TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Viktoria Tisler, 176817 IAPM

FEATURE SELECTION FOR MACHINE LEARNING BASED IOT BOTNET ATTACK DETECTION

Master's thesis

Supervisors: Sven Nõmm, PhD

Alejandro Guerra Manzanares, MSc

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Viktoria Tisler, 176817 IAPM

TUNNUSTE ANALÜÜS MASINÕPPEL BASEERUVAS IOT BOTNET RÜNNAKUTE TUVASTAMISES

Magistritöö

Juhendajad: Sven Nõmm, PhD

Alejandro Guerra Manzanares, MSc

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: Viktoria Tisler

07.05.2019

Abstract

Nowadays IoT (*Internet of Things*) technology has earned a wide usage in both home and corporate networks and thus has led to the increase of botnet attacks due to wrong misconfiguration of smart devices that are parts of deployed IoT networks. Therefore, anomaly detection systems are crucial for timely preventing remaining IoT network devices from being exploited by botnet and thus reduce financial losses caused by recovering the network after attack. Machine learning and deep learning approaches have found a wide application in network traffic data analysis dedicated to anomaly detection.

The primary goal of this thesis is to demonstrate the importance of feature selection in simplifying machine learning models (interpretability) and boosting their performance. In the scope of current research, feature selection models were applied in classifying network data as normal or indicating being attacked by IoT botnet – Mirai or Gafgyt. Reducing the training data by selecting most appropriate features leads to deploying less complex machine learning models and to achieving results that are comparable with deep learning approach results in terms of accuracy. Moreover, this thesis demonstrates that it is possible to get optimal feature subsets using not only supervised learning based models trained on network traffic data containing both normal and anomalous data, but also using unsupervised learning algorithms being preliminarily fit on normal data only.

This thesis is written in English and is 75 pages long, including 13 chapters, 28 figures and 14 tables.

Annotatsioon

Tunnuste analüüs masinõppel baseeruvas IoT botnet rünnakute tuvastamises

Tänapäeval asjade Interneti tehnoloogia on laialdaselt kasutatav nii kodu kui ka korporatiivsetes võrgustikudes ja niiviisi mõjutab botneti rünnakute kasvu, mis on tingitud asjade Interneti võrgustikku kuuluvate nutikate seadmete vale konfiguratsiooniga. Seetõttu anomaalia tuvastamise süsteemid on olulised ülejäänud nutikate seadmete õigeagsel kaitsmisel botneti poolt ekspluateerimise vastu ja niiviisi vähendavad rahalist kahjumi mis on omapoolt tingitud võrgustiku taastamisega pärast rünnakut. Masin- ja sügavõppe lähenemisviisid on laialt rakendatavad võrgu liikluse analüüsis mis on anomaaliate tuvastamise osa.

Selle lõputöö esmane eesmärk on demonstreerida andmete tunnuste valiku olulisust masinõppe mudelite lihtsustamisel (tulemuste interpreteerimine) ja mudelite jõudluse tõstmisel. Käesoleva uuringu raames tunnuste valimise mudelid olid rakendatud võrguliikluse klassifitseerimisel kas normaalseks või rünnatud botnetiga Mirai või Gafgyt'i tüüpi. Treenimisandmestiku vähendamine sobivate tunnuste kasutamise abil viib vähima keerukusega masinõppe mudelite rakendamisele ja selliste tulemuste saavutamisele, mis on võrreldatavad sügavõppe lähenemisviisi tulemustega täpsuse suhtes. Peale selle, käesolev töö demonstreerib et see on võimalik valida optimaalseid tunnuste rühmi kasutades mitte ainult valvega õppimisel põhinevaid algoritme mis olid õpetatud nii normaalse kui ka anomaaliatega võrguliikluse andmetel, kui ka valveta õppivaid algoritme mis olid treenitud ainult normaalse võrguliikluse andmetel.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 75 leheküljel, 13 peatükki, 28 joonist, 14 tabelit.

Acknowledgment

I would first like to express my sincere gratitude to my supervisor Prof. Sven Nõmm for the continuous support of this MSc study, for his motivation, patience, and inspiration, co-supervisor Alejandro Guerra Manzanares for knowledge sharing, encouragement, and providing valuable comments on this thesis.

Finally, I would like to express my gratitude to family and friends, especially to my mother, boyfriend, and grandmother who have always provided me countenance.

List of abbreviations and terms

AUC	Area Under the Curve
DDoS	Distributed Denial-of-Service
DRDoS	Distributed Reflection Denial of Service
FN	False Negative
FP	False Positive
FSA	Feature selection Algorithm
IDS	Intrusion Detection System
IoT	Internet of Things
IQR	Interquartile Range
k-NN	k nearest neighbors
LIME	Local Interpretable Model-Agnostic Explanations
LOF	Local Outlier Factor
ML	Machine Learning
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic
RUS	Random Majority Undersampling
FN	False Negative
FP	False Positive

Table of Contents

Acknowledgment....6
1 Introduction....14
1.1 Overview....14
1.2 Network traffic anomaly detection background....16
1.2.1 Background on botnet attacks....16
1.2.2 Background on machine learning types....17
1.2.3 Background on ML models validation....19
1.2.4 Background on outlier and anomaly detection....22
1.2.5 Background on network-based anomaly detection....23
1.3 Problem statement....26
1.3.1 Motivation....26
1.3.2 Focus of the thesis....27
1.3.3 Methodology....28
1.4 Related work....30
2 Implementation....33
2.1 Implementation overview....33
2.2 Tools and technologies....38
3 Data pre-processing for supervised classification based and filter FSAs....40
3.1 Input dataset description....40
3.1.1 Original data set....40
3.2 Data cleaning....41
3.2.1 Class imbalance problem....41
3.2.2 Sample preparation....43
3.2.3 Data standardization....44
4 Data pre-processing for unsupervised anomaly detection....47
5 Feature selection....48
5.1 Filter model....48
5.1.1 Removing features based on Pearson's coefficient....49

5.1.2 Selecting features based on Fisher score ranking....51
5.2 Wrapper models....52
5.2.1 Wrapper model approach....52
5.2.2 Sequential forward feature selection....53
5.2.3 Sequential backward feature selection....53
5.3 Hybrid models....56
5.4 Ensemble models....57
6 Classifiers training for supervised learning....59
6.1 Methodology....59
6.1.1 Classifier algorithms....59
6.1.2 Validation of the classifiers....60
6.1.3 Classifiers training process....63
7 Anomaly detection algorithms training for unsupervised learning....65
7.1 Methodology....65
7.1.1 Anomaly detection algorithms....65
7.1.2 Anomaly detection algorithms training at wrapper feature selection stage....66
7.1.3 Validation of unsupervised feature selection wrapper models....66
8 Feature selection results based on filter models....68
8.1 Filter method based on Pearson's linear correlation coefficient....68
8.2 Filter method based on Fisher score....72
9 Feature selection results based on supervised learning....76
9.1 Feature selection with hybrid models....76
9.2 Features selection with ensemble models....78
10 Feature selection results based on unsupervised learning....82
11 Predictions interpretation with LIME....83
11.1 Methodology....83
11.2 LIME technique....83
11.3 LIME interpretation results....83
12 Discussion and future work....86
13 Summary....87
References....89
Appendix 1 – Sequential Forward Feature Selection....96
Appendix 2 – Sequential Backward Feature Selection....97

Appendix 3 – Heuristics Calculation For Wrapper Method.....98

Index of Figures

Figure 1: Confusion matrix for n-class classification, where n - number of classes, k - class in range $0 \leq k \leq n$.....20
Figure 2: The general structure of an autoencoder[32].....24
Figure 3: Data pre-processing strategy: cleaning, undersampling, and train / test splitting 42
Figure 4: Classes distribution before and after applying standardization.....46
Figure 5: Samples preparation flow for unsupervised anomaly detection.....47
Figure 6: Filter models workflow.....49
Figure 7: Wrapper feature selection algorithms.....55
Figure 8: Classifiers training procedure for hybrid feature selection.....56
Figure 9: Ensemble models workflow.....57
Figure 10: Train, validation, and test samples extraction overview.....61
Figure 11: Hybrid models cross-validation flow on the test set.....62
Figure 12: Classifiers cross-validation with and without applying feature selection techniques.....64
Figure 13: Training process for unsupervised anomaly detection algorithms.....66
Figure 14: Validation process for optimal subsets generated by unsupervised wrapper models.....66
Figure 15: Pearson's linear correlation heatmap for features that have remained after applying model with coefficient values threshold [0, 0.80].....70
Figure 16: Classes distribution plot for 3 best features from Fisher scoring rank - MI_dir_L0.1_weight, MI_dir_L0.01_weight, and H_L0.01_weight.....73
Figure 17: Classes distribution plot for features that have not been selected by filter methods.....74
Figure 18: Confusion matrix for subsets generated by hybrid model based on combination of sequential forward feature selection and Fisher score ranking threshold, random forest classifier accuracy 0.9990.....77

Figure 19: Confusion matrix for subsets generated by hybrid model based on combination of sequential backward feature selection and Fisher score ranking threshold, random forest classifier accuracy 0.9990.....77
Figure 20: ROC curve for classification on MI_dir_L3_weight and H_L0.1_weight features.....79
Figure 21: ROC curve for classification on H_L3_weight and H_L0.1_weight features 80
Figure 22: Decision boundary for random forest on the subset generated by ensemble model.....81
Figure 23: Decision boundary for random forest on the subset generated by ensemble model.....81
Figure 24: Confusion matrix for k-NN classifier on unsupervised wrapper model output subset.....82
Figure 25: Confusion matrix for random forest on unsupervised wrapper model output subset.....82
Figure 26: Sequential forward feature selection.....96
Figure 27: Sequential backward feature selection.....97
Figure 28: Heuristic calculation based on stratified k-fold cross-validation F1-score....98

Index of Tables

Table 1: Original dataset overview.....40
Table 2: Sample dataset with class labels to represent the dataset structure.....41
Table 3: Dataset overview after removing duplicated records.....43
Table 4: Features with linear correlation in range [0, 0.80].....69
Table 5: Features with linear correlation above 0.50 from the filtered set.....71
Table 6: Features with linear correlation in range [0, 0.001).....71
Table 7: Twenty features with the highest Fisher score in descending order.....72
Table 8: Comparison of cross-validation scores for algorithms performing on the whole data and the data reduced after applying filters.....75
Table 9: Comparison of predictions accuracies made by hybrid models.....76
Table 10: Comparison of predictions accuracies made by ensemble models.....78
Table 11: Comparison of predictions for optimal subset generated by unsupervised wrapper model.....82
Table 12: LIME explanation for predicting random instance belonging to Mirai class with random forest classifier.....84
Table 13: LIME explanation for predicting random instance belonging to benign class with random forest classifier.....84
Table 14: LIME explanation for predicting random instance belonging to Gafgyt class with random forest classifier.....85

1 Introduction

1.1 Overview

Nowadays IoT (*Internet of Things*) technology is part of people's lives, though many people using IoT refuse to change default credentials for smart devices, thus smart device default configuration became a vulnerability that is exploited by the malefactors [15]. Devices with the default authorization settings can be compromised by the botnet – group of the Internet-connected devices (hosts), where each of the host is running the software called "bot". The bot turns compromised smart devices into part of the remotely controlled botnet [15].

IDS (*Intrusion Detection System*) is a part of cyber security system that helps to identify unauthorized use, alteration, duplication, and destruction of information systems [40]. Misuse based, anomaly based, and hybrid based detection systems are common types of IDS. Misuse based detectors identify an attack based on its signature, whereas anomaly based detector identifies the deviation of the observed system behavior from the normal one. Analyzing network pattern of IoT devices is part of anomaly based cyber analytics [46]. Although vast majority of deployed detection systems use signature based approach, their main drawback is high false positive and false negative rate [4]. Problem of high false positive and negative rates might be solved using ML (*Machine learning*) approach.

ML is a set of methods for automatically detecting patterns in existing data and making predictions about future data [34]. There are two types of ML: supervised (predictive) and unsupervised (descriptive) [34]. In case of supervised ML, building ML model consists of training the model on data containing set of observations and automatically predicting target output based on previously collected knowledge. Target output can be of two types – categorical or real-valued. When dealing with predicting categorical output, this kind of task is called classification or pattern recognition, whereas if we

would like to find real-valued target output, this task is called regression. In case of unsupervised ML, the goal is to discover patterns in data without having target patterns to look for [34].

Scope of this thesis covers both supervised and unsupervised ML methods applied to the IoT devices network data analysis. Network pattern data in case of Mirai, Gafgyt, and normal traffic will be used in development of ML models for solving supervised classification task: the predefined set of parameters and their values captured at particular time should be classified as malicious (Mirai, Gafgyt) or normal (benign) with high accuracy [59]. In case of unsupervised ML approach, there are only two target outputs (classes) for the observations – normal (benign) and anomalous (malicious).

The primary goal of this thesis is to extract sets of the most relevant and interpretable features (also called *variables* or *attributes*) from already existing dataset containing normal and anomalous network traffic parameters values by applying different data reduction techniques, compare results achieved by several feature selection methods using classical ML models and statistical criteria and deep learning approach accuracies in classifying network data as benign (non-malicious) and malicious (indicating that the networked device is compromised by the IoT botnet of two types – Mirai and Gafgyt).

The optimal subsets of features will be selected using filter, wrapper, hybrid (combination of filter and wrapper), and ensemble models. Filter model is a features selection method that requires statistical evaluation criteria[18]. Wrapper model is a feature subset search that is *wrapped* around the learning classifier. Wrapper models can be combined with filter and thus form hybrid models [18]. Ensemble models use combination of different feature selection models outputs (optimal subsets) [13].

Novelty of this thesis is based on combining different feature selection models for extracting not more than 10 features and demonstrate that the results that are achieved by applying classical ML algorithms (learning algorithms for wrapper method and final validation on previously unseen data) and statistical approach (data standardization and irrelevant features elimination, selecting the most relevant features using filter methods) are trustworthy and comparable with the deep learning approach that needs more computational resources. Moreover, this thesis demonstrates that it is possible to boost classical ML models performance by selecting optimal attributes subsets using feature

selection based on unsupervised learning approach applied to the data that contains no contamination (benign records only).

This thesis is organized as follows. *Chapter 1* consists of background information, problem statement, and related work. *Chapter 2* provides feature selection models implementation overview. *Chapter 3* contains detailed description of data pre-processing for supervised learning based FSAs and filter models. *Chapter 4* contains detailed description of data pre-processing for unsupervised learning based FSAs. *Chapter 5* provides overview of all feature selection models that are used in the scope if this thesis. *Chapter 6* provides the training and validation strategy for supervised multi-class classification models, while *chapter 7* contains description of training and validation processes for unsupervised binary classification models. Chapters 8 – 10 provide results of FSAs. *Chapter 11* consists of LIME methodology description and interpretation results. Chapters 12 – 13 provide results overview and possible future work.

1.2 Network traffic anomaly detection background

1.2.1 Background on botnet attacks

DDoS (*Distributed Denial-of-Service*) – attack that consists of packet streams sent by different sources to the target (victim). DDoS goal is to consume particular resource that is critical for the victim and thus deny the service [19].

