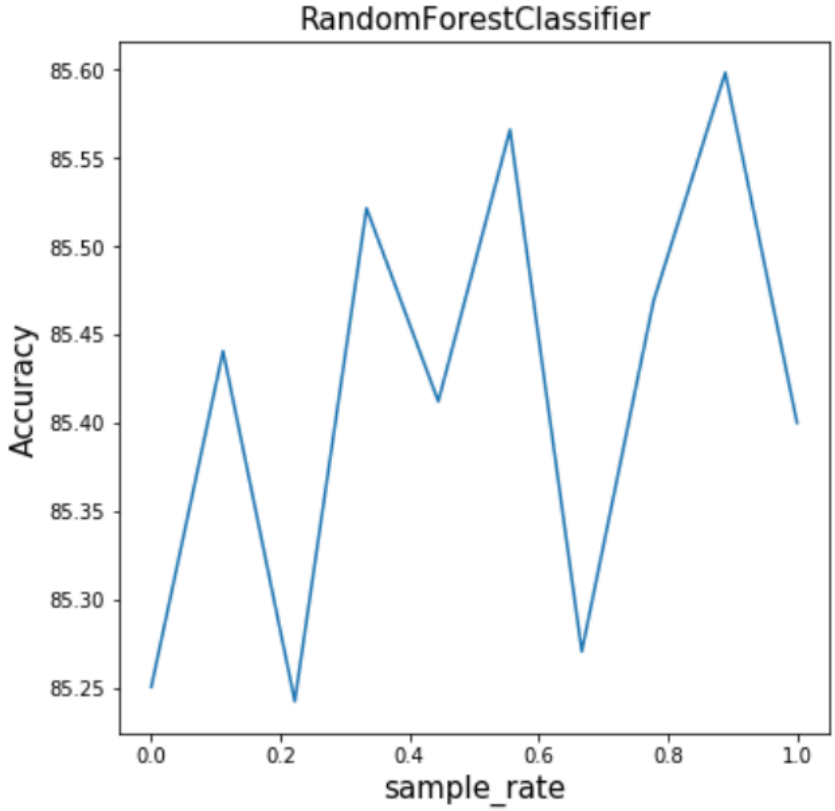



XGBClassifier
Accuracy
85.2
85.1
85.0
84.9
84.8
0.0
0.2
0.4
0.6
0.8
1.0
sample_rate


DecisionTreeClassifier
Accuracy
94.00
93.95
93.90
93.85
93.80
93.75
0.0
0.2
0.4
0.6
0.8
1.0
sample_rate

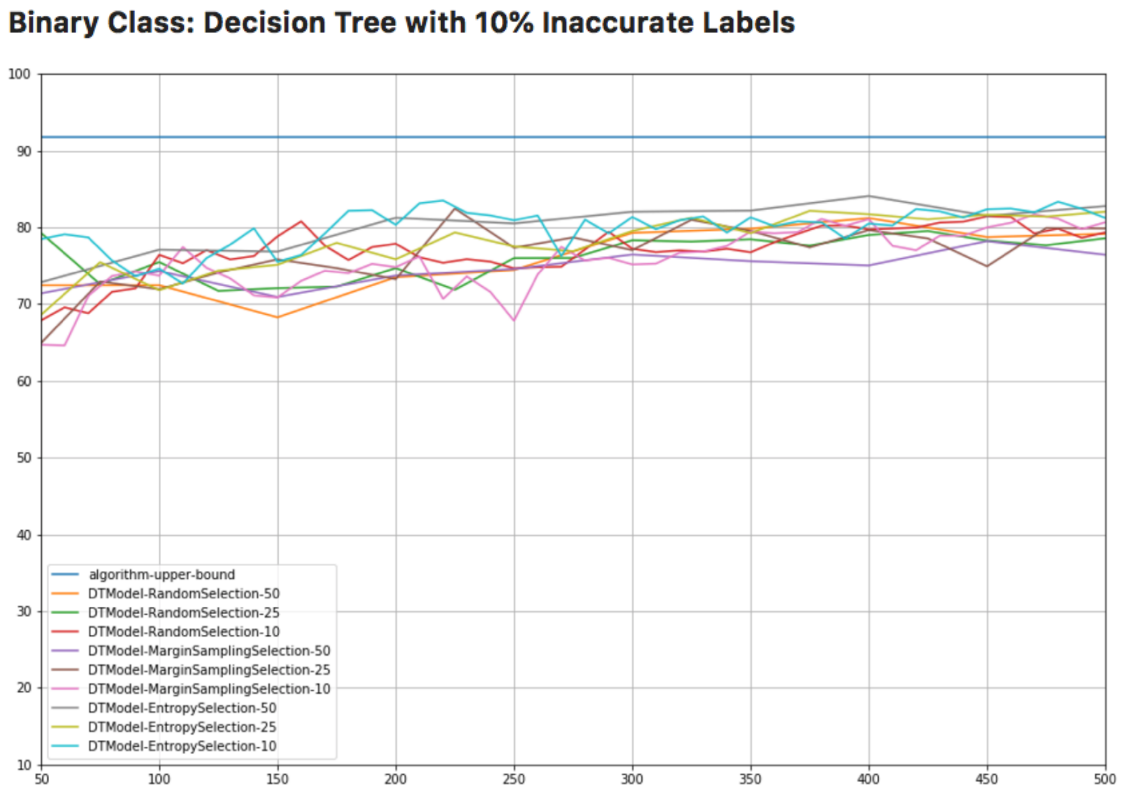

DecisionTreeClassifier
Accuracy
83.30
83.25
83.20
83.15
83.10
83.05
0.0
0.2
0.4
0.6
0.8
1.0
sample_rate


RandomForestClassifier
Accuracy
95.65
95.60
95.55
95.50
95.45
95.40
95.35
95.30
0.0
0.2
0.4
0.6
0.8
1.0
sample_rate

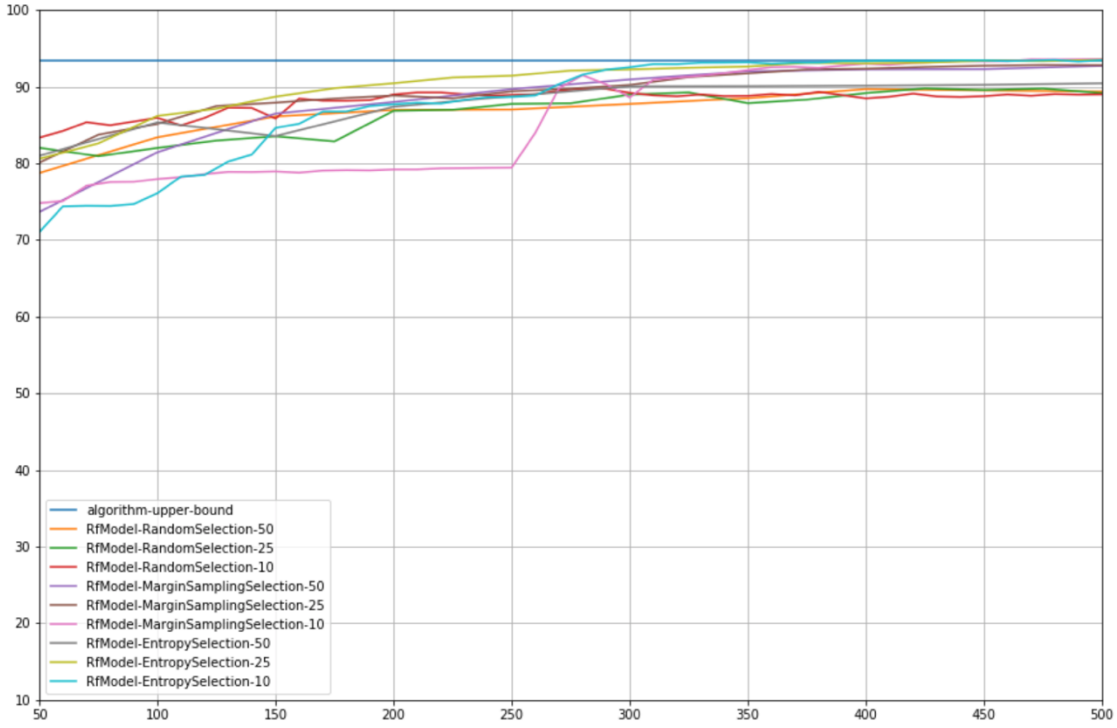

RandomForestClassifier
Accuracy
85.60
85.55
85.50
85.45
85.40
85.35
85.30
85.25
0.0
0.2
0.4
0.6
0.8
1.0
sample_rate

Appendix 6 – Active learning performance results when queried label is wrong

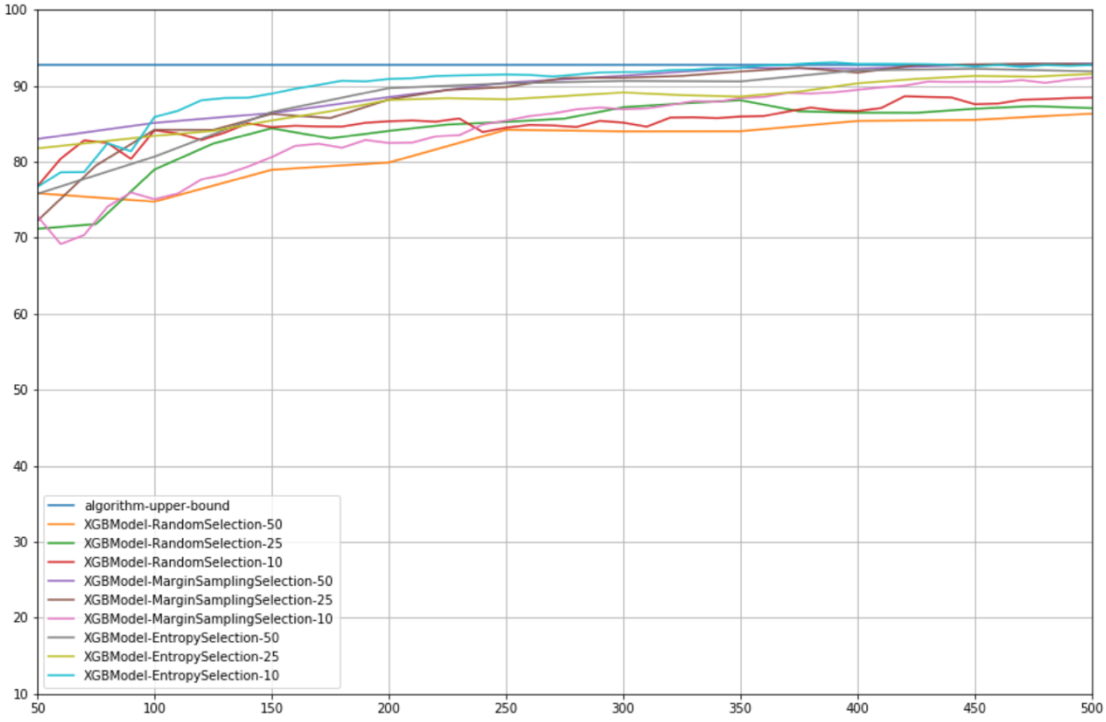
In this section, graphs for three models with three query selection methods are presented. Graphs contains binary class and multi class scenarios with 10%, 20%, 30%, 40%, 50% wrong labels. The blue straight line is the upper bound supervised classification accuracy with correct label. This line indicated to give overall comparison between fully supervised model using accurate label class and the active learning methods using wrong label class.