The botnet - the network of compromised Internet-connected smart devices that are running one or more bot. Aim of the bot is to propagate the infection from the networked devices that are wrongly configured straightly to the target network after receiving particular command from the malefactor [15].

There are four types of network architecture models that can be used to perpetrate a DDoS attack [3]:

- *Agent-Handler Model* – composed of clients, handlers (masters), and agents (daemons); attackers use clients when communicating with handlers – malicious software packages located in the Internet;

- *Reflector Model* – similar to the *Agent-Handler*, whereas uses additional group of devices called *reflectors* for sending a stream of packets against a victim; DDoS attacks using this model are also called DRDoS (*Distributed Reflection Denial of Service*) with lower traceability [2],[6],[7],[11];

- *Internet Relay Chat-Based Model* – model that is similar to *Agent-Handler*, whereas the client connects to the agents via IRC (I*nternet-Relay Chat*) communication channel;

- *Web-Based Model* – similar to IRC – model, but the communication protocol is HTTP/HTTPS based; the prevailing amount of agents are fully configured through PHP scripts.

Mirai – one of the most prevalent botnet malware with agent-handler architecture model; spreads by infecting such IoT devices as web cameras, home routers, DVRs (*Digital Video Recorders*) and many other smart devices that run some versions of BusyBox – Unix executable software. Vulnerable devices are mainly manufactured by XiongMai Technology [49]. Mirai launches a DDoS against multiple target servers by propagating via misconfigured smart devices with default credentials, thus this malware has been used in the largest botnet attacks [52]. Mirai is used in perpetrating several types of DDoS attacks exploiting such protocols as GRE, TCP, UDP, DNS and HTTP.

Gafgyt (also known as BASHLITE) – open-source botnet malware with lightweight IRC architecture model, but heavily modified thus Gafgyt botnet architecture becomes totally non-dependent on *IRC* servers. Gafgyt botnet attacks are of type SYN, UDP, and ACK flood [52].

1.2.2 Background on machine learning types

As mentioned in section 1.1, there are two main types of ML:

1. supervised (predictive) learning;

2. unsupervised (descriptive) learning – sometimes called *knowledge discovery* [34].

There is third less commonly used type of ML called *reinforcement learning* that is helpful when it is necessary to define behavior of software agents in an environment by bringing in reward or punishment signals [34].

Depending on the output that needs to be predicted, problems that might be solved by supervised learning algorithms are usually divided into the following types:

1. **classification** – ML approach of mapping a set of unlabeled inputs (data instances) to the categorical output variable called *class* – corresponding group membership for the single data instance [1]; for example, the data set[1] used in this thesis includes samples containing 115 variables (features) with their values, and each row of the data set file contains the class with its value– benign , Mirai, and Gafgyt, that are converted to the corresponding numeric value for simplicity – 0, 1, and 2; solving classification problem means predicting class labels to the unlabeled data records [34];

2. **regression** – predicting the target value of continuous output variable that is real-valued (integer or floating number) [34], for example – weight, price of the house, etc.

Classification tasks can be of the following types:

1. binary classification – there are two possible classes;

2. multiclass classification – there are more than two possible classes [34].

In the scope of this thesis two types of classification tasks are going to be solved:

1. anomaly detection as a binary classification: there are two target classes – benign (normal) and anomalous (malicious, i.e. compromised by Mirai or Gafgyt botnet);

2. multi-class classification: there are three target classes – benign, Mirai, and Gafgyt.

[1]https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT

1.2.3 Background on ML models validation

If we would like to evaluate the quality of predictions made by the algorithm, we will use separate *test set* that might be created from the original data set: the most common approach is to use 80% of the original data for training purposes (*train set*) and remaining 20% as test set[34].When we are talking about training a ML model, we mean that particular algorithm is trained on the *train set.* By saying that ML algorithm makes *predictions*, we mean that preliminarily trained ML model attempts to predict outputs for test set records – in other words, ML model attempts to *classify* particular record (row of the test set) [34].

In case there is a lack of training observations, it might affect the model performance. One of possible solutions is *cross-validation* – randomly splitting the dataset into *K* folds, train a model on each of *K* – 1 folds, then test on *Kth* fold [34]. K-fold cross-validation helps preventing the overfitting of the model. Cross-validation method is helpful in assessing the model quality when there is a lack of representative test data.

Stratified cross-validation is the variant of cross-validation where all *K* folds are formed by taking roughly equal proportions of each class, training the model *K-1* times on all splits except the last one (the *K*th split), and evaluation the model performance on the *K*th split [34].

There are different metrics used for evaluating the performance of ML models, the most commonly used ones are provided in the list below.

1. **Accuracy score (classification accuracy)** – in binary and multi-class classification this term means the ratio of correctly classified samples in the test set to total predictions amount; accuracy score often coincides with the *Jaccard index* that measures similarity between two set samples (in case of ML – training and test samples), and alternatively defined as the size of intersection divided by the size of the union of the sample sets[43] and is calculated according to the formula (1):

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{|A\cap B|}{|A|+|B|-|A\cup B|} \quad (1)$$

The most common equation used for calculating classification accuracy as follows (2):

$$Accuracy = \frac{correct\ predictions}{all\ predictions} \tag{2}$$

2. **Confusion matrix** - visualization of the predicted and actual classification results in the form of table with size n x n, where n is a number of classes [57]. The confusion matrix represents the way the model is confused when making predictions [57]. There are four different values that can be obtained from the confusion matrix:

   - TP (*True Positive*) – correct positive predictions,
   - FP (*False Positive*) – incorrest positive predictions,
   - TN (*True Negative*) – correct negative predictions,
   - FN (*False Negative*) – incorrect negative predictions.

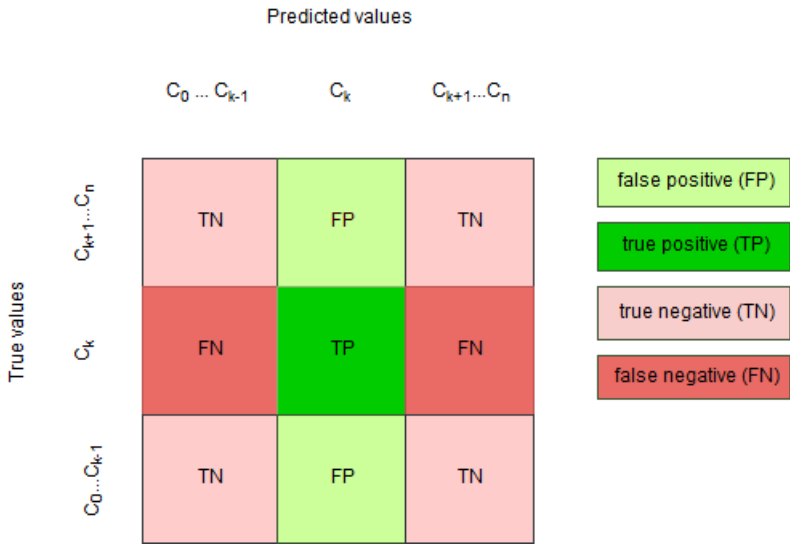

Figure 1: Confusion matrix for n-class classification, where n - number of classes, k - class in range $0 \leq k \leq n$

3. **Precision** (also called **confidence**, **positive predicted value,** or **true positive accuracy**) – the proportion of predicted positive cases [45] that follows the equation (3):

$$Precision = \frac{TP}{Predicted\ Positives} = \frac{TP}{TP + FP} \tag{3}$$

4. **Recall** (also called **sensitivity**) – TPR (*True Positive Rate*) that is calculated according to the formula (4):

$$TPR = Recall = \frac{TP}{Real\ Positives} = \frac{TP}{TP + FN} \tag{4}$$

5. **PR (*False Positive Rate*) –** also called type I error rate, calculated according to the equation (5):

$$FPR = \frac{FP}{Real\ Negatives} = \frac{FP}{FP + TN} \tag{5}$$

6. **ROC (*Receiver Operating Characteristic*) curve –** 2-dimensional plot, where x axis (independent variable) is the FPR, y axis is the TPR; each point of the ROC scape represents a pair of the data TP and corresponding FP rates. Perfect result is achieved in the point (FPR = 0, TPR = 1) in case when system is able to perfectly separate the positive values from the negative ones [60]. The quality of ROC curve can be evaluated using AUC (*the Area Under the Curve*) value. The higher AUC value is, the better model is; the maximum value of AUC is 1 [34].

7. **F1-score -** harmonic mean of precision and recall that can be found according to the equation (6):

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{R + P} \tag{6}$$

Applied to multiclass-classification, F1 score can be generalized in two ways:

a) **macro-averaged F1** that is suitable for distinguishing one class among other ones when dealing with balanced dataset [34]; macro-averaged F1 score is calculated according to the equation (7) [61]:

$$macro-averaged\,F1=\left(\sum_{j=1}^{M}\frac{2P_jR_j}{P_j+R_j}\right)/M \quad , \tag{7}$$

where:

- M – number of classes,
- j – the individual class that belongs to the M set of classes,
- P – precision,
- R - recall

b) **micro-averaged F1 –** F1 score that is defined as pooled predictions across classes [61] and is calculated according to the equation (8):

$$micro-averaged\,F1=\sum_{j=1}^{M}\mu_j\theta_{jj} \quad , \tag{8}$$

where:

- M – number of classes,
- j – the individual class that belongs to the M set of classes,
- θ_{ij} – the probabilities that each of the test samples is classified into different classes,
- $\mu = (\mu_1, \ldots, \mu_M)$ – the probabilities that each test record truly belongs to each class.

1.2.4 Background on outlier and anomaly detection

Outlier detection is a data mining task with aim to uncover abnormal knowledge within all gathered observations related to a particular event [42]. Outlier detection main goal is to find patterns in data or single data points (instances) that do not fit the expected normal behavior [17]. Outlier detection has such application domains as intrusion detection in cyber security, fault detection in critical systems and many others.

Anomalous patterns are referred to as outliers, anomalies, novelties (new observations), faults, exceptions.

Outlier is a pattern in data that does not fit the normal data pattern. Reason of the outlier in the network traffic data that is being discussed in the following paper is botnet attack [18].

Outlier detection technique is a specific approach chosen for solving the task of detecting the outliers in the certain data [44].

Outlier score is the degree to which the pattern is considered as outlier; outlier score can be used in several outlier detection techniques [18].

There are two types of anomaly detection:

a) **supervised anomaly detection** – anomaly detection with classification approach that requires labeled training set that contains both anomalous an normal samples, and unlabeled test set for assessing a trained model; classical ML algorithms that are commonly used for training are k-NN, decision trees, and SVM [44];

b) unsupervised anomaly detection – anomaly detection that is based on assumption that the minority of the network traffic is anomalous, thus unsupervised technique does not need the training set; the network data is grouped to the normal records (vast majority of samples) and to the anomalous records that differ from the prevalent network traffic pattern [44].

1.2.5 Background on network-based anomaly detection

One of the methods for detecting IoT device that is connected to the corporate network and compromised by a botnet is called *autoencoder*. Autoencoder is a neural network that is trained to reconstruct the input [32].Autoencoder has a hidden layer **h** that describes used for representing the output. Network of an autoencoder consists of two parts: an encoder function $\boldsymbol{h}=f(\boldsymbol{x})$ and a decoder function for reconstruction $\boldsymbol{r}=g(\boldsymbol{h})$ (see Figure 1). Encoder function converts input data into another format, decoder function

attempts to reconstruct original format of the data by decoding given representation [32].

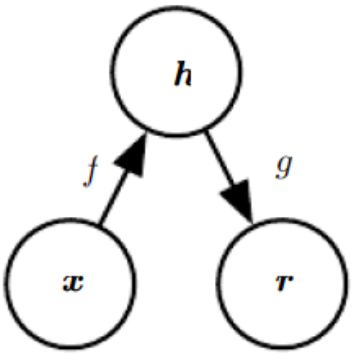

Figure 2: The general structure of an autoencoder[32]

The autoencoder with more than one additional hidden layer is called *deep autoencoder* [32]. Deep autoencoder as a deep learning technique can be used for finding compromised smart device with the help of network traffic data analysis taking into account benign, i.e. normal network traffic data pattern [59]. Deep autoencoder is trained to reconstruct network traffic pattern [59], [58]. Previously trained on normal IoT network behavioural traffic, deep autoencoder captures observed snapshot, attempts to compress and to reconstruct it. Failure in observed snapshot reconstruction indicates that IoT network traffic of the particular device is anomalous.

Anomaly detection method proposed by [58] consists of the following steps and are described further:

- data collection (normal and malicious network traffic data collection);
- features extraction;
- training an anomaly detector;
- continuous monitoring of the model

Step 1. Data collection

The typical normal behavior and botnet attacks were simulated by [58] in lab for collecting raw network traffic data on nine IoT devices in the most recent five time windows – 100 ms, 500 ms, 1.5 sec, and 1 min. The data was collected by port mirroring. For guaranteeing that the data is pure and contains no anomalous contamination, normal traffic of each smart device was collected immediately after installation in the deployed network. The data set created by [58] contains 502,605 normal, 2,835,371 BASHLITE, and 2,935,131 Mirai records.

Step 2. Features extraction

115 traffic statistics features were extracted over five time windows by taking a snapshot after receiving each packet from particular host [58].

Each smart IoT device data set has 115 features related to the stream aggregation, such as:

- the statistics extracted from the packet stream,
- the statistics summarizing channel jitter,
- time-frame indicating how relevant the observation is,
- the statistics summarizing the recent traffic from the packet's source IP, source MAC-IP, socket, and from source IP to the destination host IP address.

Step 3. Training an anomaly detector

The autoencoder is trained on normal behaviour instances for further recompression of its inputs – in case the recompression fails, it indicates that the input data is malicious as it contains abnormal observations compared to the benign ones [58].

Step 4. Continuous monitoring of the model

The optimized model was applied to features vectors extracted from packets for labelling each instance as benign (normal) or anomalous (malicious). Then observed packets instances sequence was marked as normal or malicious, thus it is possible to use

model for deciding whether the particular IoT device connected to the network is sending malicious data stream or not [58].

1.3 Problem statement

1.3.1 Motivation

According to the study [38], the number of deployed IoT (Internet of Things) devices dramatically increases worldwide. Interest towards IoT devices has led to the increase of vulnerabilities rate and occurrence of huge botnet attacks. Botnet is the group of different compromised Internet-connected smart devices where each of these devices is running special software called *bot.* The bot helps the outside attacker to keep compromised devices (hosts) under the remote control. Due to exposing IoT devices environment infrastructure, such malwares as the Mirai, BASHLITE, their variants and many other malware types are able to infect smart devices and launch distributed denial-of-service (DDoS) attacks [15].

According to recent studies, ML and deep learning approaches have demonstrated high prediction accuracies in classifying network traffic data as benign or malicious [30], [54]. Although nowadays there is a lack of solutions that are adopted to the bigger IoT environments, such as corporate networks, device-based network traffic anomaly detection using ML approach is a promising field due to its capability of learning complex network traffic patterns and detecting anomalies. Challenges of ML based botnet attacks detection that need solving are scalability to the bigger size of network traffic, reducing computational power consumption, encompassing various attack types, multi-class classifying traffic aggregation as benign or malicious, achieving ML model outcomes high interpretability by simplifying the fitted model in order to avoid further investments when deploying anomaly detection model in real IoT operational environments.

One of recent studies [54] was dedicated to comparing deep learning and ML alhorithms performances after selecting 2, 3, and 10 best features based on Fisher score ranking (filter method), and the results have demonstrated that it is possible to achieve even better performance with applying feature selection methods combined with

classical ML algorithms and without applying deep learning models that are more complex for deployment in the future.

Present thesis is a part of research series related to botnet attacks detection based on network traffic pattern analysis [58],[54]. The main focus of this thesis compared to the previous studies is boosting classical ML models performances using different feature selection algorithms.

1.3.2 Focus of the thesis

The main goals of this thesis are as follows:

- analyze network traffic behavioural patterns and select the most relevant features to reduce dimensionality, boost the performance of classifiers and anomaly detection algorithms, and achieve higher interpretability of the results;

- compare classical ML models performances in ability to classify network data as benign (normal), Mirai, or BASHLITE botnet attack; classifiers that are going to be compared are random forest, extra trees classifier, and k-NN (*k-nearest neighbors*);

- compare the results achieved by supervised ML approach combined with different features selection algorithms with the results achieved by unsupervised learning approach, evaluate chosen models quality using *precision and recall* metrics adjusted to multi-class classification [34], and interpret predictions using LIME (*Local Interpretable Model-Agnostic Explanations*) technique [8];

- novelty of the ML model should be achieved by at least one of the factors listed below:

  - spending less computational resources while training the model on large datasets;

  - reduce dimensionality and achieve better results interpretability by extracting the most relevant features [34]; total amount of features is 115, whereas it is necessary to select not more than 10 features by reducing redundant attributes in

order to spend less computational resources while training the models in the future when anomaly detection system will be adopted to the bigger environments, get the higher prediction accuracy compared to predicting accuracy on the whole data set (or at least nearly the same in case of high accuracy for training on the whole dataset containing all features; agreed expectation for predictions accuracy is at least 0.90).

1.3.3 Methodology

Dataset consists of nine different IoT devices network traffic data files – each IoT device has dataset file containing benign, i.e. normal network traffic data, and dataset files containing malicious traffic data related to the most common botnet attacks – Mirai and Gafgyt (also known as BASHLITE) malware families [58],[59]. Real network traffic data was collected by infecting nine commercial smart applications related to different kinds of devices, such as doorbell, security cameras, and thermostats.

The data set created by [58] provides opportunity to model multiclass classificator – besides classifying a sample as benign, Mirai, or Gafgyt, it is possibly to detect more specific attack. There are 10 types of botnet attacks that can be detected using network traffic data:

- Gafgyt attacks – scanning the network for device vulnerabilities, sending spam data, UDP and TCP flooding, sending spam data to a particular IP address;
- Mirai attacks – automatic device vulnerabilities scanning, Ack, Syn, and UDP flooding, UDP flooding optimized for higher packets delivery rate [58].

Labeled training and test datasets will be combined from initial datasets related to benign, Mirai, and Gafgyt using the following approaches:

- train / test split – splitting the dataset to train (80%) and test (20%) subsets using random permutation; train split is further used for features selection models training, while the test split will be used for final assessment of the models quality;

- stratified k-fold with 3 folds (k=3) on the train set used for evaluating performance of ML algorithms on different features subsets when constructing feature selection models, 2 folds are used as training set, third one used as validation fold for choosing candidate subset of attributes;

- stratified k-fold with 3 folds on the test set - 2 folds taken from the test set will be used as the train set, and the 3th fold will be used for final evaluation of the models quality based on *cross-validation accuracy*

Features selection applied to supervised learning (3-class classification) will be done based on the following techniques and their combinations:

- filter method using Pearson's linear correlation coefficient and Fisher score that expresses feature discriminatory power;

- wrapper method – features extraction using greedy sequantial forward and greedy sequential backward features elimination algorithms;

- hybrid method – combination of filter and wrapper methods for selecting optimal features subsets;

- ensemble method – combination of all previous models outputs (found subsets of features).

Multiclass classification problem will be solved using classical ML algorithms, such as random forest, extra trees classifier, and k-NN (k-nearest neighbors).

Unsupervised anomaly detection will be done using LOF (*Local Outlier Factor*).

Feature selection models performance will be evaluated using *precision and recall* metrics: precision is a fraction of relevant instances among the retrieved instances, recall is the fraction of relevant instances among total amout of relevant instances. The classification outcomes may be further interpreted as true positive, true negative, false positive, and false negative. Precision-recall approach is suitable for current task due to a large skewness in class distribution of dataset [21]. Moreover, it is crucial to assess trustworthiness of results when reducing dataset and thus lowen the risk of classifiers to be confused in distinguishing one class from other ones.

1.4 Related work

High dimensional data consists of attributes that can be irrelevant or containing the similar information that other attributes already have, which leads to problems in making trustworthy predictions. When training the model on data set for classification or anomaly detection purposes, it is essential to take into account dimension of data (features amount) for gaining the efficiency in training the model, boosting the classifier or outlier detector performance, and correctly interpreting results. FSA (*Feature Selection Algorithm*) is a model for selecting attributes based on their relevance [16]. Related studies [24] have shown that traditional detection techniques often fail on the multi-dimensional data due to the curse of dimensionality, thus it leads to the question: how to select optimal set of the most relevant features and choose the most appropriate FSA when solving classification and anomaly detection tasks in case of high-dimensional data?

The majority of studies have shown that the most of anomaly detection systems are based on particular ML methods for distinguishing between a normal and anomalous patterns of observed traffic [35]. Although these methods vary in subsets of extracted attributes, they are still based on the same concept of using particular criteria for finding dissimilarities between normal and anomalous patterns [47]. In most cases it is difficult to select optimal and at the same time trustworthy set of relevant features for improving performance of ML model in solving anomaly detection task.

FSAs can be characterized as follows [41]:

1. by search organization (exponential, sequential, random);
2. by features generation (forward, backward, compound, weighting, random);
3. by evaluation measure (divergence, accuracy, consistency, information, dependence, distance metric).

One of possible solutions is to have a learning algorithm itself for selecting relevant features automatically [35]. The problem of automatic features selection has been studied in the context of classification problem (assigning correct label to the particular observation of possible 2, 3, or more labels) by [1], [27], [28], and [29]. However, the classification setup is hardly appropriate for selecting features in anomaly detection problem.

When dealing with anomaly detection, we are dealing with one-class-classification, kind of anomaly detection using SVDD (*Support Vector Data Description*) when only single feature mapping is given [20]. The solution proposed by [35] offers extended version of SVDD when several feature mappings are given, and the objective is to learn a linear combination of attributes mappings from a particular subset.

Other studies have demonstrated that a lot of effort have been performed for features selection and extraction using mRmR (*Minimal Redundancy and Maximal Relevance*), RELIEF, CMIM (*Conditional Mutual Information Maximization*), Correlation Coefficient, BW-ratio (*Between-Within Ratio*), INTERACT, GA (*Genetic Algorithm*), SVM-RFE (*Recursive Feature Elimination*), PCA (*Principal Component Analysis*), Non-Linear Principal Component Analysis, Independent Component Analysis, and Correlation based feature selection [41]. L. Ladha et al [37] have presented an empirical comparison of different feature selection algorithms.

PCA is a non-parametric method that is used for transofrming the data by reducing the dimensional space and constructing the features that better represent the pattern and the observed variability in data [53]. PCA techniques fit better for approaches that are noise-tolerate when the data has a linear correlation [41].

CMIM selects featu0res subset based on the maximum relevance to the target class, thus applying CMIM is relevant when having both the features values and binary classes [41]. Correlation Coefficient method evaluates how well an individual feature influences the classes separation [41]. BW-ratio uses the ratio of between group to within group sums of squares for each feature, and allows to select the feature wih the maximum value of BW-ratio [41].

INTERACT methods take into account feature interaction with measurement of consistency contribution [41].

Genetic algorithm is a randomized approach that contains particular class of evolutionary algorithms and inspired by evolutionary biology (inheritance, mutation, selectiong, crossingover, etc.). This approach is suitable for features selection in pattern recognition, combinatorial optimization, and neural networks application [48].

SVM-RFE is a wrapper method performing backward elimination; applied in microarray gene expressions [33].

The mRmR method uses mutual information (MI) of two randomly selected features. MI is the quantity that measures the mutual dependency of two features [41].

2 Implementation

2.1 Implementation overview

Implementation consists of several stages that are listed below:

1) Data pre-processing for supervised classification
    1) Input dataset description
    2) Data cleaning:
        - cleaning original dataset from duplicated records
        - resolving class imbalance problem
        - undersampling
        - data standardization
2) Data pre-processing for unsupervised anomaly detection
    - sample preparation
3) Feature selection
    a) Filter models:
        - Fisher score
        - Pearson’s linear correlation coefficient
    b) Wrapper models:

- forward feature selection
- backward feature selection

c) Hybrid models:

- combination of wrapper models and filter models

d) Ensemble models:

- intersection of optimal subsets found by wrapper models with 20 best features from Fisher score ranking;
- intersection of optimal subsets found by wrapper models with features that have remained after applying data reduction method based on Pearson's linear correlation.

4) Classifiers training for supervised learning

a) Methodology overview:

- classifier algorithms
- classifiers training process
- results validation

5) Anomaly detection algorithms training for unsupervised learning

a) Methodology overview:

- anomaly detection algorithms
- algorithm training process
- results validation

6) Supervised learning based feature selection results analysis

7) Unsupervised learning based feature selection results analysis

8) Results interpretation using LIME technique [8]

Data cleaning phase of data pre-processing stage includes solving the class imbalance problem by removing duplicated records per each existing class, taking roughly the same amount of samples per each of three classes (benign, Mirai, and Gafgyt) with under-sampling, and then randomly undersampling the whole subsample by taking the 30% of the balanced random subsample. Data pre-processing is necessary because original subset is extremely skewed towards anomalous data, especially Mirai class. Class imbalance may lead to the unwanted peformance of ML models – this can appear as trained ML model tendence to be confused when classifying one or more classes and tend to predict accurately only particular class(es).

The randomly formed balanced subsample is then standardized based on IQR (*Interquartile Range*) robust scaling measure for supervised learning part. IQR is a measure of statistical dispersion that is equal to the difference between the upper and lower quartiles [56]. Data standardization step is essential, because some classifiers that are going to be used as training models in feature selection stage are very sensitive to the data pattern, thus fitting the unscaled data may lead to the overfitting or underfitting performance. Moreover, as the dataset chosen for the current research contains normal and anomalous network traffic data, it is significant to select the scaling approach that is robust to anomalous data. According to the study [55], robust scaling with IQR measure is suitable for normalizing the data that contains outliers and extreme values.

Test set for unsupervised learning will be prepared from normalized balanced subset that already contains all three classes. Training set for unsupervised anomaly detection will be prepared using undersampling technique and further normalized with IQR robust scaling from all benign records that were generated from the original dataset.

From the supervised learning perspective, the standardized subsample is later used for creating 2 separate sets: one for training the feature selection models, another one for

final assessment of trained models quality. Splitting the subsample to two separate sets is necessary for evaluating performance of trained ML models on the previously unseen data.

After pre-processing the initial dataset, training set will be used for training the ML models that will be able to select the optimal sets of not more than 10 features amongst 115 attributes. Feature selection is essential for boosting the ML models performance and avoid misleading results. The motivation for reducing set of attributes is based on the following factors:

- preventing the model overfitting – the situation when the parameters learned on a training sample are not reflective and contain noise [12]; high-dimensional data that contains a lot of features leads to the model overfitting, thus effectively reducing amount of features in the training phase reduces the model complexity and leads to more accurate and trustworthy results in the final validation phase [12];

- create more simple model that is easier to interpret – this factor is crucial especially when dealing with features that are linearly correlated to each other; even when the trained model gives good accuracy results, it is crucial to interpret the results by evaluating their trustworthiness [8];

- computational efficiency – training the model on the set with less features takes less time.

First step is applying two filter methods on the normalized sample in parallel for comparison:

1) removing redundant features based on Pearson's linear correlation coefficient value;

2) keeping only 20 attributes amongst 115 ones based on Fisher scoring rank.

Elimination of all features that are linearly correlated according to Pearson's correlation coefficient value and keep only those features that are significant are necessary for

reducing the data sparsity [22]. The linear correlation is a similarity measure between two random features [22]. For example, if one random feature is linearly correlated to another one, it already contains enough information about another one, thus it will be enough to provide the ML algorithm the set without redundant features, because the features existing in the reduced set already include the knowledge about the eliminated ones [22].

Fisher scoring method is designed for selecting the features with the highest discriminatory power. Fisher score is defined as the ratio of the average interclass separation to the average intraclass separation. Larger values indicate higher discriminatory power of numeric attributes [18].

After that, wrapper models will be trained in parallel on:

1) train split that contains all features,

2) train split after applying filter method based on Pearson's coefficient,

3) train split after applying filter method based on Fisher scoring.

Wrapper model goal is to find the subset of the most discriminative features by running classification algorithm in iterative way by forward addition and backward subtraction of features using classification accuracy or another measure as internal cluster validity criterion [18].

Hybrid method is a combination of wrapper and filter methods [18]: first step will be selecting the most suitable features set based on filter methods and then apply wrapper method to the filtered subset.

When the most optimal subsets of features are selected, classifiers are further trained and cross-validated on the separate test split on the selected features only and on all 115 attributes for comparison. Training data contains samples for all of three classes – benign, Mirai, and Gafgyt.

Anomaly detection algorithm used in solving unsupervised binary classification task is

preliminarily fitted on data without anomalous contamination and cross-validated on the sample with instances belonging to three classes. Unsupervised learning based wrapper feature selection will be done on the data that contains benign samples only – this is the main difference compared with supervised approach; final assessment of selected features subset generated by unsupervised wrapper model will be performed using classical ML models on a separate test set by cross-validation.

The final stage is results interpretation. LIME technique [8] will be applied in predictions explanation for evaluating importance values of all 115 features and comparing attributes subsets generated by FSAs with the most important attributes based on LIME ranking. Evaluation of prediction results trustworthiness is essential stage before making a decision whether to deploy particular model in the future or not.

2.2 Tools and technologies

All research stages were done using Python 3.7 programming language due to the wide choice of open-source libraries that were implemented for solving ML tasks. All experiments were done using the PyCharm IDE[1]. The following open-source libraries were used:

- NumPy[2], pandas[3]– data manipulating
- Scikit-learn[4] – data analysis, dimensionality reduction, features selection, cross-validation, classification, accuracy evaluation metrics, exceptions handling
- Scikit-feature[5] – Fisher score calculation

[1]https://www.jetbrains.com/pycharm/

[2]https://www.numpy.org/

[3]https://pandas.pydata.org/

[4]http://scikit-learn.org/

[5]http://featureselection.asu.edu/

- matplotlib[1], Seaborn[2], graphviz[3] – data visualization
- logging[4] – generating log files
- LIME library[5] – predictions trustworthiness interpretation
- Mlxtend[6] – plots drawing.