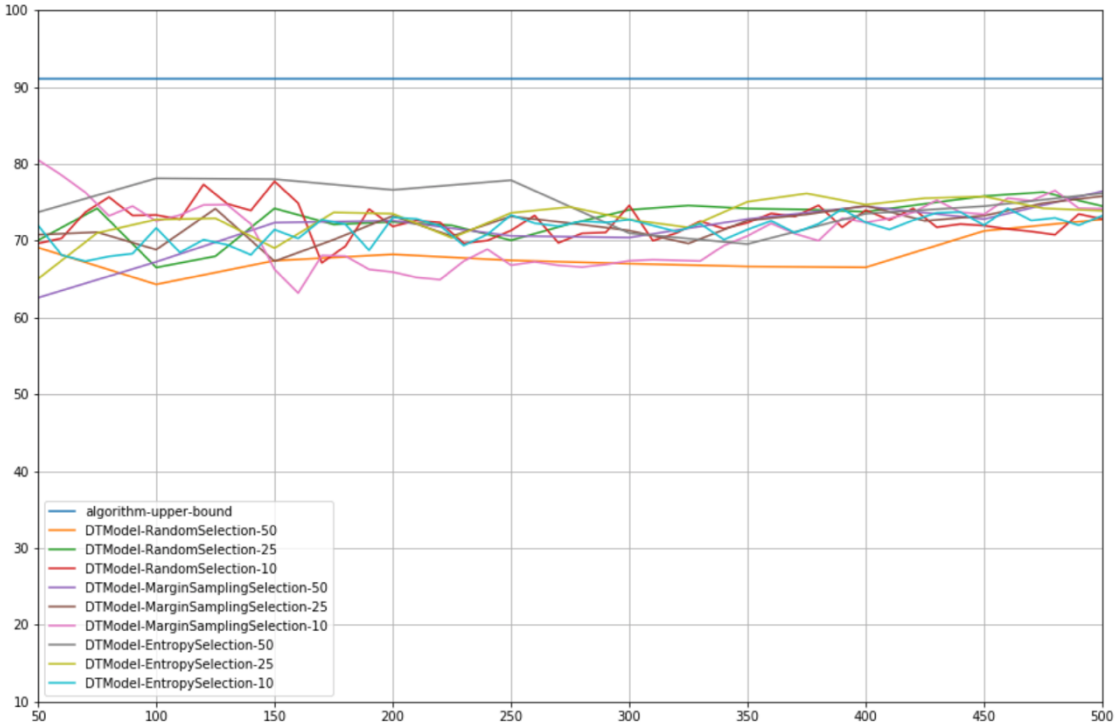
Binary Class: Random Forest with 10% Inaccurate Labels



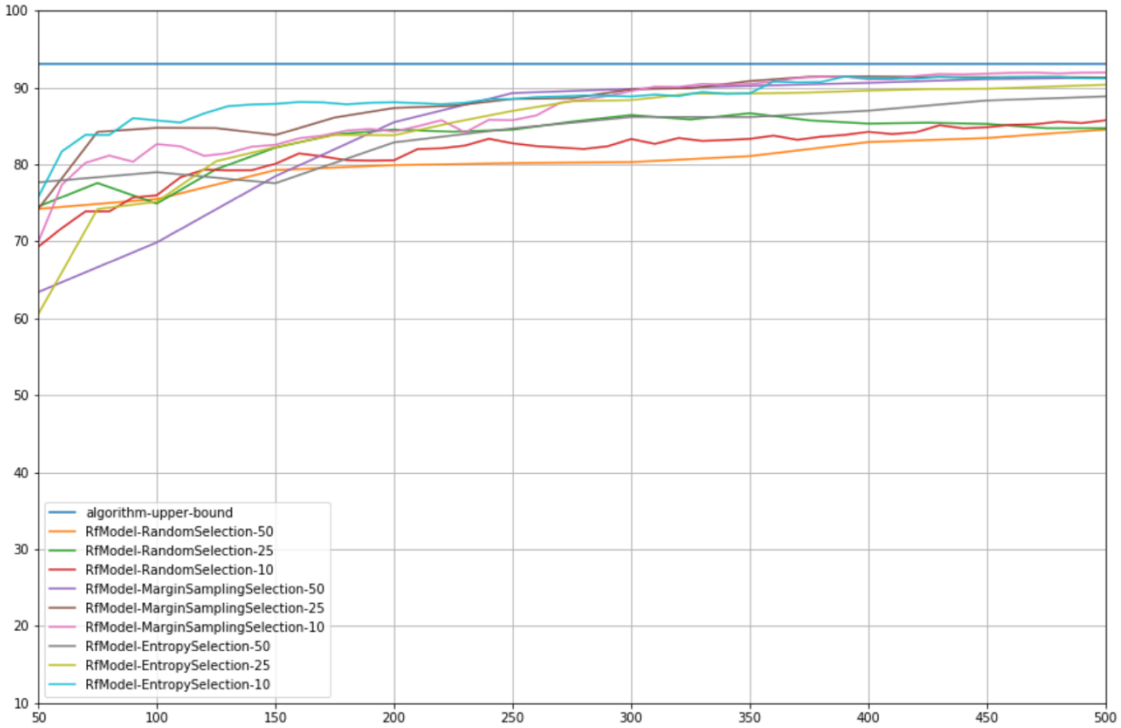
Binary Class: XGBoost with 10% Inaccurate Labels



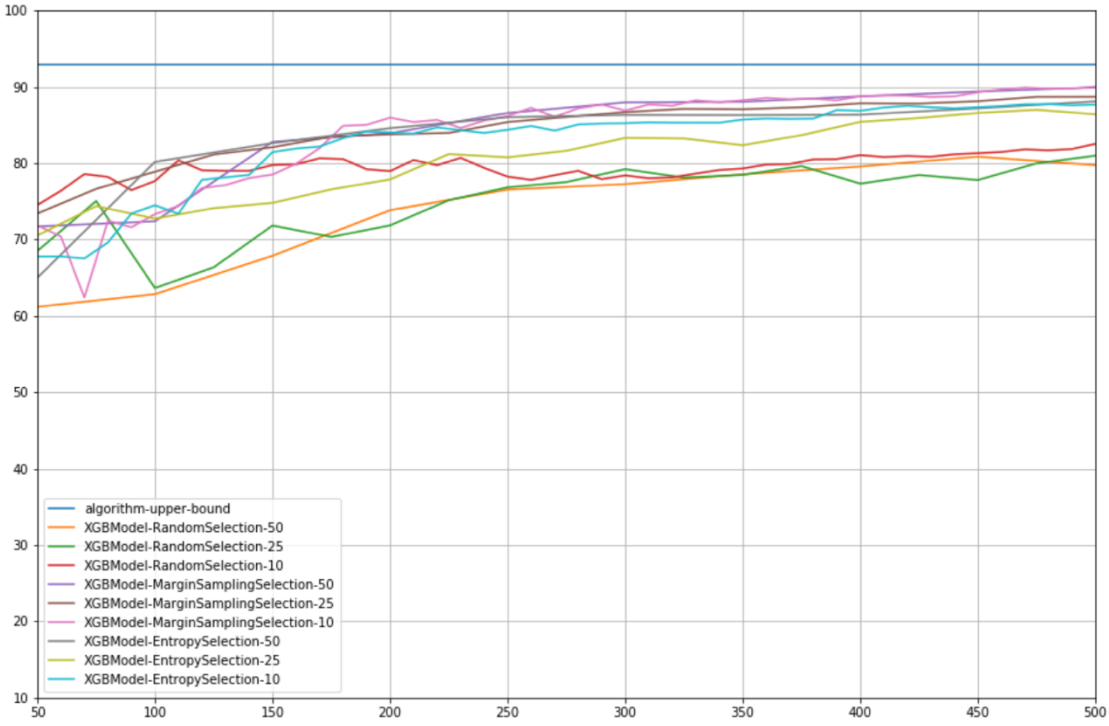
Binary Class: Decision Tree with 20% Inaccurate Labels



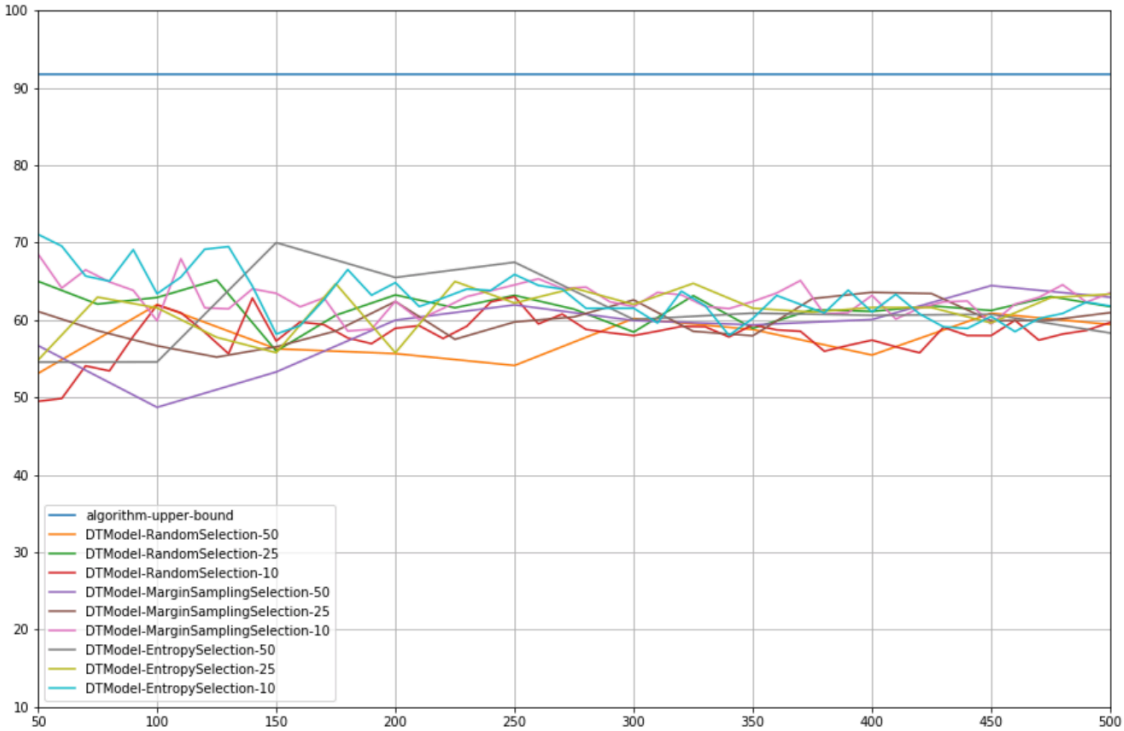
Binary Class: Random Forest with 20% Inaccurate Labels