[1]https://matplotlib.org/

[2]https://seaborn.pydata.org/

[3]https://www.graphviz.org/

[4]https://docs.python.org/3/library/logging.html#module-logging

[5]https://github.com/marcotcr/lime

[6]http://rasbt.github.io/mlxtend/

3 Data pre-processing for supervised classification based and filter FSAs

3.1 Input dataset description

3.1.1 Original data set

Original input data set mentioned in sections 1.2.5 and 1.2.6 is a group of files that contain network traffic data collected from nine IoT devices.

Class	Samples amount	Ratio of records to the total, %
Benign	555 932	~10.6
Mirai	3 668 402	~69.8
Gafgyt	1 032 056	~19.6
Total	**5 256 390**	100

Table 1: Original dataset overview

Sample was extracted from the initial group of sub-datasets. The original dataset was created for separate devices and for additional types of attacks, the data had no labels and was stored in different subfolders to keep benign, Mirai, and Gafgyt attack data separately for each devce, thus numerical values were added as class labels to all the files in sub-datasets (also see the Table 2):

- benign class – 0,
- Mirai class – 1,
- Gafgyt class – 2.

HpHp_L0.01_radius	HpHp_L0.01_covariance	HpHp_L0.01_pcc	Class
268.709047342919	-0.292602111483136	-0.012345266908117	0
0.000000002270098775	0	0	1
0	0	0	2

Table 2: Sample dataset with class labels to represent the dataset structure

3.2 Data cleaning

3.2.1 Class imbalance problem

As it may be seen in the Table 1, Mirai records are prevailing from the first sight. However, after observing the data, it is seen that there are duplicated records for Mirai and Gafgyt classes.

In order to avoid models overfitting, first of all it is necessary to clean data from the duplicated samples that may become a noise for training the ML models.

As classification and anomaly detection algorithms used in this thesis are accuracy driven, it is necessary to resolve class imbalance problem in order to avoid misleading predictions [23] and avoid models overfitting on the training sets. Class imbalance in the set may lead to one or more classes misclassification, thus the model may be confused in predicting one or more classes and tend to classify single class [9].

Class imbalance problem was solved in the steps described below.

Step 1. Original raw dataset was separated to 3 groups – benign, Mirai, and Gafgyt. While creating 3 samples, duplicates were removed from each separate file. After that, the data was stell skewed towards anomalous data.

Step 2. In order to resolve class imbalance problem, it is necessary to follow the *undersampling* technique called RUS (*Random Majority Undersampling*), i.e. proceed the random underrepresentation of particular classes to make the dataset roughly balanced with respect to the minority class [55].

The undersampling should be done for majority classes in regard to the minority class according to the formula (9):

$$Majority\ class\ fraction = 1 - \frac{majority\ class\ samples - minority\ class\ samples}{majority\ class\ samples} \quad (9)$$

At this step, majority classes are Mirai and Gafgyt, so the fractions were calculated in respect with benign records amount (see Figure 3).

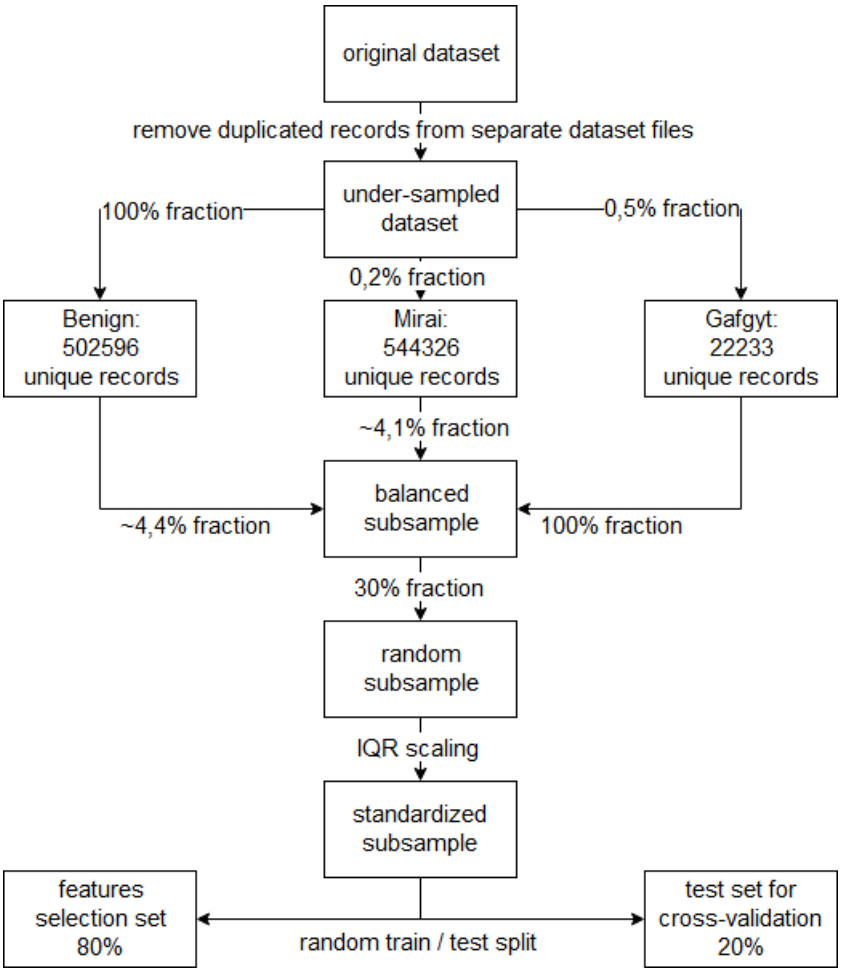

Figure 3: Data pre-processing strategy: cleaning, undersampling, and train / test splitting

Class	Unique samples amount	Fraction to keep, %
Benign	502596	~4.4
Mirai	544326	~4.1
Gafgyt	22233	100

Table 3: Dataset overview after removing duplicated records

Step 3. After applying RUS technique in the step 2, Mirai, benign, and Gafgyt subsamples were checked again for the duplicated records. After removing duplicates, the cleaned dataset is now skewed towards benign and Mirai data, while Gafgyt class is insufficiently represented (see Table 3). Majority classes (benign and Mirai) are now undersampled again following the same equation that was used for RUS technique in the previous stage.

3.2.2 Sample preparation

Sample used for standardization, training, and final assessment of ML models quality was done by randomly extracting 30% of the balanced dataset described in the section 3.2.1.

Balanced subsample will be used as an input in all the following stages:

- standardization,
- the most relevant features subsets selection using wrapper, filter, hybrid, and ensemle methods;
- validation of trained classification models on the separate test split;
- validation of features subset generated by wrapper model based on unsupervised anomaly detection.

3.2.3 Data standardization

Before applying feature selection techniques, it is important to take into account the data pattern and classifiers sensitivity towards extreme values of the data.

As the dataset chosen for current research is a network traffic data that contains both normal and anomalous behavior patterns, it is crucial to select such a normalization approach that is robust to the outliers.

Balanced subsample is scaled with IQR, that is also called the midspread or middle 50% - a measure of statistical dispersion that is calculated according to the equation (10):

$$IQR = Q_3 - Q_1 \quad , \tag{10}$$

where Q_1 – first quartile, Q_3 – third quartile [56].

IQR is applied as a **robust measure of scale** – statistics measure that is optimal scaling approach for the data that has distribution differing from normal and at the same time has outliers[1].

The *RobustScaler*[1] algorithm of the Scikit-learn library with default parameters was applied to the balanced subsample.

As it may be noticed in the Figure 4, classes distribution for two features that are the most important according to Fisher score values (H_L0.01_weight and MI_dir_L0.01_weight, will be discussed in section 5) in the balanced subsample is extremely skewed towards Mirai class, whereas benign class seems to be missing for the pair of this features.

After applying IQR scaling, the data pattern looks different: Mirai class is now less skewed, while the benign class records values are performing here as outliers and have left unscaled.

For comparison, the *StandardScaler* algorithm of the Scikit-learn library was applied to the same pair of features in the same balanced subsample to compare classes distribu-

[1]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

tion after normalization. As it is seen, classes distribution is still skewed towards Mirai, but this time the benign class distribution is ignored as an outlier. The *StandardScaler*[1] scales the attributes according to the equation (11):

$$z = \frac{x - u}{s} \quad , \tag{11}$$

where u is the mean value of data samples, and s is the standard deviation of samples.

As the feature selection leads to the data reduction, it is crucial to get the scaled data with keeping the outliers unscaled. For this reason, the robust scaling approach has been chosen.

[1]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

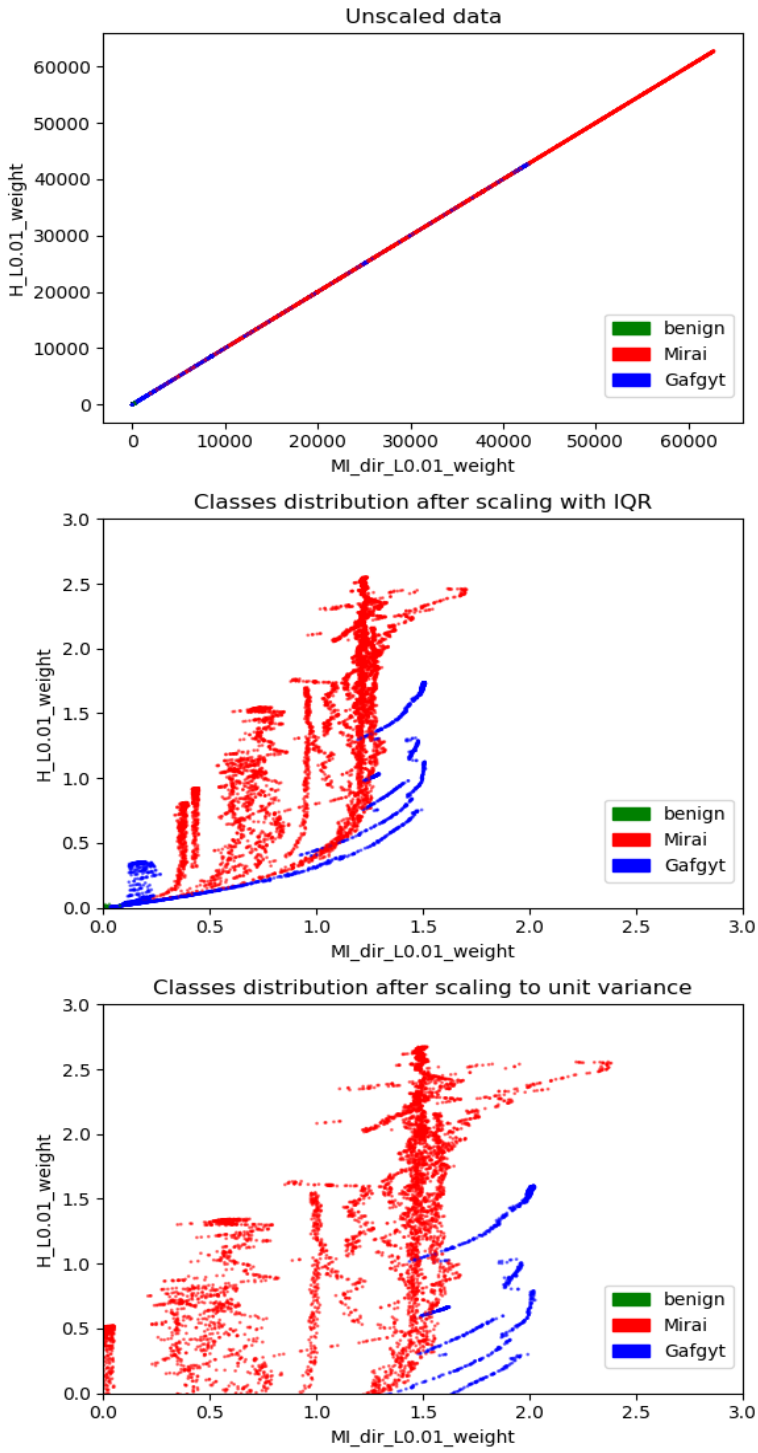

Figure 4: Classes distribution before and after applying standardization

4 Data pre-processing for unsupervised anomaly detection

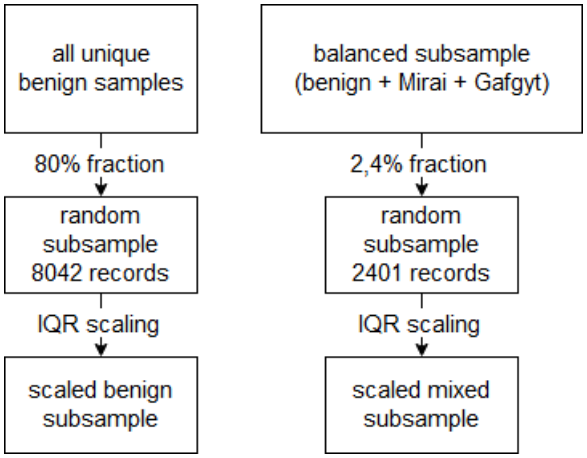

Figure 5: Samples preparation flow for unsupervised anomaly detection

When solving unsupervised anomaly detection as binary classification problem, anomaly detection algorithm attempts to classify sample as **normal** or **benign** after preliminarily being fitted on the data containing normal data only. Anomaly algorithms will be fitted on the data containing no contamination (benign records only) that was generated using undersampling approach from all benign samples existing in original dataset. Benign sample is extracted as a random 80% - fraction from all unique benign samples, and further standardized using IQR robust scaling.

After fitting benign data to the anomaly detection algorithms, it is necessary to fit test sample that contains both normal and anomalous data. Test data was extracted from the balanced dataset already mentioned in the section 3.2.2 that contains all three classes. After applying random undersampling and taking 2,4% fraction from the balanced dataset, IQR robust scaling was applied. Class labels for records belonging to benign group (inliers) were changed to label ’1’, while class labels for Mirai and Gafgyt families were renamed to ’-1’ (outliers). Benign and test samples ratios are roughly 80% and 20% accordingly.

5 Feature selection

5.1 Filter model

As proposed by John and Kohavi in 1997, there are two basic feature selection approaches – wrapper (described further in section 5.2) and filter method. Filter method is variable ranking based on specific criteria [36].

Threshold method was applied in the scope of this thesis for both filters – Fisher score and Pearson coefficient rankers. Applying threshold will be done in the following ways (see the Figure 6):

1) form the subset of features with dropping out attributes that are irrelevant (filter based on Pearson's coefficient) and later use this subset for hybrid and ensemble selection;

2) form the subset with fixed size of 20 attributes based on Fisher score rank and use this subset in wrapper method forming the hybrid; running wrapper models on the reduced datasets is for taking advantage of lowering the greedy wrapper selection algorithms complexity;

3) form the subset of 10 best features from Fisher score ranking and use as input in ensemle feature selection – find the intersection between this subset and hybrid models output subsets;

4) form the subset based on Fisher score ranking with keeping only those features that have score value greater or equal than 1, evaluate performance and compare with other filter, hybrid, and ensemble models outputs at the final validation stage on the disjoint test set.