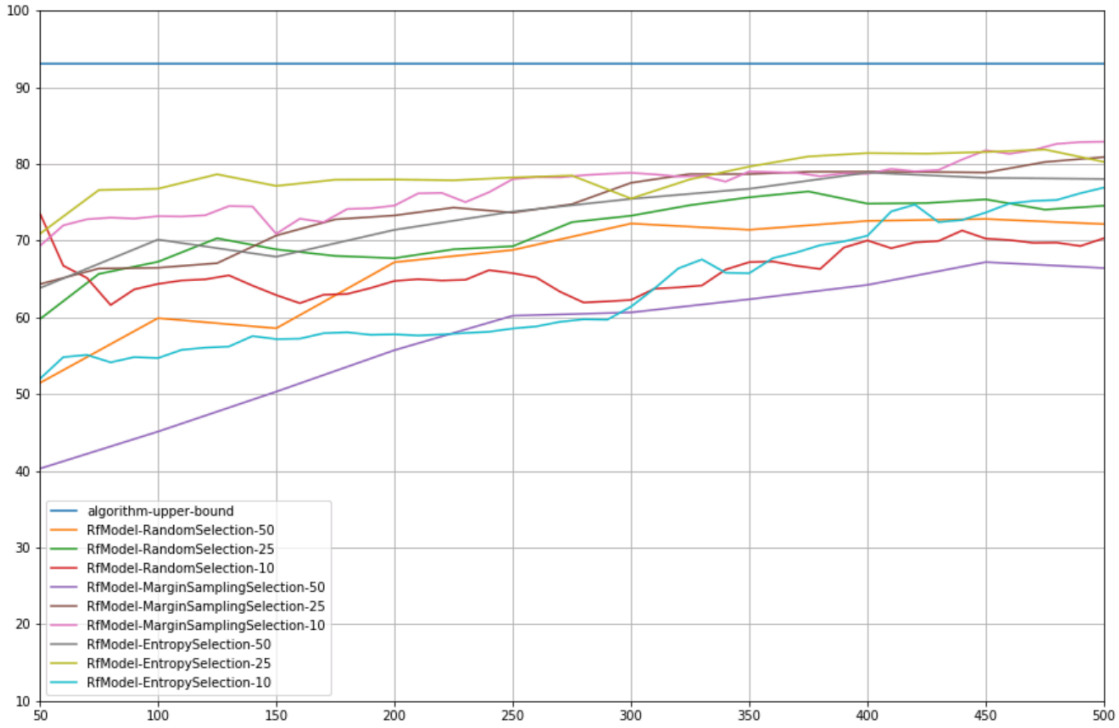
Binary Class: XGBoost with 20% Inaccurate Labels



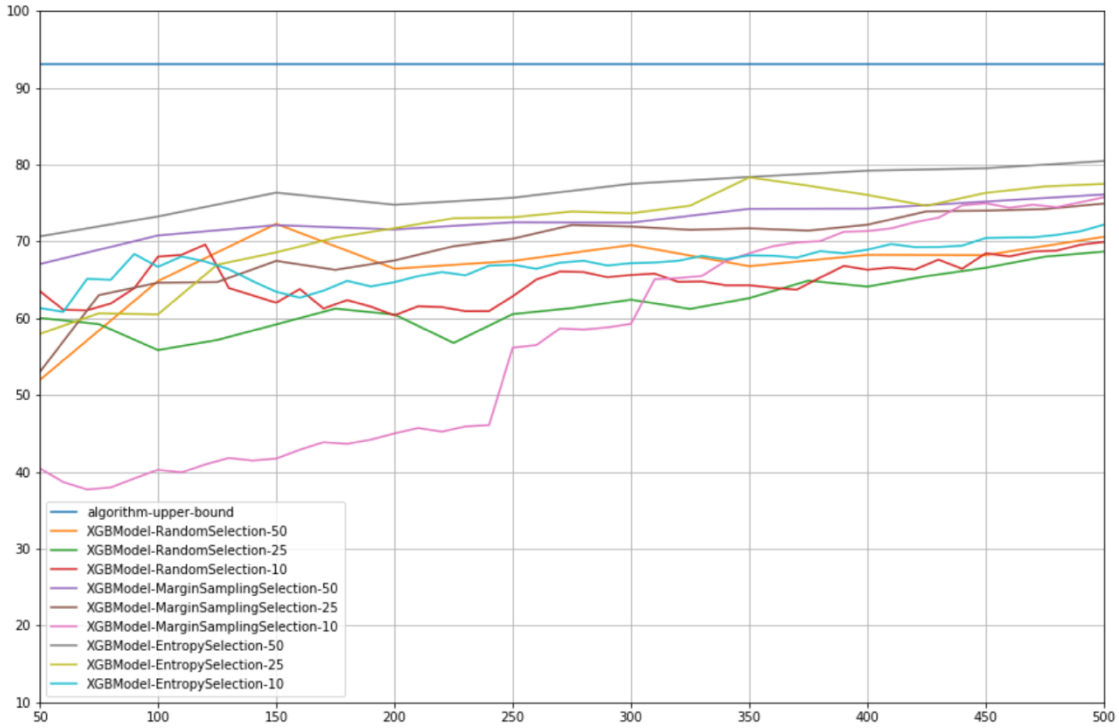
Binary Class: Decision Tree with 30% Inaccurate Labels



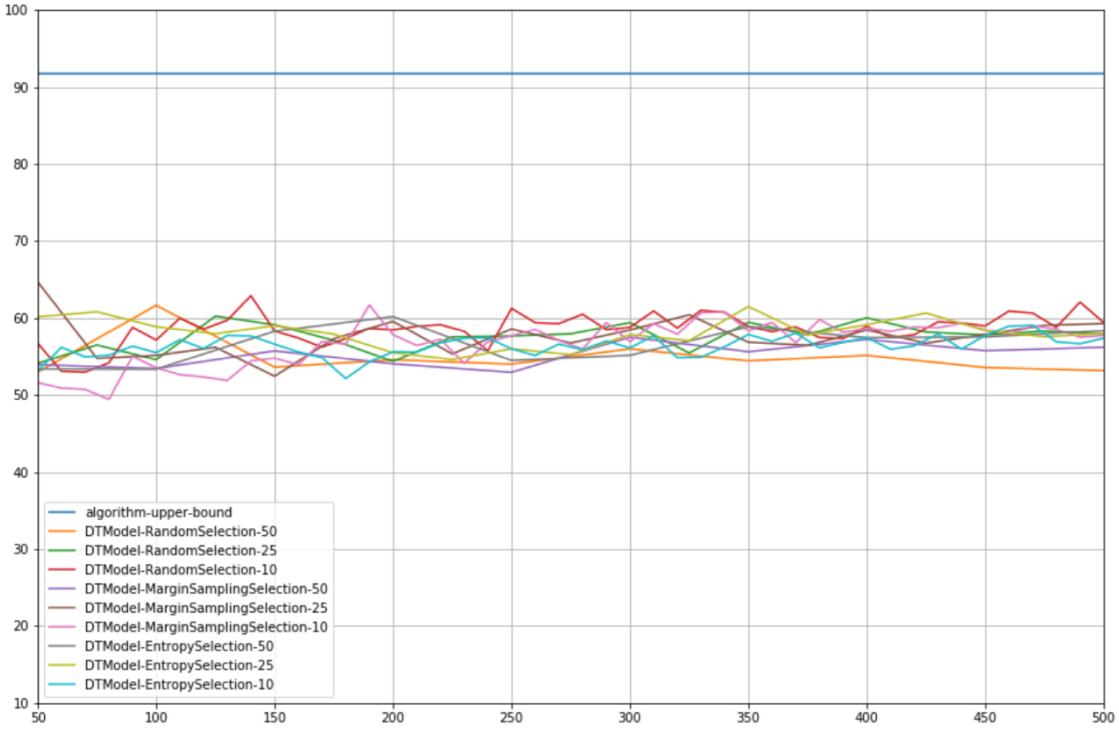
Binary Class: Random Forest with 30% Inaccurate Labels



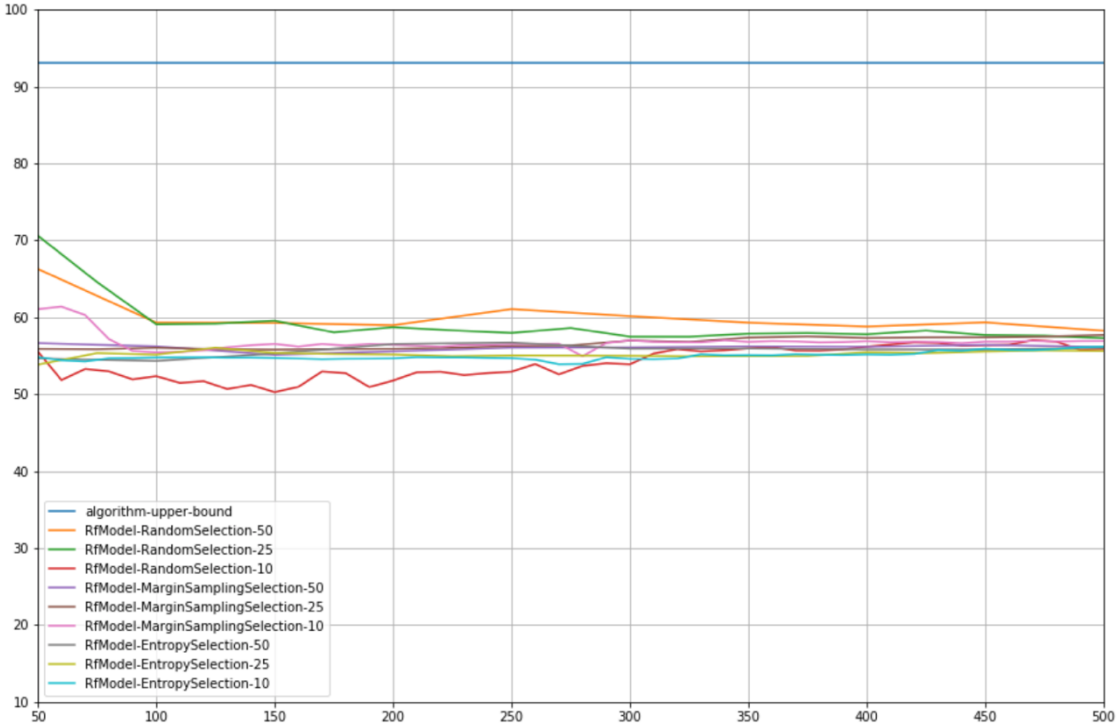
Binary Class: XGBoost with 30% Inaccurate Labels



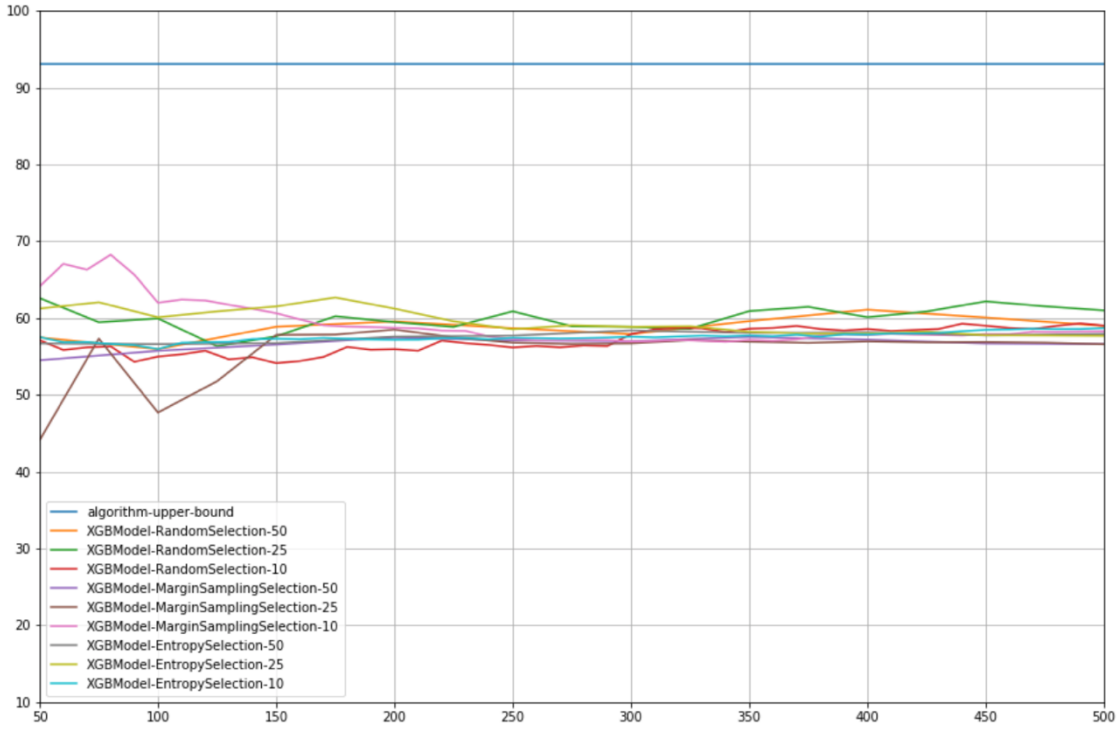
Binary Class: Decision Tree with 40% Inaccurate Labels



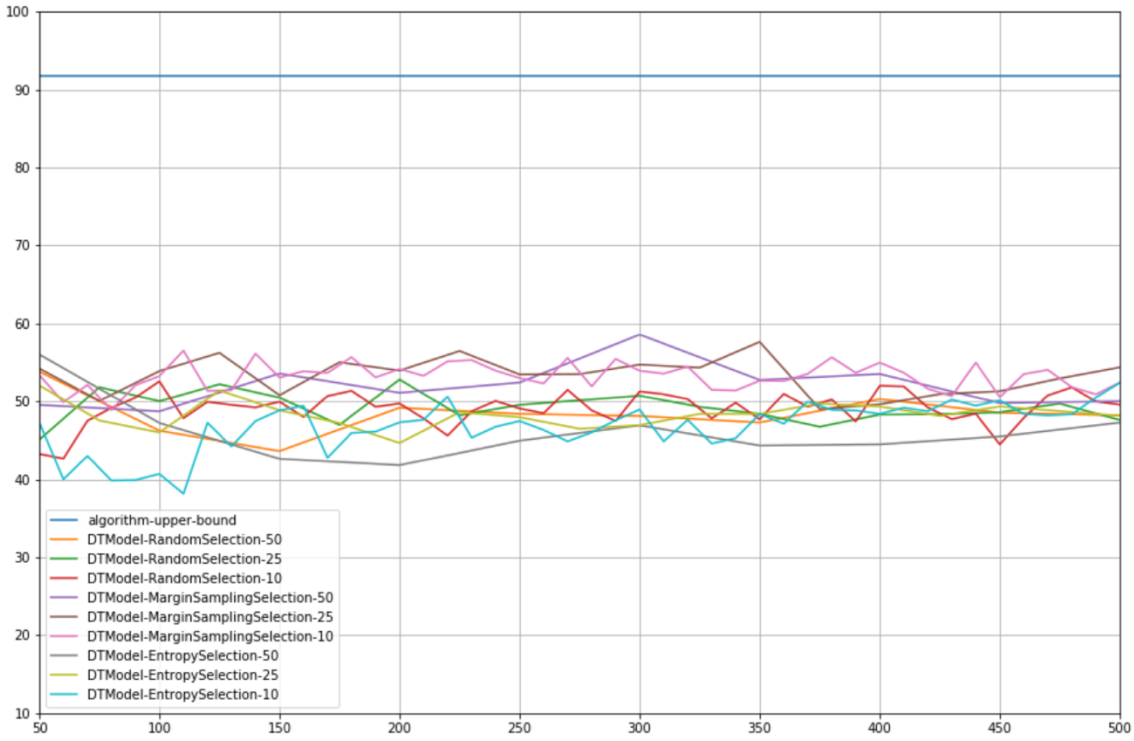
Binary Class: Random Forest with 40% Inaccurate Labels



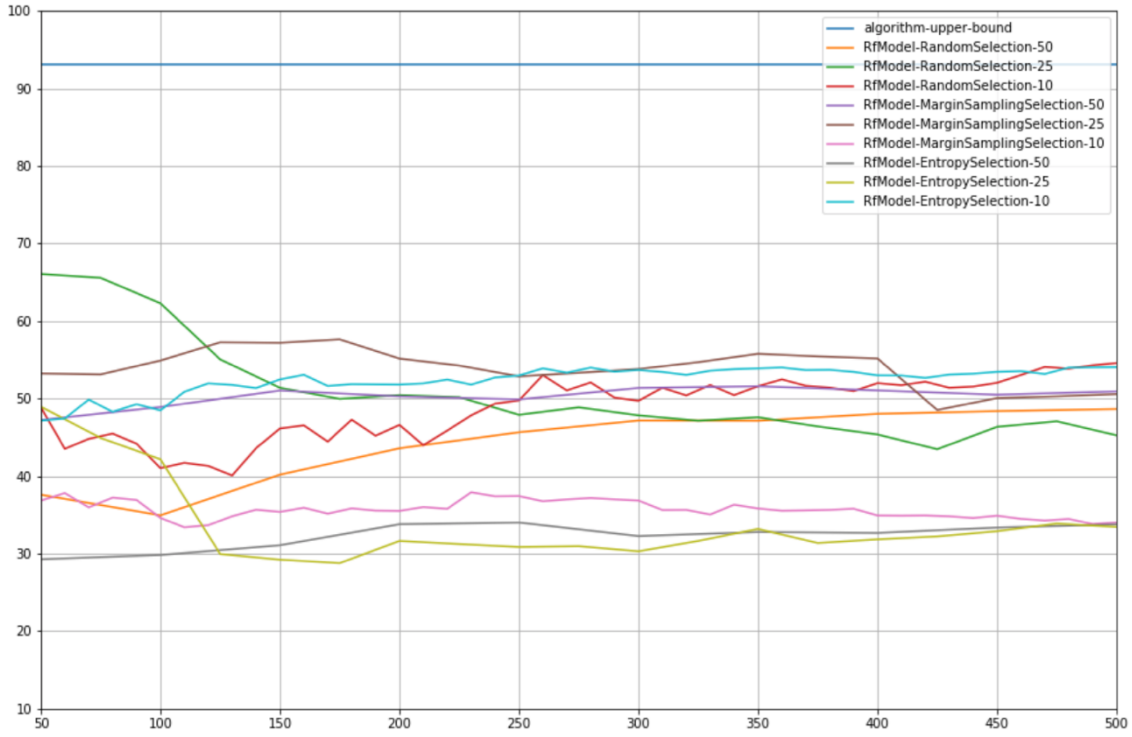
Binary Class: XGBoost with 40% Inaccurate Labels



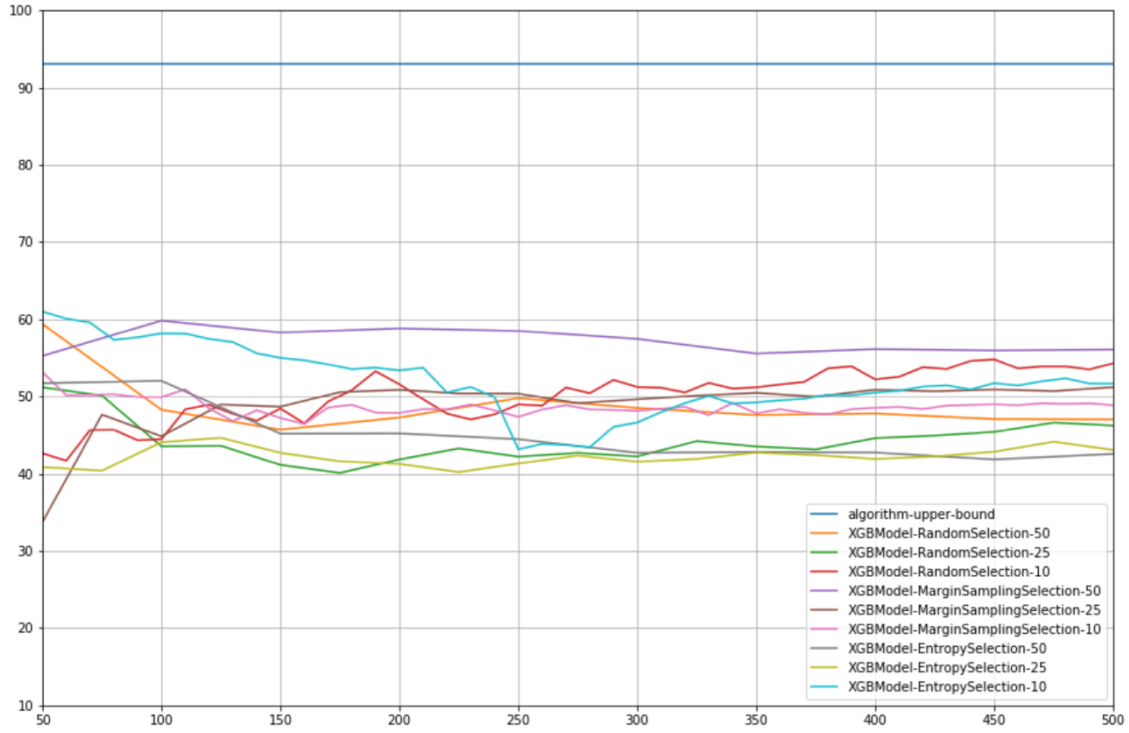
Binary Class: Decision Tree with 50% Inaccurate Labels