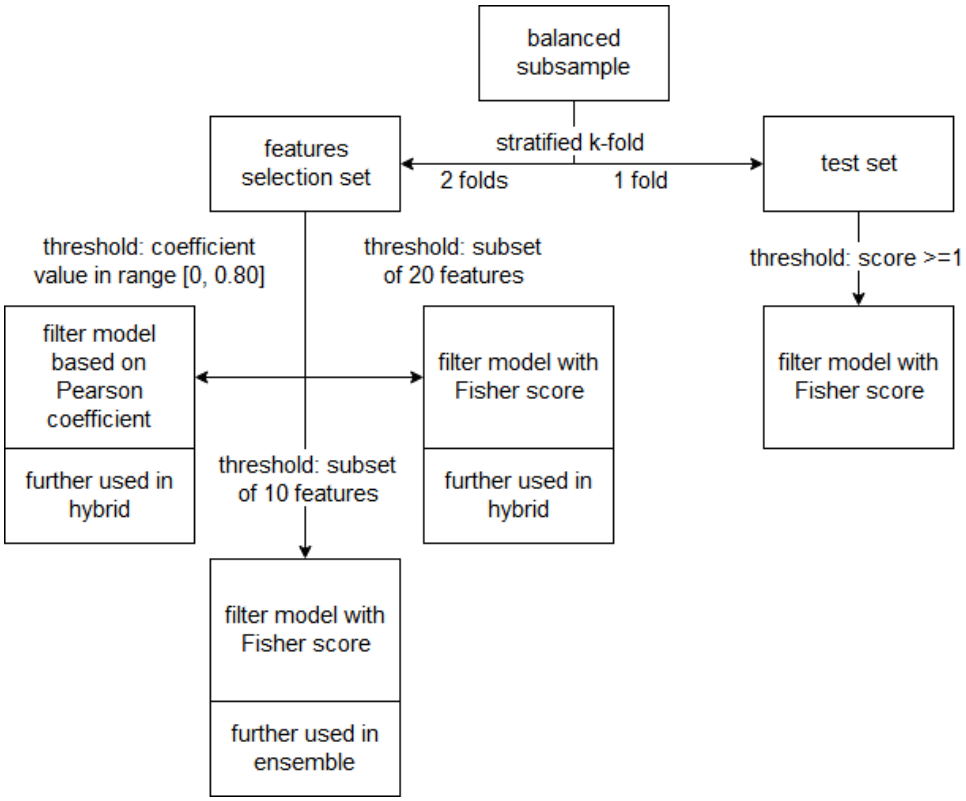

Figure 6: Filter models workflow

5.1.1 Removing features based on Pearson's coefficient

One of the essential phases in data pre-processing is removing features that are related to each other. Current approach proposes the filter method based on *Pearson's linear correlation coefficien*t.

The Pearson's linear correlation coefficient measures the strength of linear association between features [50]. The bigger linear correlation value is, the more similar values of adjacent features are [18]. In other words, linear correlation coefficient value expresses the features relationship strength.

The Pearson's linear correlation coefficient is calculated according to the equation (12):

$$\rho = \frac{E[X \cdot Y] - E[X] \cdot E[Y]}{\sigma(X) \cdot \sigma(Y)} \quad , \tag{12}$$

where:

X, Y – randomly selected variables (features),

$E[X]$ - the expectation of X,

$\sigma(X)$ - the standard deviation of X [18].

If *supp*(*i*) and *supp*(*j*) are the relative supports of individual items, *supp*({*i,j*}) – the relative support of the itemset {*i,j*}, then the overall correlation for the whole dataset is calculated according to the formula (13) [18]:

$$\rho_{i,j} = \frac{supp(\{i,j\}) - supp(i) \cdot supp(j)}{\sqrt{supp(i) \cdot supp(j) \cdot (1 - supp(i)) \cdot (1 - supp(j))}} \tag{13}$$

Pearson's linear correlation coefficient value always lies in range [-1,1], so there are the following boundary values with interpretations:

- if coefficient value is **-1** (strongly negative) between two variables, it indicates that variables linear relationship is perfectly negative, i.e. features are absolutely not related to each other;
- if coefficient value is **0**, it indicates that there is no linear relationship between variables X and Y, however, it does not yet mean that features X and Y are *independent* [34]
- if coefficient value is **1** (strongly positive), it indicates that linear relationship between such variables is perfectly positive; as an example, Pearson's coefficient value for a feature itself (i.e. between variables X and X, or Y and Y) is 1.

Linear correlation of 115 features of the dataset is calculated based on Pearson coefficient of correlation between a pair of features. Optimal features subset will contain only those attributes that have linear correlation value in range [0, 0.8] – such features will have no strongly positive nor strongly negative relationship.

5.1.2 Selecting features based on Fisher score ranking

Fisher score is a measure of the features discriminatory power – the higher the score is, the greated discriminatory power of particular feature is.

Fisher score is defined as the ratio of the interclass separation to intraclass separation [18] and calculated according to the formula (14):

$$F = \frac{\sum_{j=1}^{k} \rho_j (\mu_j - \mu)^2}{\sum_{j=1}^{k} \rho_j \sigma_j^2} \quad , \tag{14}$$

where:

- ρ_j - fraction of data points belonging to class j,
- μ_j – the mean deviation of data points belonging to class j,
- σ_j - the standard deviation of data points belonging to class j.

Alternatively, Fisher score can be derived from the Laplacian score according to the equation (15):

$$F_r = \frac{1}{L} - 1 \quad , \tag{15}$$

where F_r is the Fisher score of the r-th feature.

Laplacian score is based on Laplacian Eigenmaps and Locality Preserving Projection. Calculation algorithm is based on the assumption that the dataset can be represented as a weighted graph with edges connected to the nearby points. Laplacian score evaluates features according to their locality preserving power [31] and is calculated according to the formula (16):

$$L_r = \frac{\sum_{ij} (f_{ri} - f_{rj})^2 S_{ij}}{Var(f_r)} \quad , \tag{16}$$

where:

- f_{ri} - the i-th sample of the r-th feature,
- S_{ij} – similarity between the i-th and j-th nodes in weighted graph,
- $Var(f_r)$ – estimated variance of the r-th feature.

Fisher score can be used as a filter method for keeping attributes with the higher values.

5.2 Wrapper models

5.2.1 Wrapper model approach

Wrapper model is a feature subset selection model that „*uses the performance of the learning algorithm as heuristics*" [14] when comparing set of models.

Learning algorithms used for the wrapper models in this thesis are classification algorithms. Chosen algorithms and their parameters will be discussed in chapter 6 of this thesis.

Current solution proposes cross-validating learning algorithms based on **macro-averaged F1 score** heuristic. This approach can also be called cross-validation based as proposed in [51], but the difference in current solution is that heuristic chosen for evaluating a candidate subset of attributes is the average of all macro-averaged F1 scores calculated on $k - 1$ iterations instead of cross-validation accuracy. Cross-validation average accuracy will be calculated in the final assessment stage when all optimal feature subsets will be evaluated on a separate disjoint test set by the same classification algorithms that were used in wrapper models for lowering the risk of overfitting ML models on reduced and previously unseen data.

Sequential forward and backward feature selection algorithms implemented for this thesis are extending open-source project authored by S. Shinde[1] (current implementation of sequential forward and sequential backward feature selection is adapted from the code implemented by S. Shinde, see examples of adapted methods code in Figure 26, Figure 27, and Figure 28).

5.2.2 Sequential forward feature selection

Forward feature selection is greedy search algorithm that starts with empty features subset, adds new feature to the optimal subset at each of the iterations in case the candidate feature leads to the maximum accuracy or another heuristic [51].

Current solution is sequential forward feature selection with maximum allowed number of elements in optimal features subset is 5, each candidate subset is cross-validated based on the macro-averaged F1 score (see Figure 7 **A)**).

Implemented greedy sequential forward selection algorithm:

1) create empty set of features, best macro-averaged F1 score = 0;
2) split the input dataset based on stratified k-fold to 3 folds: 2 folds are used for training, third one for calculating the average of macro-averaged F1 scores got on 2 splits;
3) add the most promising feature from the input dataset until the threshold of 5 features is reached, recalculate F1 score, best F1 score = temporary F1 score;
4) go to 3.

5.2.3 Sequential backward feature selection

Sequential backward feature selection (sequential backward feature elimination) is a feature selection algorithm that is similar to the forward feature selection with the

[1]https://github.com/sachin1092/Feature-Selection

difference that in the first iteration candidate subset contains all the features and then at each iteration the less promising features are eliminated [27].

Current implementation proposes the sequential backward elimination with maximum allowed number of features 5 (the same threshold as in sequential forward selection), estimation criteria is the same as in sequential forward feature selection – average of macro-averaged F1 cross-validation scores (see Figure 7 **B)**).

Implemented greedy sequential backward feature elimination algorithm:

1) create set of all input dataset attributes, best macro-averaged F1 score = 0;
2) split the input dataset based on stratified k-fold to 3 folds: 2 folds are used for training, third one for calculating the average of macro-averaged F1 scores got on 2 splits;
3) eliminate less promising feature from the input dataset based on recalculated F1 score until the threshold of 5 features is reached, best F1 score = temporary F1 score;
4) go to 3.

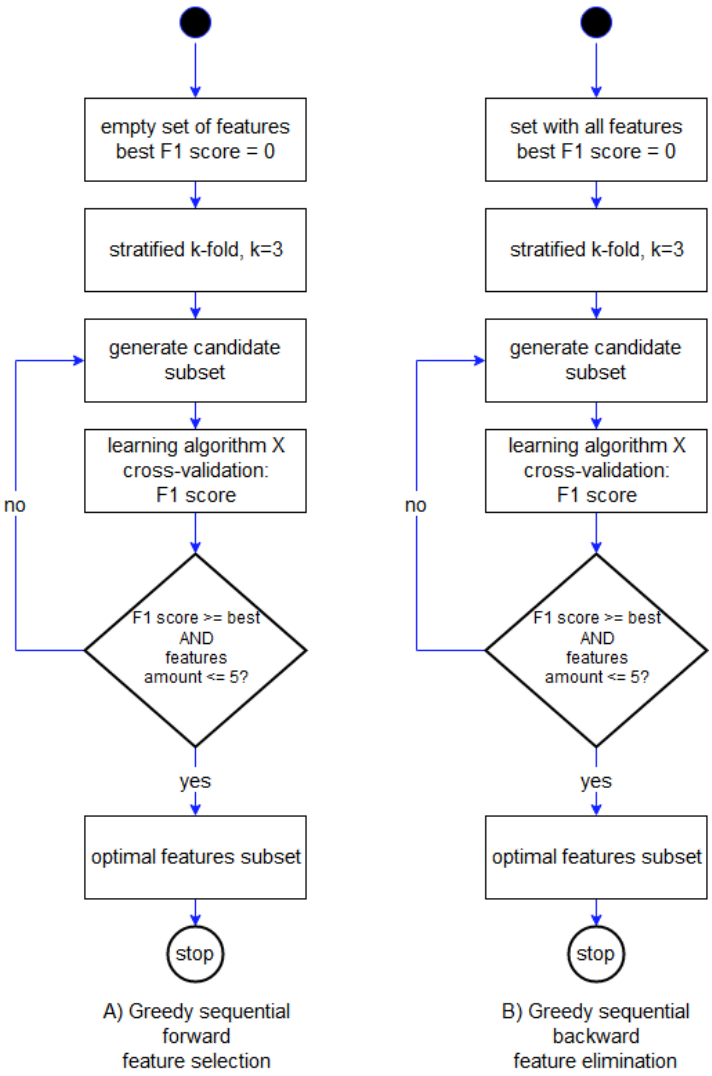

Figure 7: Wrapper feature selection algorithms

5.3 Hybrid models

Hybrid model is combination of wrapper and filter models for gaining better ML model performance [18]. Applying classification algorithms combined with filter models based on Fisher score ranking and filtering based on Pearson's coefficient values range, hybrid model consists of two phases:

1. selecting candidate subset according to filter model,
2. evaluating candidate features subset with learning algorithm using the average macro-averaged F1 score as heuristic.

Proposed solution for applying hybrid method in this thesis is combination of wrapper methods (greedy forward and backward features subsets selection) and the reduced subsets based on filter methods (see the Figure 8).

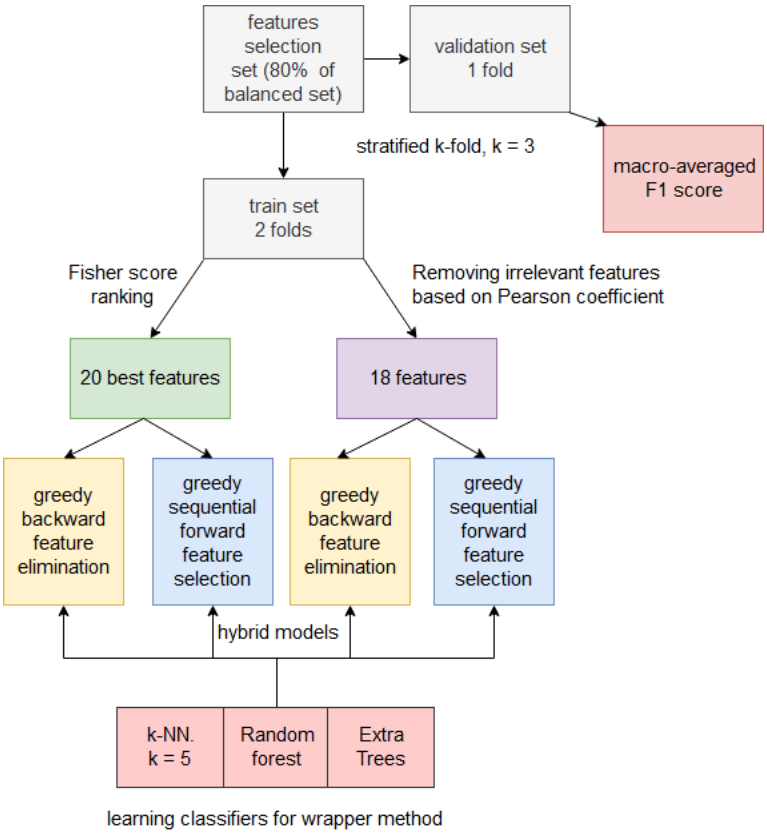

Figure 8: Classifiers training procedure for hybrid feature selection

5.4 Ensemble models

Ensemble feature selection model is combining the outputs (optimal subsets of attributes) found by several feature selection models. Ensemble approach developed in the scope of current research includes finding the intersections of optimal feature subsets that are previously found by filter, wrapper, and hybrid models.

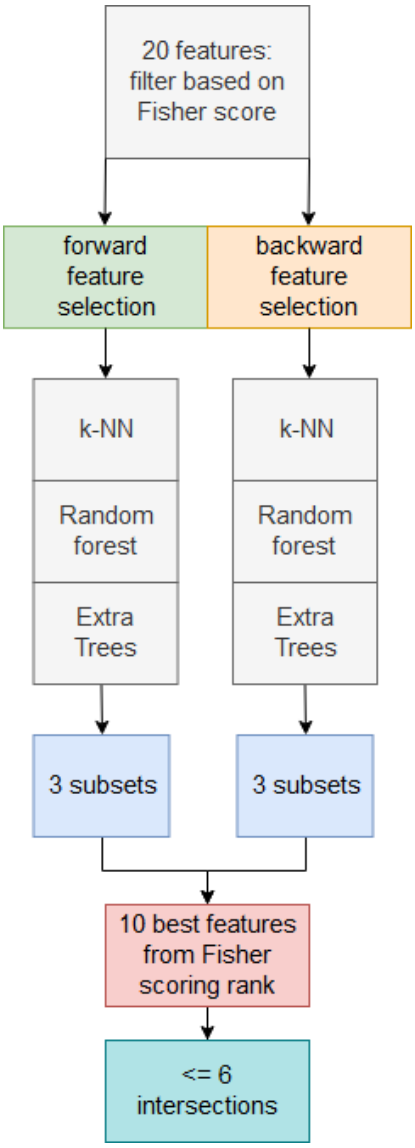

Figure 9: Ensemble models workflow

The whole workflow of getting feature subsets intersections is shown in the Figure 9.