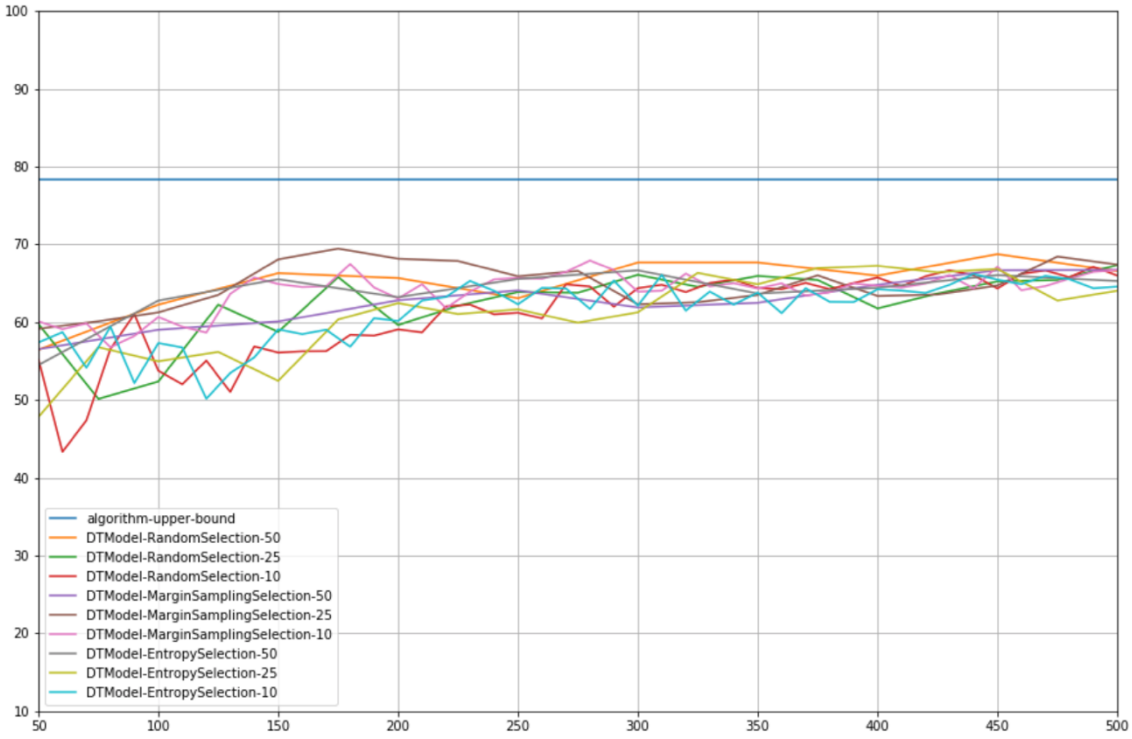
Binary Class: Random Forest with 50% Inaccurate Labels



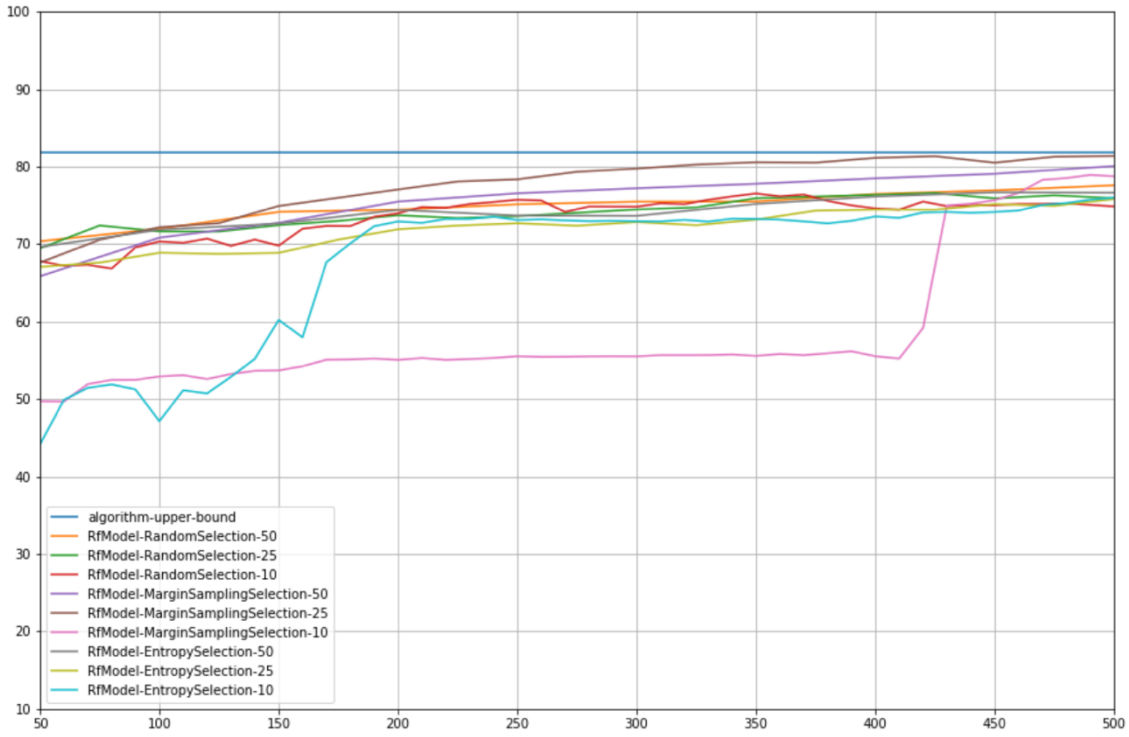
Binary Class: XGBoost with 50% Inaccurate Labels



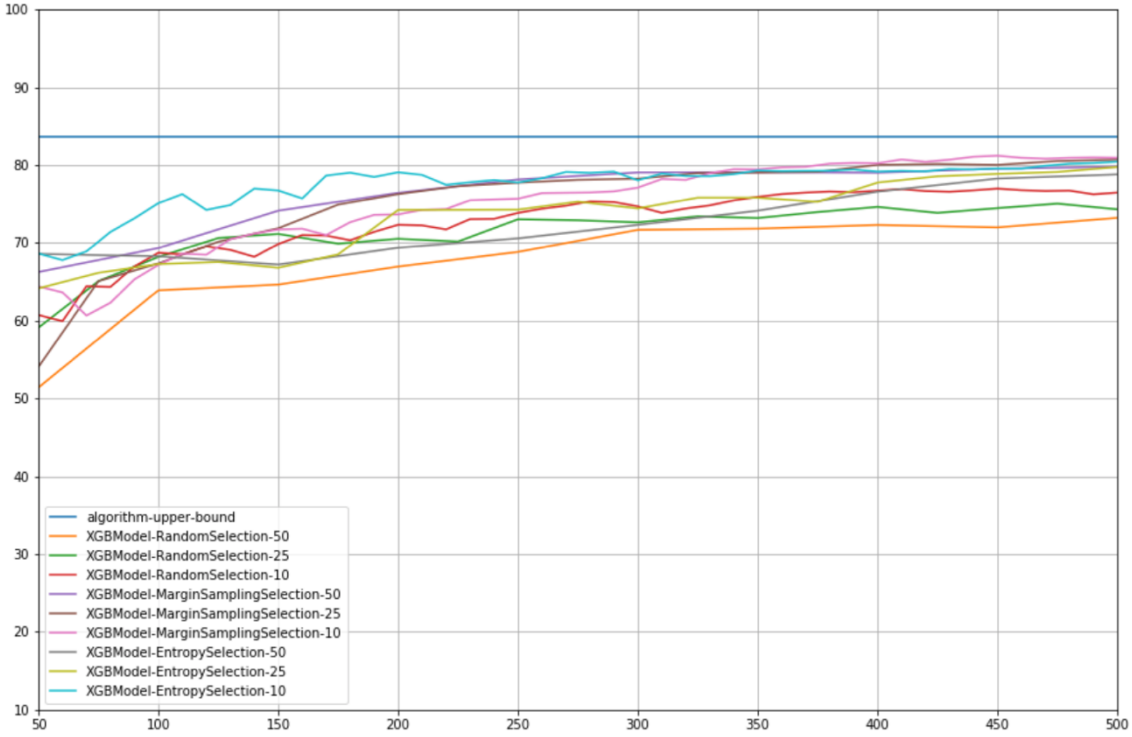
Multi Class: Decision Tree with 10% Inaccurate Labels



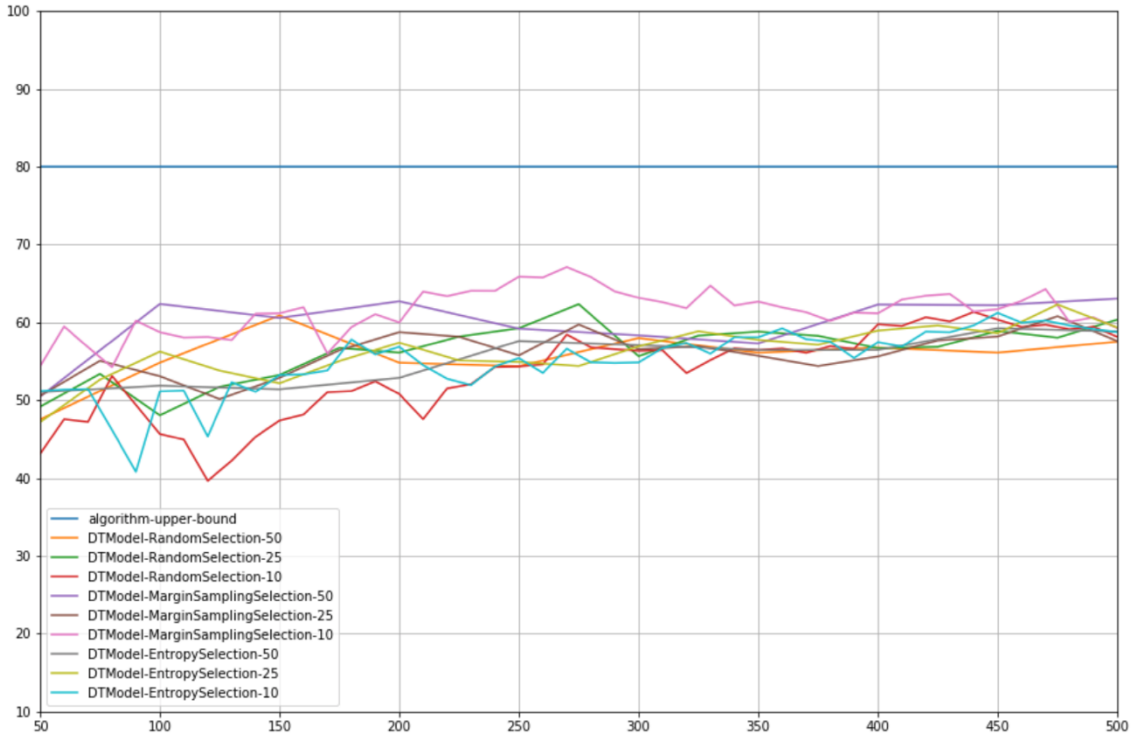
Multi Class: Random Forest with 10% Inaccurate Labels



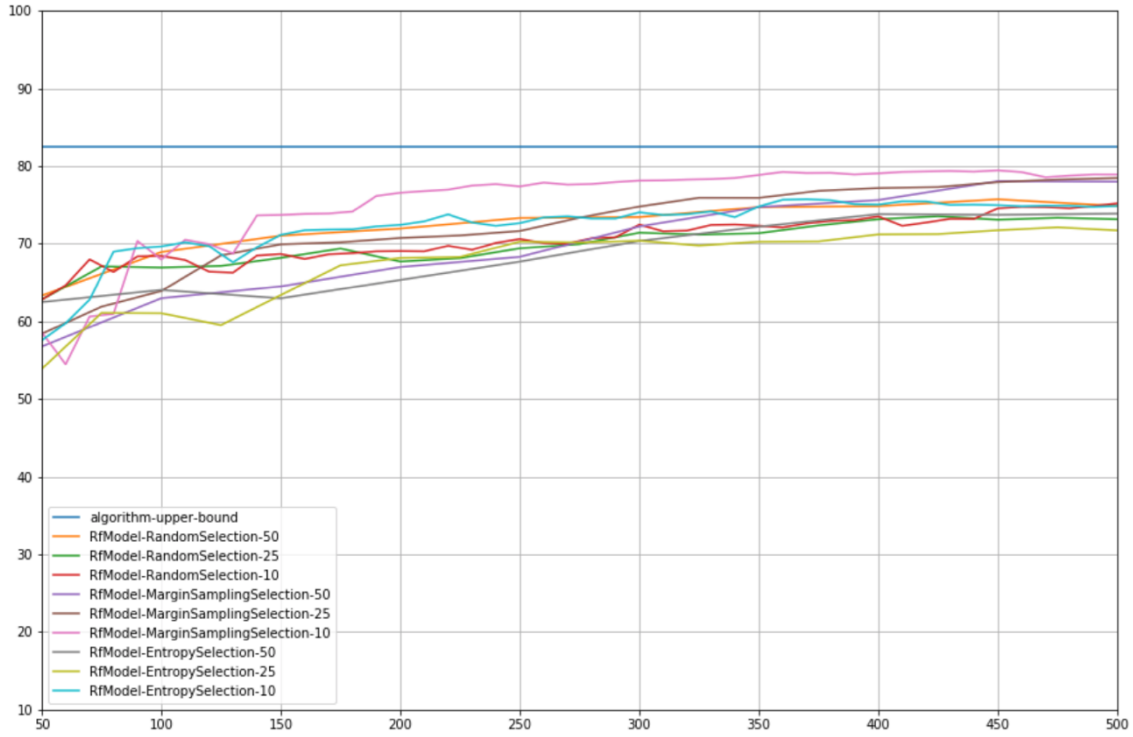
Multi Class: XGBoost with 10% Inaccurate Labels



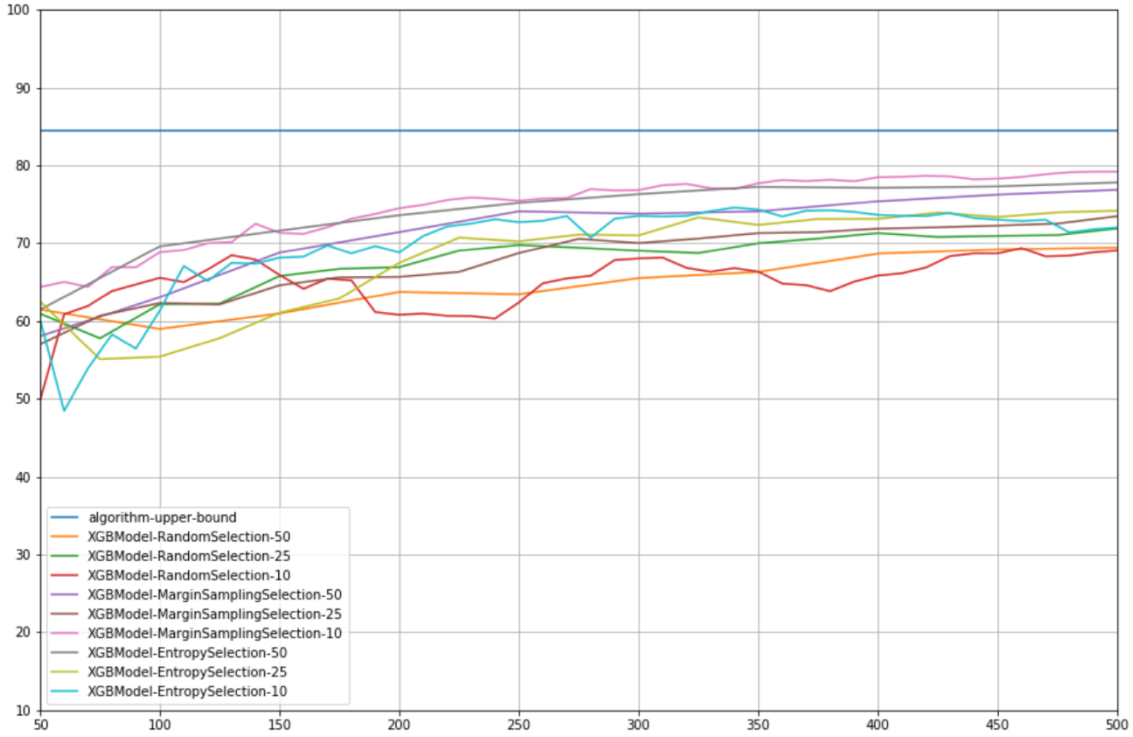
Multi Class: Decision Tree with 20% Inaccurate Labels



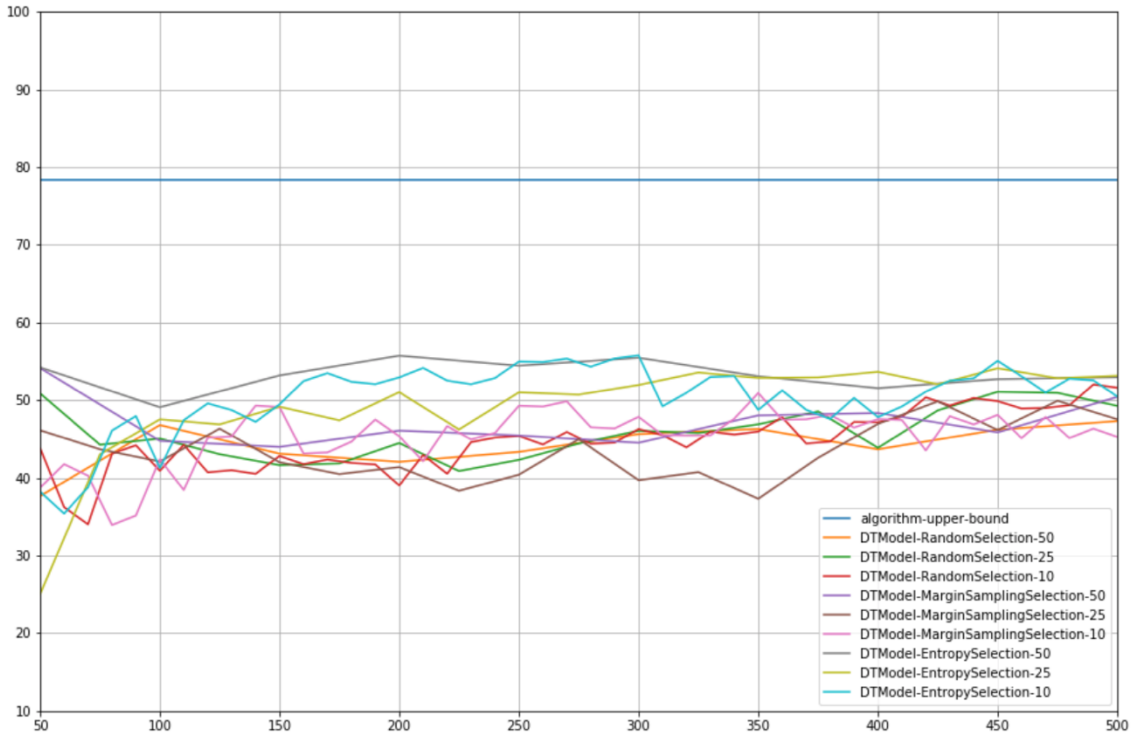
Multi Class: Random Forest with 20% Inaccurate Labels



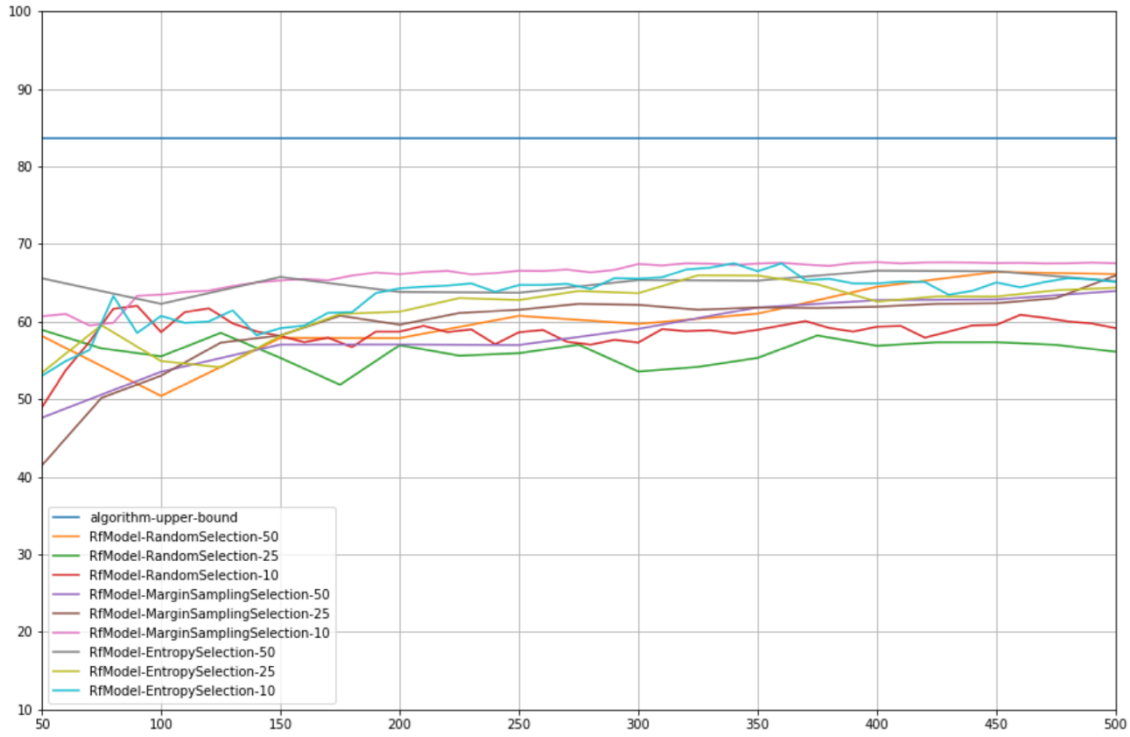
Multi Class: XGBoost with 20% Inaccurate Labels



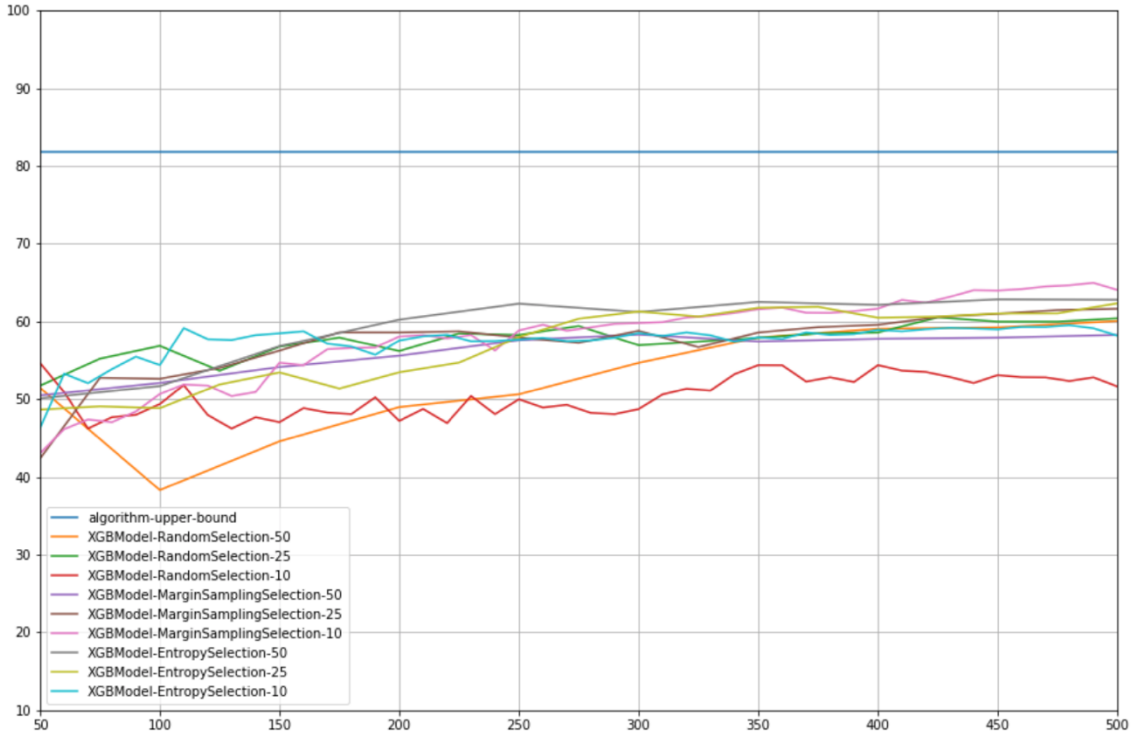
Multi Class: Decision Tree with 30% Inaccurate Labels