Procedure of finding features subsets using ensemble model consists of the following steps:

1) reduce the same features selection set to 20 features based on Fisher score ranking;

2) run greedy forward and backward feature selection with threshold of 5 features on set using cross-validation based approach using each of 3 classifiers (combination of filter and wrapper approach – 2 hybrid models);

3) outputs that were produced by 2 hybrid models trained in the previous step are 3 subsets x 2 hybrid models = 6 optimal subsets;

4) extract 10 best features based on Fisher score rank from the same features selection sample;

5) for each subset described in step 3 find the elements that are in common with features found in the subset generated at step 4; as a result, there will be 6 or less intersections.

6 Classifiers training for supervised learning

6.1 Methodology

After the data pre-processing phase, running feature selection models with classifiers as learning algorithms for solving supervised classification problem will be done in parallel on one balanced subsample in three ways:

a) whole balanced subsample that contains all 115 features,

b) reduced balanced subsample that contains only 18 features that have remained according to the Pearson's linear coefficient threshold,

c) reduced balanced subsample that contains only 20 best features based on Fisher scoring rank.

Balanced subsample contains records for all three classes, thus the problem to be solved as a part of feature selection models is supervised *multi-class classification*.

Classifiers are going to be trained on the 80% split extracted from the balanced dataset. Each model selection method has its own specific training procedure.

6.1.1 Classifier algorithms

Three-class supervised classification performance of the following classical ML algorithms is going to be boosted by applying several feature selection methods..

1. *Random forest* – tree-based ensemble method that constructs each decision tree using a separate bootstrap sample, grows unpruned tree, randomly samples subset of the predictors, choses the best split, and makes prediction about new data based on the aggregation of the estimators predictions [39].

RandomForestClassifier of the Scikit-learn library was used for current implementation with 50 estimators, maximum depth of value 5 for lowering the risk of overfitting.

2. *Extremely randomized trees* – tree-based ensemble method that randomizes „attribute and cut-point while splitting tree node" [25]; main advantages of this algorithm are accuracy and computational efficiency. The Extra-Trees algorithm builds ensemble of unpruned decision trees and, compared with other tree-based algorithms, uses the whole learning sample instead of the bootstrap for building the tree in order to minimize the bias. *ExtraTreesClassifier* of the Scikit-learn library was used for current implementation with 50 estimators, maximum depth of value 5 for lowering the risk of model overfit.

3. *K-NN (k nearest neighbors)* – non-parametric classification method based on the majority voting of the retrieved *k* nearest neighbors of the data among the data records in neighborhood [26]. *KNeighborsClassifier* of the Scikit-learn library was used for current implementation with default parameters (k = 5).

6.1.2 Validation of the classifiers

Classifiers will be validated at two different stages (see the Figure 10):

a) hybrid models learning algorithms training stage: each of four classifiers described in section 6.1.1 will be trained and tested iteratively using stratified k-fold approach (k = 3), 2 splits will be used for training, last one for validating the candidate features subset based on **macro-averaged F1-score**;

b) final assessment stage: each optimal feature subset that was generated by different will be evaluated with 3 classifiers described in section 6.1.1 using stratified k-fold **cross-validation accuracy** (k = 3) on the separate test set for evaluating models quality on previously unseen data using the same classifiers that were used in the training stage; 2 splits will be used as training set, and the last one for validation; for example, if subset X was generated by hybrid model using random forest classifier as learning algorithm that was *wrapped* around the

fitered dataset, subset X will be tested on the disjoint set using the same ML algorithm – random forest (see the Figure 11);

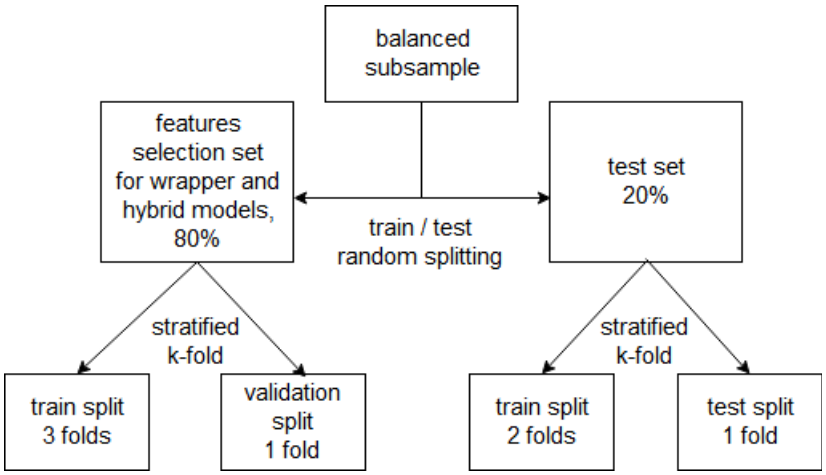

Figure 10: Train, validation, and test samples extraction overview

Cross-validation is ML models performance technique where the labeled dataset divided to k separate equal-sized parts, ML models are iteratively trained on k - 1 disjoint subsets and tested on the separate k-th subset. The procedure is done in k – 1 iterations. Finally, **cross-validation accuracy** is calculated as the average of all k - 1 accuracies [18]. The main advantage of applying cross-validation is lowering the risk of ML models overfitting on the test set.

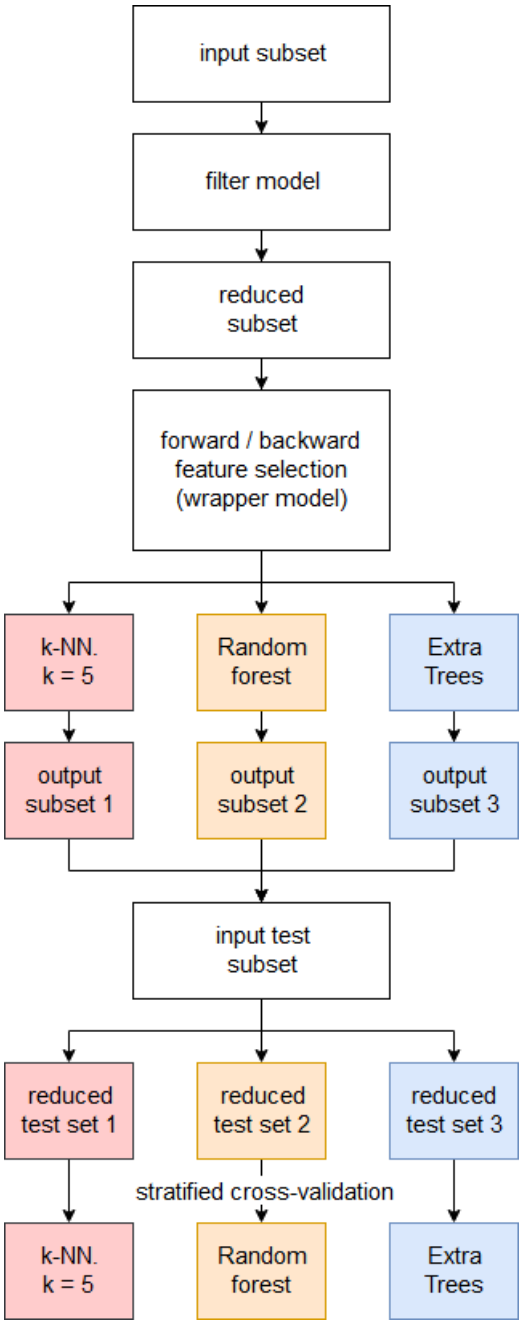

Figure 11: Hybrid models cross-validation flow on the test set

6.1.3 Classifiers training process

At features selection stage, all three classifiers will be trained in parallel for hybrid models in two ways (also see the Figure 8 from previous chapter):

1) on the train folds containing best 20 features according to Fisher score ranking,

2) on the train folds containing 18 features after removing linearly uncorrelated features based on Pearson's coefficient value.

At optimal attributes subsets final validation stage, the same classifiers will be trained on train folds substracted from the test subsample in parallel following five approaches (see the Figure 12):

1) the whole dataset containing 115 features,

2) 4 best features based on Fisher score ranking (Fisher score value ≥ 1, filter model),

3) 12 optimal features subsets that were chosen by hybrid models shown in the Figure 8,

4) 18 features remained after dropping irrelevant features based on Pearson's coefficient value (filter model),

5) hybrid models (see the Figure 8) outputs (optimal features subsets) intersection with the best 10 features according to Fisher scoring rank – ensemble model, ≤ 6 subsets.

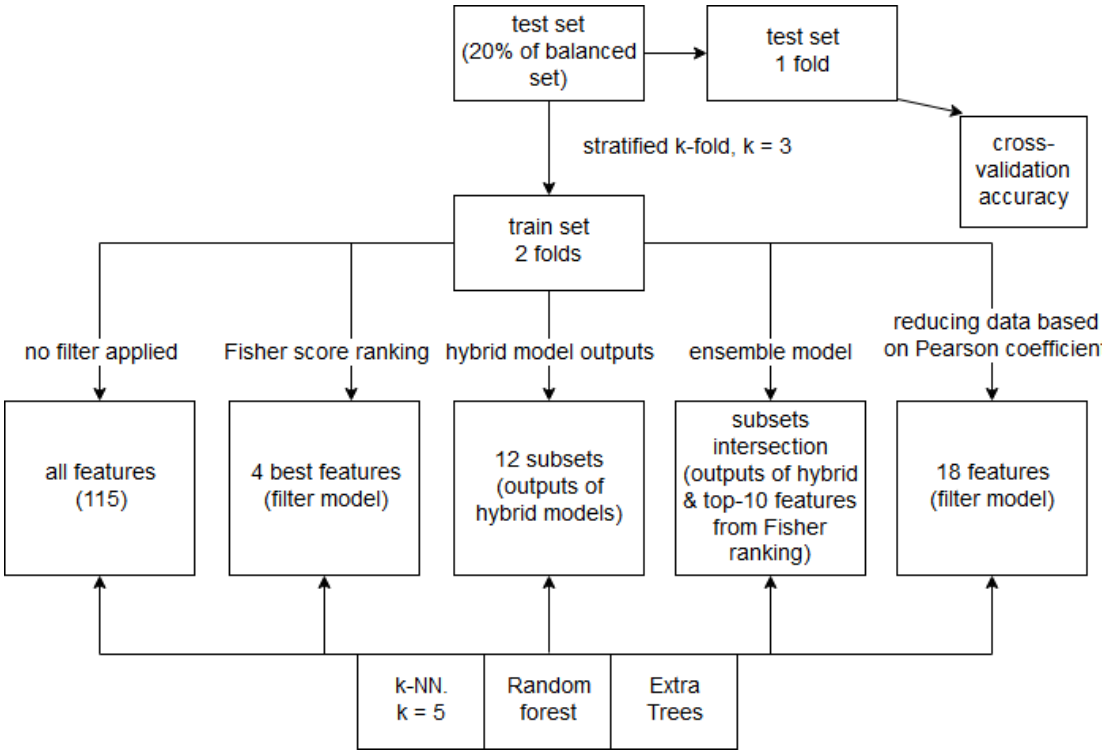

Figure 12: Classifiers cross-validation with and without applying feature selection techniques

7 Anomaly detection algorithms training for unsupervised learning

7.1 Methodology

Feature selection wrapper algorithms are going to be applied for solving unsupervised anomaly detection as binary classification task with two classes – normal (benign) and anomalous (Mirai and Gafgyt). Goal of this phase is to demonsrate that it is possible to select optimal features subsets from the data that contains normal traffic pattern and not contaminated by anomalous instances. Optimal subsets will be compared with sets of attributes that were generated during supervised learning when the training data in contrast contained samples belonging to all three classes – benign,, Mirai, and Gafgyt.

7.1.1 Anomaly detection algorithms

For comparison purposes, wrapper feature selection models based on greedy forward feature selection are going to be applied to unsupervised anomaly detection without preliminarily reducing normal and mixed test sets with any filter method. LOF (*Local Outlier Factor*) algorithm will be used as training algorithm for wrapper model.

LOF is an outlier detection algorithm that is based on the concept of a local outlier capturing the degree of a certain object being an outlier based on the density of its local neighborhood [5]. LOF is a value that can be used for evaluating any object's likelihood: higher values indicate that observable object is outlier, while lower values indicate that object is normal. The LOF value is based on the number of nearest neighbors used assigning the local neighborhood value to particular object [5], [10]. Implementation utilizes *LocalOutlierFactor* class of the Scikit-learn library with parameters n*ovelty=True, contamination='auto*'.

7.1.2 Anomaly detection algorithms training at wrapper feature selection stage

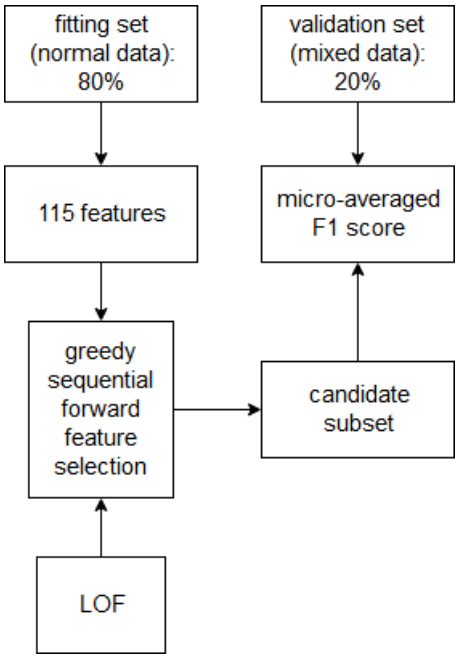

Figure 13: Training process for unsupervised anomaly detection algorithms

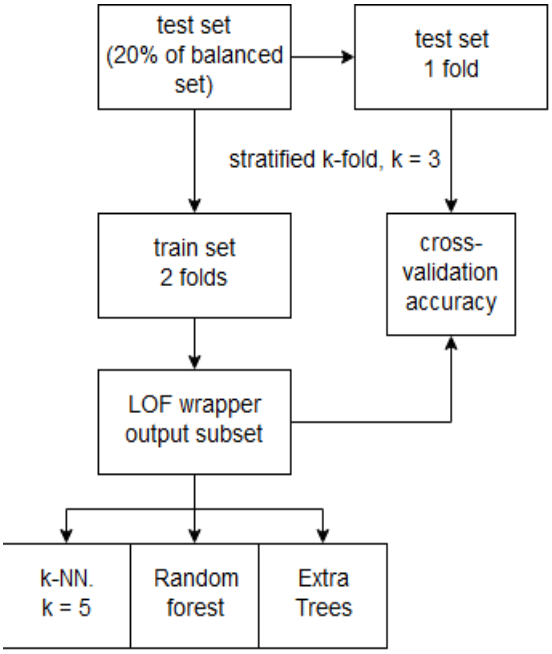

Figure 14: Validation process for optimal subsets generated by unsupervised wrapper models

Wrapper feature selection algorithm (greedy forward feature selection with attributes subset upper threshold - 5 features) based on LOF anomaly detection algorithm as learning estimator will be preliminarily fit on unlabeled normal data containing benign instances; best features are selected from all candidate subsets based on micro-averaged F1-score calculated for the predictions made on labeled validation set containing benign, Mirai, and Gafgyt class instances (see the Figure 13), i.e. LOF as wrapper learning estimator trained in unsupervised manned on benign data and cross-validated on mixed labeled data. Wrapper method output is a subset of most optimal attributes based on wrapper sequential forward selection with LOF.