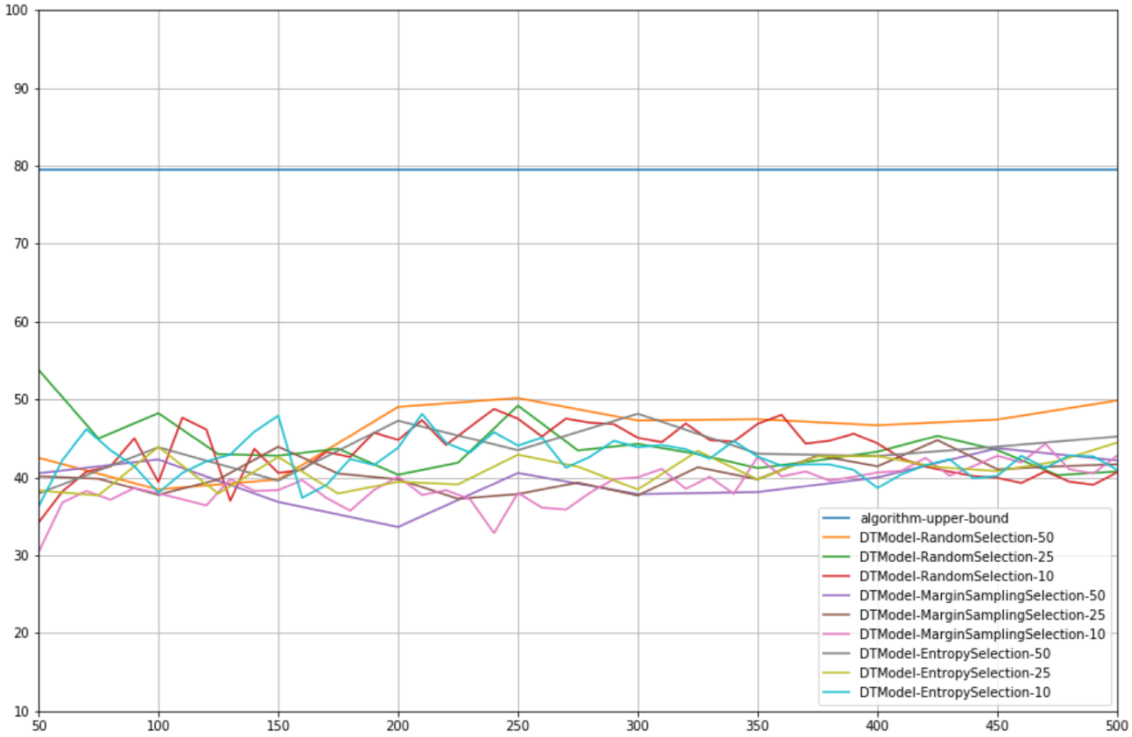
Multi Class: Random Forest with 30% Inaccurate Labels



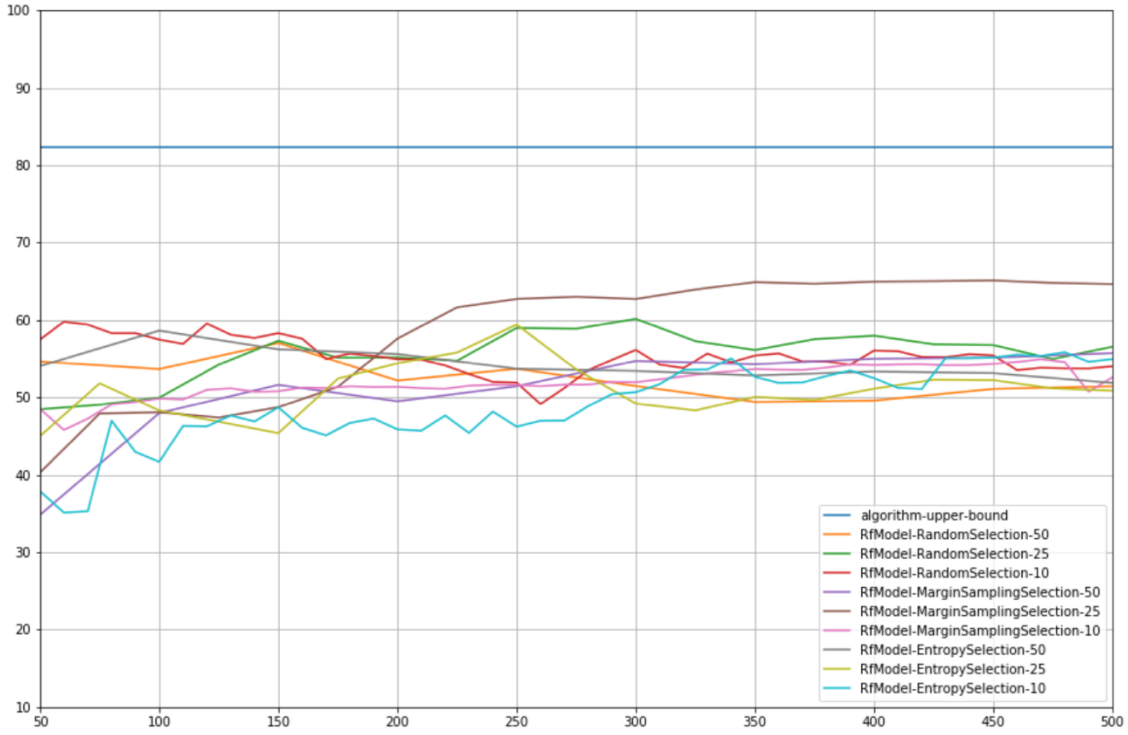
Multi Class: XGBoost with 30% Inaccurate Labels



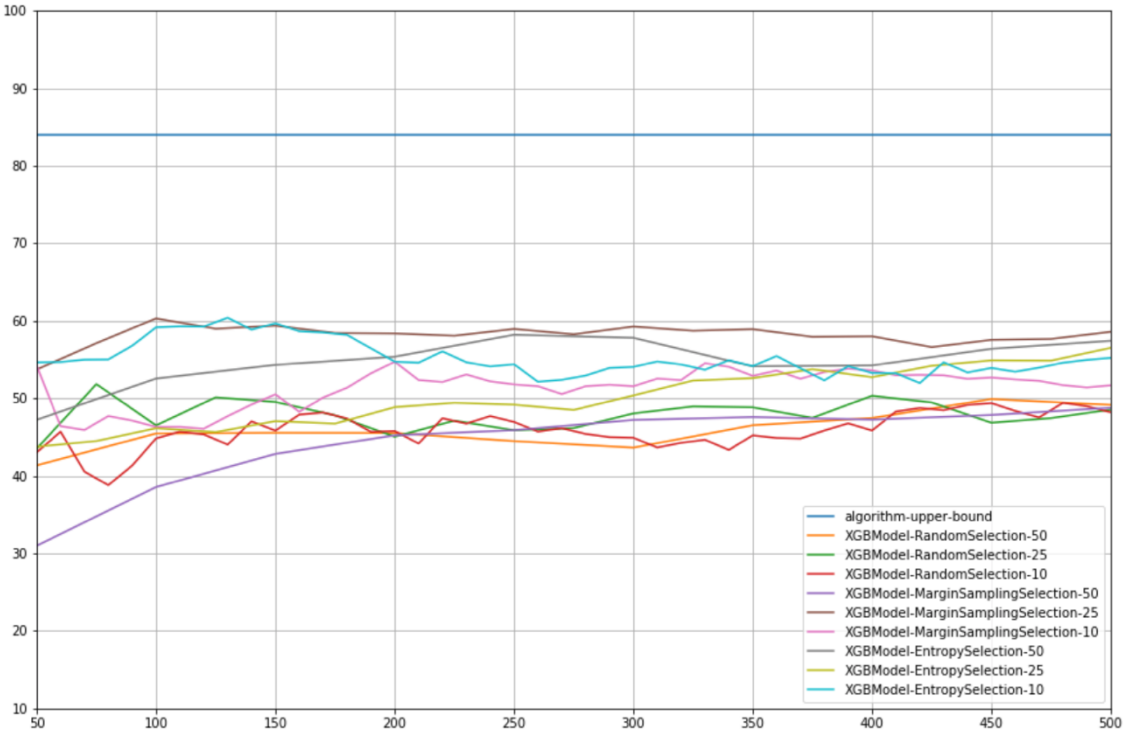
Multi Class: Decision Tree with 40% Inaccurate Labels



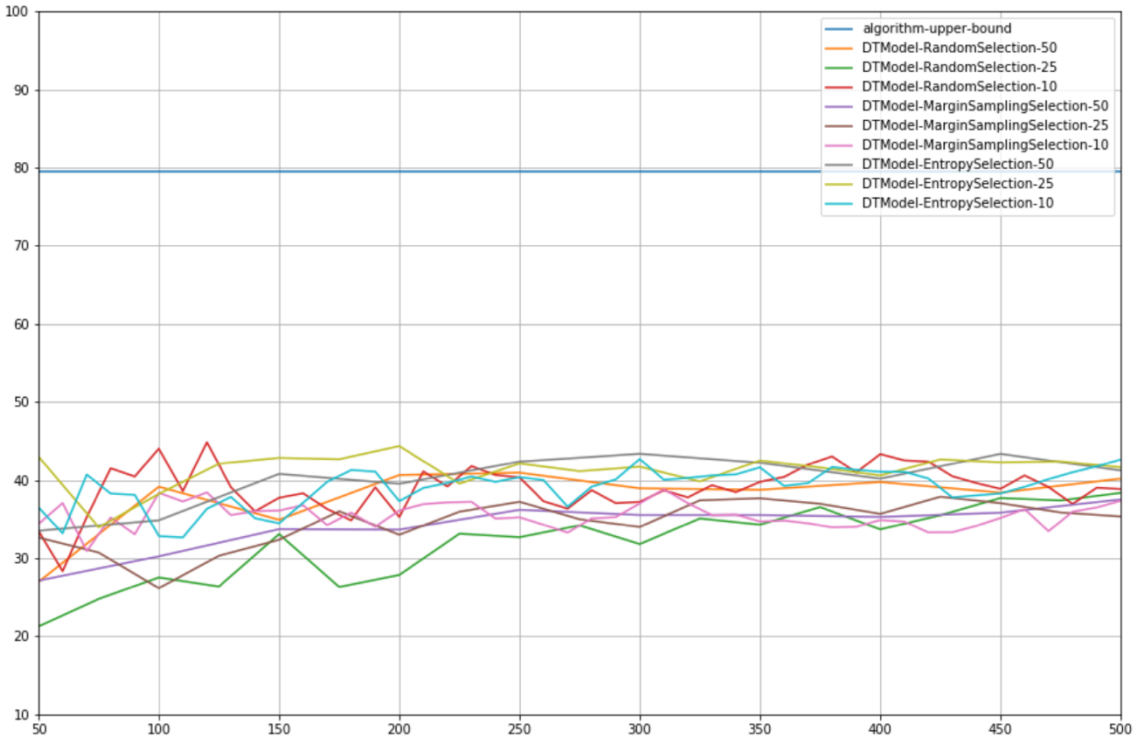
Multi Class: Random Forest with 40% Inaccurate Labels