7.1.3 Validation of unsupervised feature selection wrapper models

Optimal subset generated by unsupervised wrapper model described in the section 7.1.2 is cross-validated by classifiers that were used in supervised hybrid feature selection models (see the Figure 14). The test set used at this stage for stratified cross-validation is the same that was previously used for validating outputs generated by models based

on supervised feature selection and filter models (see the section 6.1.2) and contains instances belonging to all three classes.

8 Feature selection results based on filter models

8.1 Filter method based on Pearson's linear correlation coefficient

After constructing linear correlation matrix for all features pairwise in the balanced input data set and filtering out the features with Pearson's coefficient value in range [0, 0.80], only 18 attributes were selected (see the Table 4). The aim of selecting features with coefficients values in the predefined range is to drop out all irrelevant attributes – the features with very high relationship, and the features with strongly negative relationship.

Nr.	Feature name	Time-frame	Feature group
1	MI_dir_L5_weight	L5	**Source MAC-IP** – stats summarizing the recent traffic from this packet's host (IP + MAC) [58].
2	MI_dir_L5_mean		
3	MI_dir_L5_variance		
4	HH_L5_weight	L5	**Channel** - stats summarizing the recent traffic going from this packet's host (IP) to the packet's destination host [58].
5	HH_L5_std		
6	HH_L5_radius		
7	HH_L5_covariance		
8	HH_L5_pcc		
9	HH_L0.1_covariance	L0.1	
10	HH_L0.1_pcc		
11	HH_L0.01_covariance	L0.01	
12	HH_L0.01_pcc		
13	HH_jit_L5_mean	L5	**Channel jitter** - stats summarizing the jitter of the traffic going from this packet's host (IP) to the packet's destination host [58].
14	HH_jit_L5_variance		
15	HH_jit_L1_variance	L1	
16	HpHp_L5_weight	L5	**Socket** - stats summarizing the recent traffic going from this packet's host+port (IP) to the packet's destination host+port [58].
17	HpHp_L5_radius		
18	HpHp_L5_covariance		

Table 4: Features with linear correlation in range [0, 0.80]

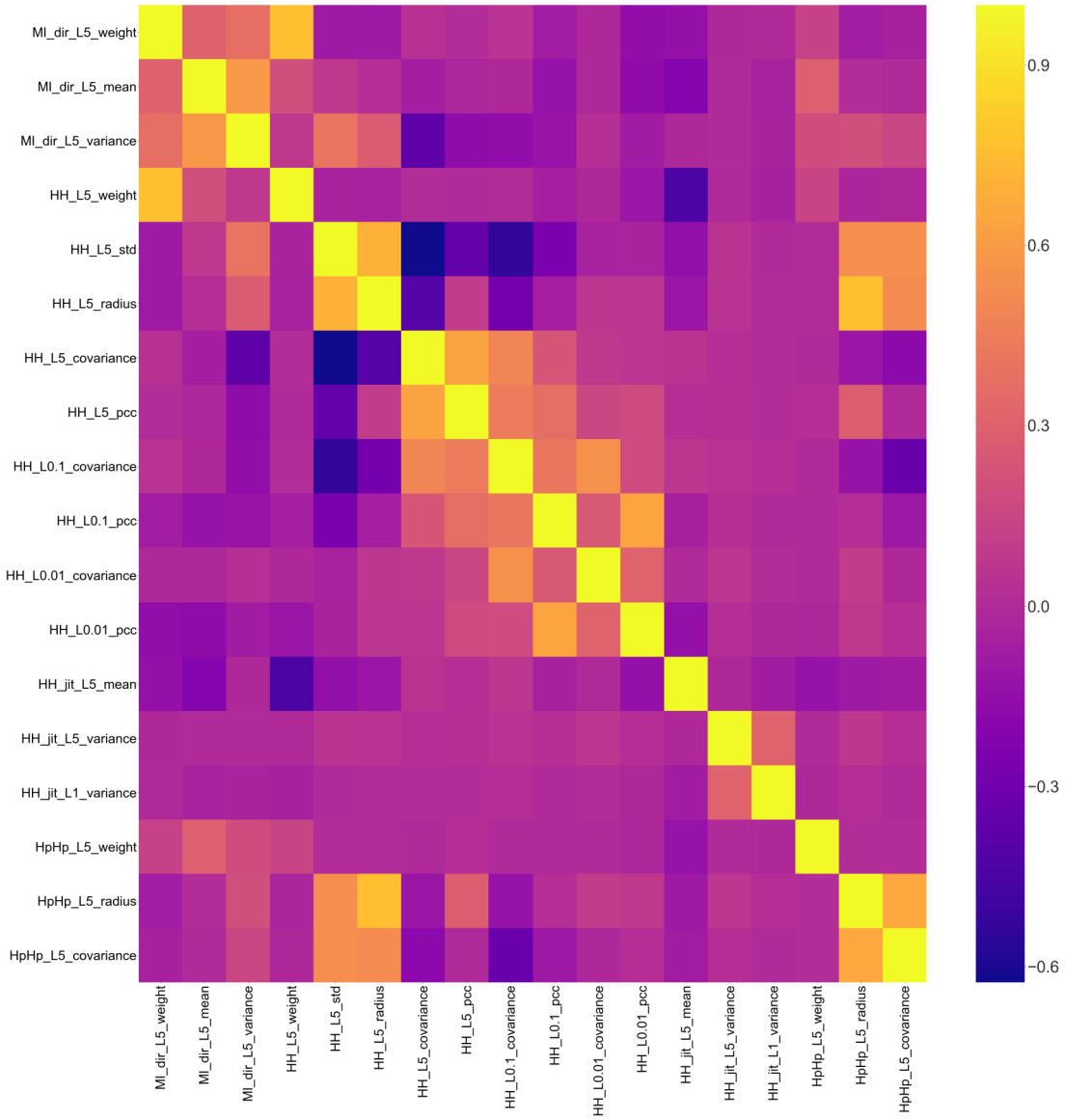

Figure 15: Pearson's linear correlation heatmap for features that have remained after applying model with coefficient values threshold [0, 0.80]

The Figure 15 illustrates the linear correlation between 18 features that have remained in the balanced dataset after applying filter model based on Pearson's linear correlation coefficient values range. Features with correlation coefficient values in range [0, 0.80] have left, all remaining features were dropped out. As it might be seen, there is a strong positive linear correlation between the features and themselves (for example, MI_dir_L5_weight and MI_dir_L5_weight) marked in yellow color, while dark blue color indicates weak linear correlation between features (see HH_L5_covariance in horizontal axis and HH_L5_std).

It is seen in the heatmap that the vast majority of features have weak linear correlation or have no linear relationship at all (except with themselves), it indicates that the set mostly contains the features that are not highly correlated to each other nor have perfectly negative linear relationship with value -1. Pairs of attributes with linear correlation coefficient value above 0.50 can be found in the Table 5:

Feature 1	**Feature 2**	**Pearson's correlation coefficient value**
HH_L5_radius	HpHp_L5_radius	0.7627915044103211
MI_dir_L5_weight	HH_L5_weight	0.7590947141618555
HH_L5_std	HH_L5_radius	0.7048698823607285
HpHp_L5_radius	HpHp_L5_covariance	0.6620891391573257
HH_L0.1_pcc	HH_L0.01_pcc	0.6442478620130772
HH_L5_covariance	HH_L5_pcc	0.6340942024422493
MI_dir_L5_mean	MI_dir_L5_variance	0.5839561591801016
HH_L0.1_covariance	HH_L0.01_covariance	0.5515356706185149
HH_L5_std	HpHp_L5_covariance	0.5352915587282823
HH_L5_std	HpHp_L5_radius	0.5308023795986179
HH_L5_radius	HpHp_L5_covariance	0.5234339063603991

Table 5: Features with linear correlation above 0.50 from the filtered set

Feature 1	**Feature 2**	**Pearson's correlation coefficient value**
HH_L0.1_covariance	HpHp_L5_weight	0.0007215915421614352
HH_L0.1_pcc	HH_jit_L1_variance	0.0009630960392771497

Table 6: Features with linear correlation in range [0, 0.001)

Attributes pairs that have no evident linear correlation can be found in the Table 6 - those features Pearson's correlation coefficient values are very close to 0 and less than 0.001.

8.2 Filter method based on Fisher score

Only 20 features were selected with higher Fisher score values among 150 ones:

Feature nr. in the rank	**Feature name**	**Fisher score**
1	**MI_dir_L0.01_weight**	**1.535444**
2	**H_L0.01_weight**	**1.535444**
3	**MI_dir_L0.1_weight**	**1.231247**
4	**H_L0.1_weight**	**1.231247**
5	MI_dir_L1_weight	0.904455
6	H_L1_weight	0.904455
7	MI_dir_L5_weight	0.840989
8	H_L5_weight	0.840989
9	MI_dir_L3_weight	0.835061
10	H_L3_weight	0.835061
11	MI_dir_L0.01_mean	0.689287
12	H_L0.01_mean	0.689254
13	MI_dir_L0.1_mean	0.645926
14	H_L0.1_mean	0.645915
15	MI_dir_L1_variance	0.619762
16	H_L1_variance	0.619762
17	MI_dir_L1_mean	0.577434
18	H_L1_mean	0.577434
19	H_L3_mean	0.528538
20	MI_dir_L3_mean	0.528538

Table 7: Twenty features with the highest Fisher score in descending order

It can be noticed from the Table 7 that four variables with the highest Fisher score values are related to each other, those are:

- MI_dir_L0.01_weight and MI_dir_L0.1_weight – stats summarizing the recent traffic from the packet's host (source MAC-IP);

- H_L0.01_weight and H_L0.1_weight – stats summarizing the recent traffic from the packet's host (source IP).

Interestingly, selected features are token from the same time-frames – L0.01 and L0.1.

As previously mentioned in the section 3.2.3, robust IQR scaling affected the pattern of classes distribution for the pair of features having the highest Fisher score values (MI_dir_L0.01_weight and H_L0.01_weight - see the Figure 4). It comes to evidence that keeping outlier values in the features that were selected as the most important ones may have crucial impact on the training process of the data, especially for lowering the risk of models overfitting and tendency to misclassify one of the classes (in particular case – benign). The classes distribution for two features selected based on Fisher score ranking is strongly skewed towards attack data.

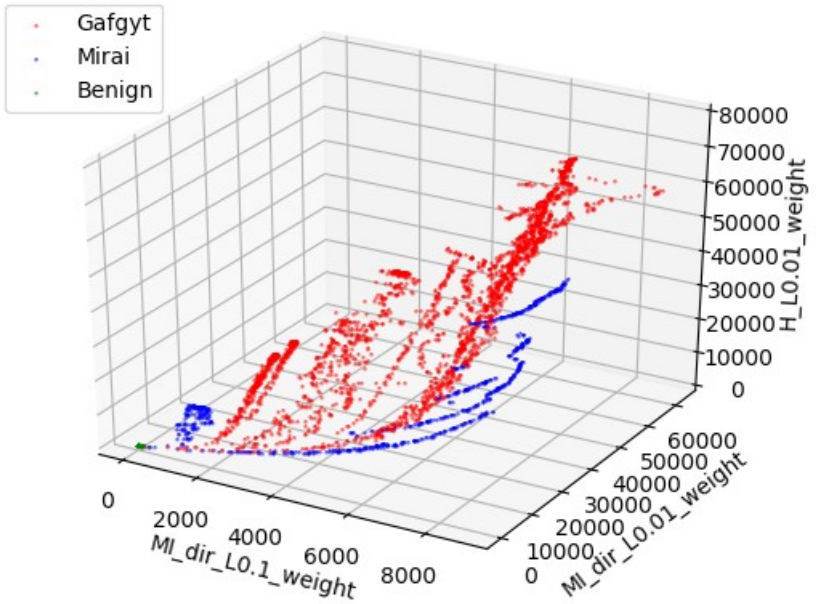

Figure 16: Classes distribution plot for 3 best features from Fisher scoring rank - MI_dir_L0.1_weight, MI_dir_L0.01_weight, and H_L0.01_weight

As it seen in the Figure 16, classes distribution for three best features from Fisher scoring rank is skewed towards attack data. In contrast, if we would like to explore three features that have not been selected by any of filter methods (based both on Pearson's coefficient values range and Fisher scoring rank), it comes to evidence that classes distribution is now skewed towards normal data (see the Figure 17), this results will be discussed later in chapter 10.

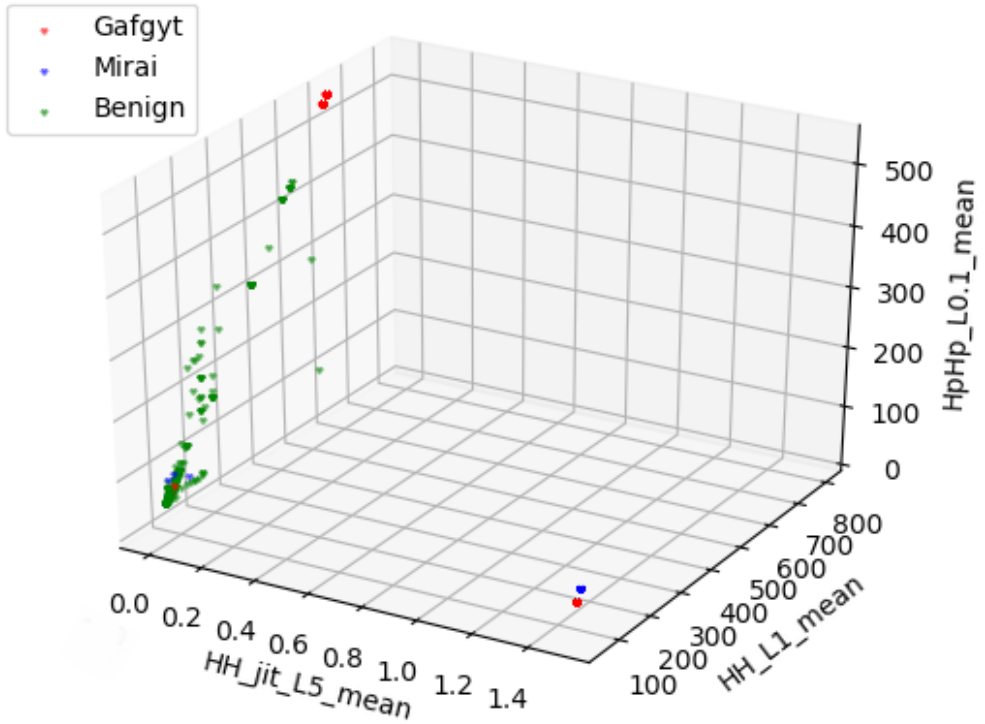

Figure 17: Classes distribution plot for features that have not been selected by filter methods

Comparison of classifiers performance can be found in the Table 8. The same test sample was used for stratified k-fold cross-validation with four classifiers. Interestingly, running data set reduced to the 10 best features based on Fisher ranking and to 20 features based on Pearson range with random forest classifier outperforms other classifiers applied to filter models.