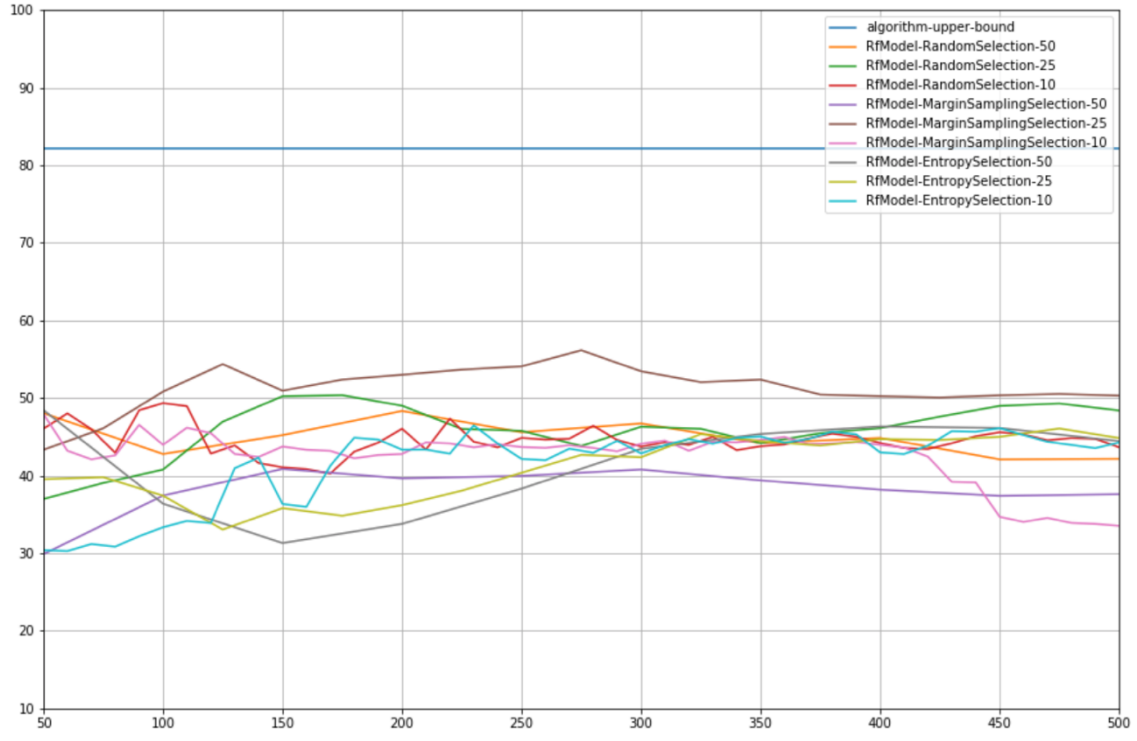
Multi Class: XGBoost with 40% Inaccurate Labels

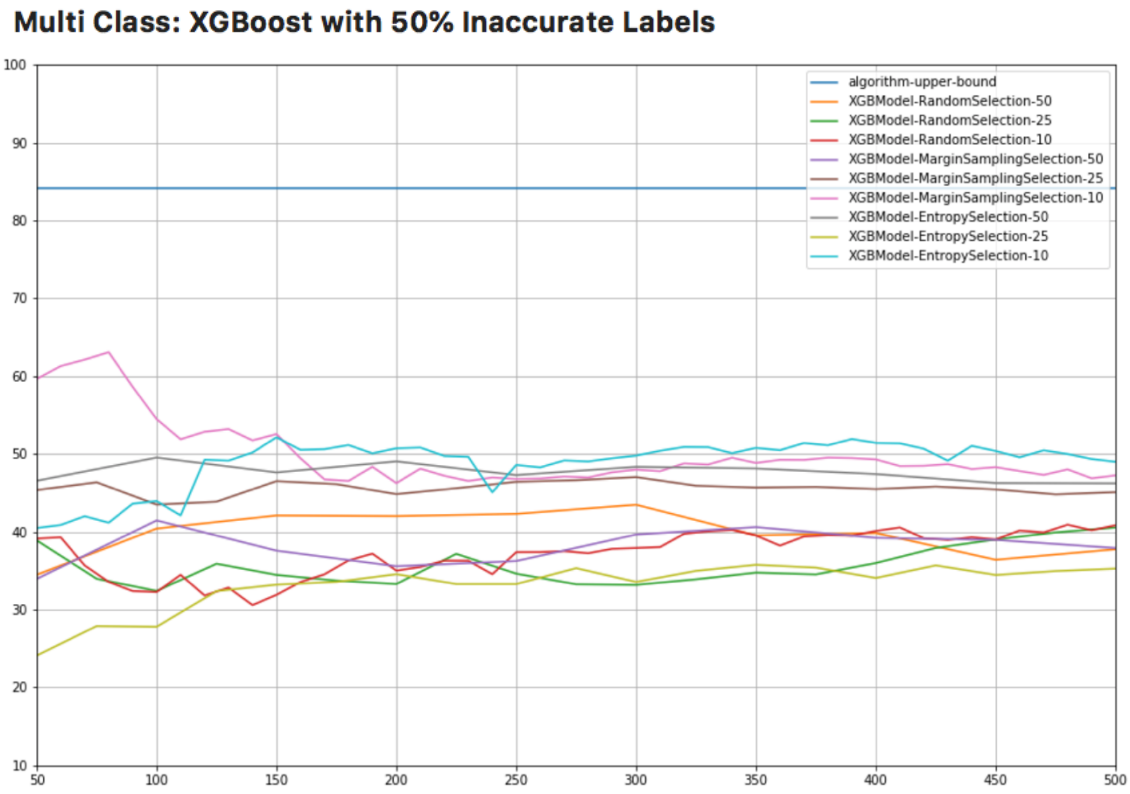


Multi Class: Decision Tree with 50% Inaccurate Labels



Multi Class: Random Forest with 50% Inaccurate Labels




Multi Class: XGBoost with 50% Inaccurate Labels
algorithm-upper-bound
XGBModel-RandomSelection-50
XGBModel-RandomSelection-25
XGBModel-RandomSelection-10
XGBModel-MarginSamplingSelection-50
XGBModel-MarginSamplingSelection-25
XGBModel-MarginSamplingSelection-10
XGBModel-EntropySelection-50
XGBModel-EntropySelection-25
XGBModel-EntropySelection-10
100
90
80
70
60
50
40
30
20
10
50
100
150
200
250
300
350
400
450
500

Appendix 7– Active Learning Evaluation Performance Results

In this section, classification performance both for binary class and multi class provided using accuracy, precision, recall and f1 scores. Results are calculated from the confusion matrix foe each label class. Since, there are too many results for each specific case, the results presented in this section were chosen randomly. All cases provide results for the 500 most informative samples queried during the active learning experiments.

Binary (correct label)	**Label**	**Accuracy**	**Precision**	**Recall**	**F_1-score**
Decision Tree	0	87.900530	86	88	87
	1		90	88	89
		weighted average	88	88	88
Random Forest	0	94.191303	93	95	94
	1		96	94	95
		weighted average	94	94	94
XGBoost	0	93.925848	93	93	93
	1		94	95	94
		weighted average	94	94	94
(0-normal, 1-botnet)					

Multi Class (correct label)	**Label**	**Accuracy**	**Precision**	**Recall**	**F_1-score**
Decision Tree	0	87.900530	86	88	87
	1		42	51	46
	2		11	28	16
	3		28	27	27
	4		54	44	48
	5		7	13	9

	6		20	15	17
	7		7	23	11
	8		10	21	14
	9		92	95	94
		weighted average	72	72	72
Random Forest	0	77.889636	82	95	88
	1		63	59	61
	2		12	4	6
	3		38	21	27
	4		64	65	65
	5		12	21	15
	6		21	4	7
	7		0	0	0
	8		14	11	12
	9		93	96	94
		weighted average	73	78	75
XGBoost	0	83.859860	92	97	94
	1		85	73	79
	2		12	17	14
	3		39	25	31
	4		63	73	67
	5		7	3	4
	6		65	51	57
	7		0	0	0
	8		37	24	29
	9		99	96	97
		weighted average	83	84	83
(0-normal, 1-fuzzers, 2-analysis, 3-backdoors, 4-dos, 5-exploits, 6-generic, 7-reconnaissance, 8-shellcode, 9-worms)					

Since 10% and 20% of the wrong labelling does not reflect visible changes in the overall classification performance 30% of wrong labeled training data for active learning

experiment chosen here to present results. All other cases are selected arbitrary, for the purpose of visual representation.

Binary (incorrect label)	**Label**	**Accuracy**	**Precision**	**Recall**	**F_1-score**
Decision Tree	0	61.809599	56	65	61
	1		68	59	63
		weighted average	63	62	62
Random Forest	0	76.927989	76	71	73
	1		78	82	80
		weighted average	77	77	77
XGBoost	0	72.178606	73	60	66
	1		72	82	76
		weighted average	72	72	72
(0-normal, 1-botnet)					

Multi Class (correct label)	**Label**	**Accuracy**	**Precision**	**Recall**	**F_1-score**
Decision Tree	0	50.463919	59	58	59
	1		65	40	49
	2		12	11	11
	3		23	19	21
	4		51	42	46
	5		1	1	1
	6		42	34	38
	7		0	24	0
	8		9	13	10
	9		98	59	73
		weighted average	63	50	55
Random Forest	0	65.114008	64	85	73
	1		83	36	50
	2		13	8	10
	3		30	14	19

	4		63	40	49
	5		12	4	6
	6		53	17	25
	7		0	0	0
	8		22	11	8
	9		99	78	87
weighted average			69	65	64
XGBoost	0	58.170868	57	72	63
	1		83	21	33
	2		10	3	4
	3		34	8	13
	4		53	19	28
	5		6	3	4
	6		53	17	26
	7		0	0	0
	8		0	0	0
	9		1	90	95
weighted average			65	58	58
(0-normal, 1-fuzzers, 2-analysis, 3-backdoors, 4-dos, 5-exploits, 6-generic, 7-reconnaissance, 8-shellcode, 9-worms)					