Classifier		**115 features**	**4 best features (Fisher rank)**	**10 best features (Fisher rank)**	**20 features (Pearson)**
K-NN, k = 5	**Accuracy**	**0.9434**	**0.98601**	**0.9874**	**0.9134**
	FP	143	28	24	170
	FN	121	31	27	205
Random forest	**Accuracy**	**0.9984**	**0.9782**	**0.9840**	**0.9878**
	FP	2	76	33	18
	FN	0	25	40	16
Extra trees	**Accuracy**	**0.9978**	**0.9674**	**0.7945**	**0.9676**
	FP	0	117	56	3
	FN	0	22	23	97

Table 8: Comparison of cross-validation scores for algorithms performing on the whole data and the data reduced after applying filters

9 Feature selection results based on supervised learning

9.1 Feature selection with hybrid models

The Table 9 contains comparison of predictions accuracy values for hybrid models that were evaluated on a separate test set using stratified k-fold cross-validation (k = 3). All subsets with length not more than 5 attributes perform well and lead to accuracy values in range 0.97 .. 0.99.

Wrapper model	**Filter model**	**Cross-validation accuracy**		
		K-NN, k = 5	**Random forest**	**Extra Trees**
Sequential forward feature selection	20 features (Fisher ranking)	0.9964	0.9990 (see Figure 18)	0.9974
	18 features (Pearson ranking)	0.9878	0.9954	0.9770
Sequential backward elimination	20 features (Fisher ranking)	0.9970	0.9990 (see Figure 19)	0.9980
	18 features (Pearson ranking)	0.9870	0.9970	0.9792

Table 9: Comparison of predictions accuracies made by hybrid models

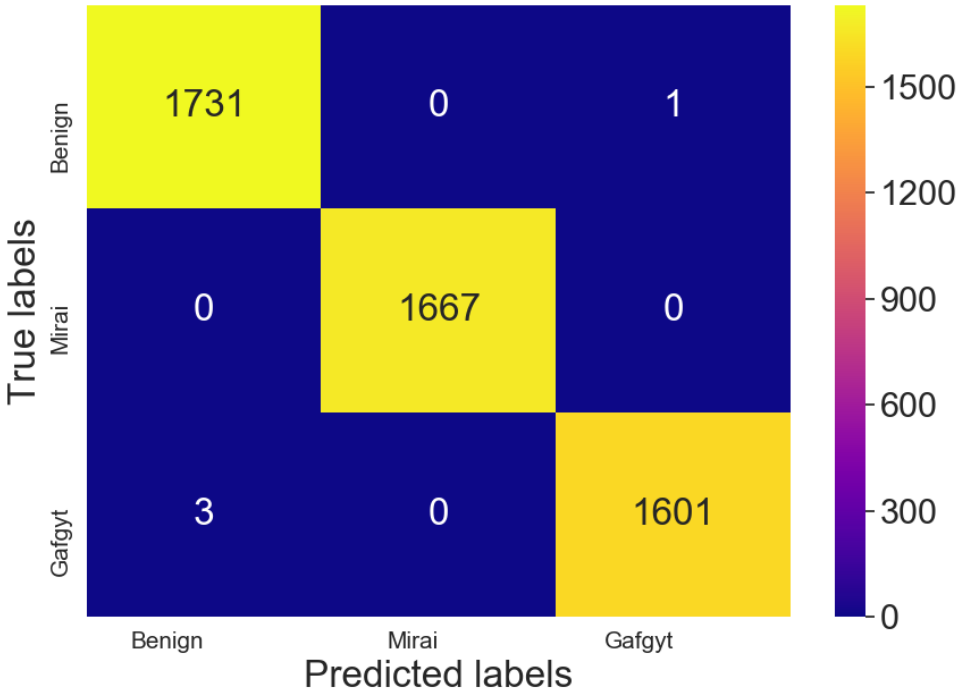

Figure 18: Confusion matrix for subsets generated by hybrid model based on combination of sequential forward feature selection and Fisher score ranking threshold, random forest classifier accuracy 0.9990

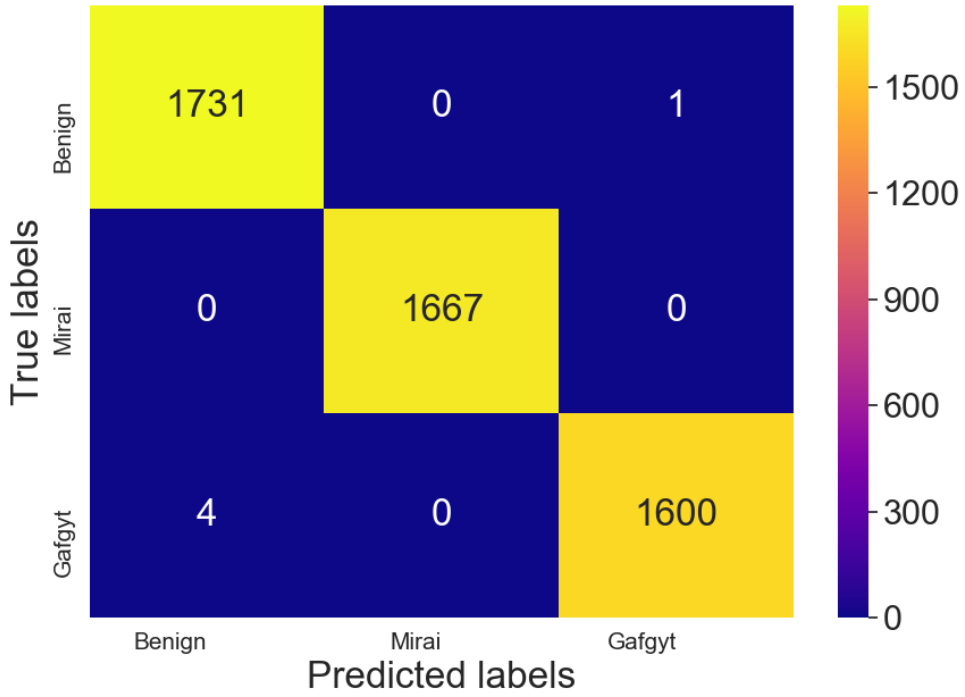

Figure 19: Confusion matrix for subsets generated by hybrid model based on combination of sequential backward feature selection and Fisher score ranking threshold, random forest classifier accuracy 0.9990

9.2 Features selection with ensemble models

Features subsets intersections found for hybrid models ant 10 best features from Fisher score ranking lead to optimal subsets that perform well on test sample using stratified k-fold cross-validation (k = 3). Curiously enough, all these subsets contain the same feature – H_L0.1_weight.

Hybrid model	**Classifier used in wrapper model & in validation**	**Intersection with 10 best features from Fisher score ranking**	**Cross-validation accuracy on the test set**
Forward selection & 20 best features from Fisher score ranking	K-NN, k = 5	No intersection found	-
	Random forest	MI_dir_L3_weight, **H_L0.1_weight**	0.9626
	ExtraTrees	MI_dir_L0.01_weight, H_L3_weight, **H_L0.1_weight**	0.7663
Backward selection & 20 best features from Fisher score ranking	K-NN, k = 5	No intersection found	-
	Random forest	H_L3_weight, **H_L0.1_weight**	0.9616
	ExtraTrees	MI_dir_L1_weight, **H_L0.1_weight**	0.7811

Table 10: Comparison of predictions accuracies made by ensemble models

Results proof that it is possible to select features optimal subset with length even less than 5 that lead to high accuracy values, the following subsets with length 2 lead to accuracy about 0.96:

1) MI_dir_L3_weight, H_L0.1_weight (see ROC curve in the Figure 20 and decision boundary in the Figure 22);

2) H_L3_weight, H_L0.1_weight (see ROC curve in the Figure 21 and decision boundary in the Figure 23).

Both these subsets of attributes perform well with AUC values in range 0.95 – 1.00 for all classes indicating almost perfect results for distinguishing three different classes. ROC curves were calculated using class *OneVsRestClassifier* of Scikit-learn library.

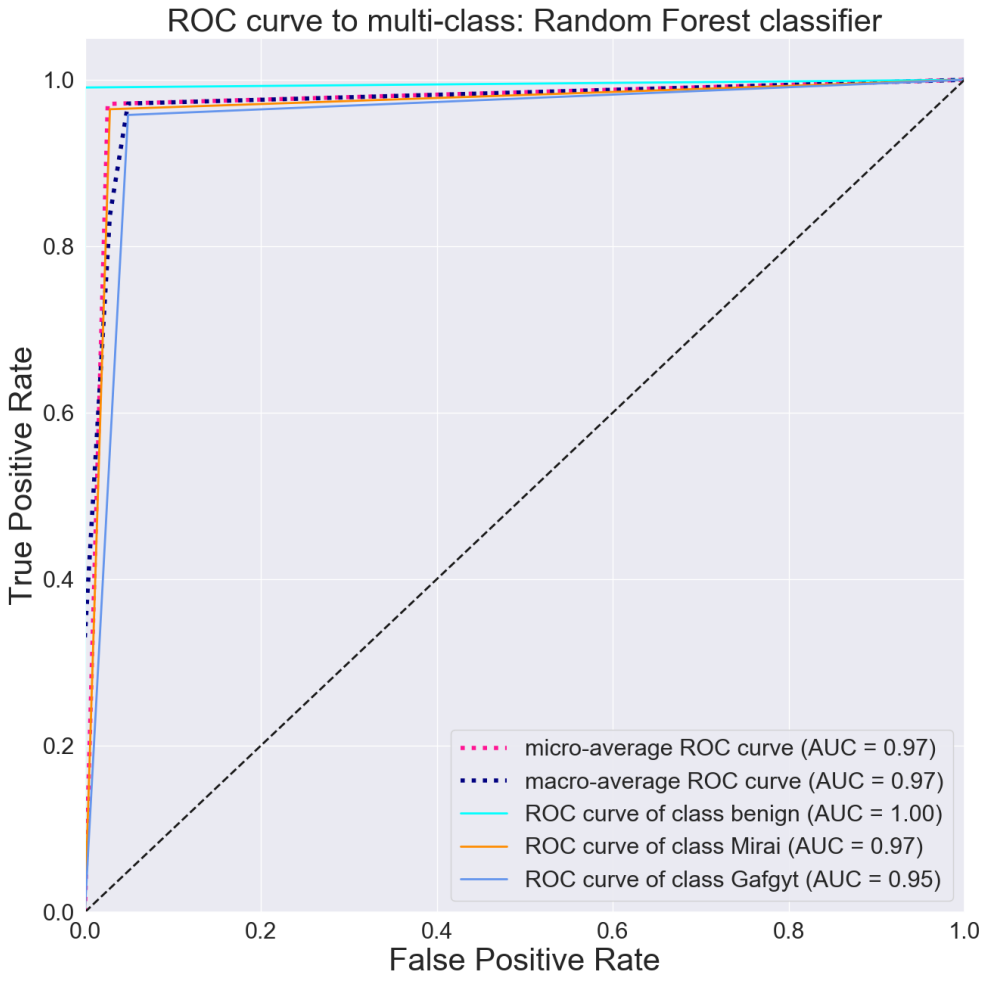

Figure 20: ROC curve for classification on MI_dir_L3_weight and H_L0.1_weight features

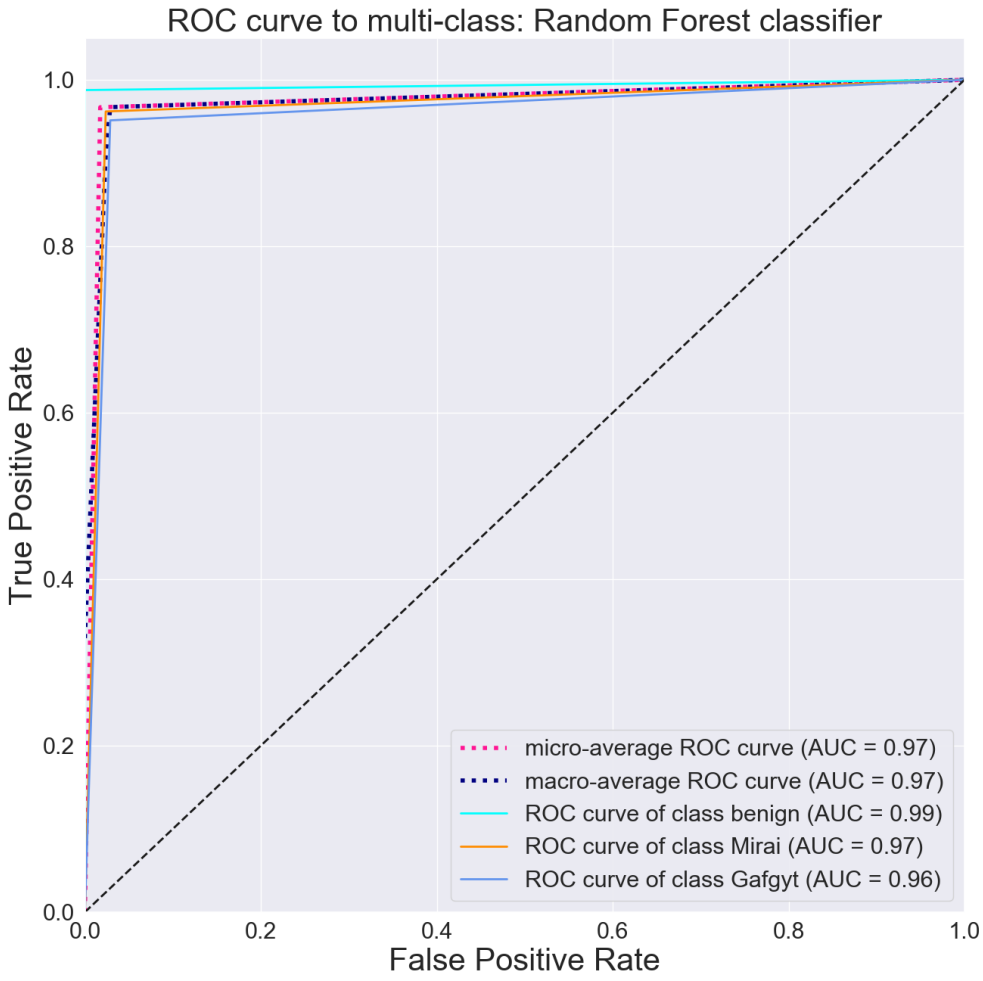

Figure 21: ROC curve for classification on H_L3_weight and H_L0.1_weight features

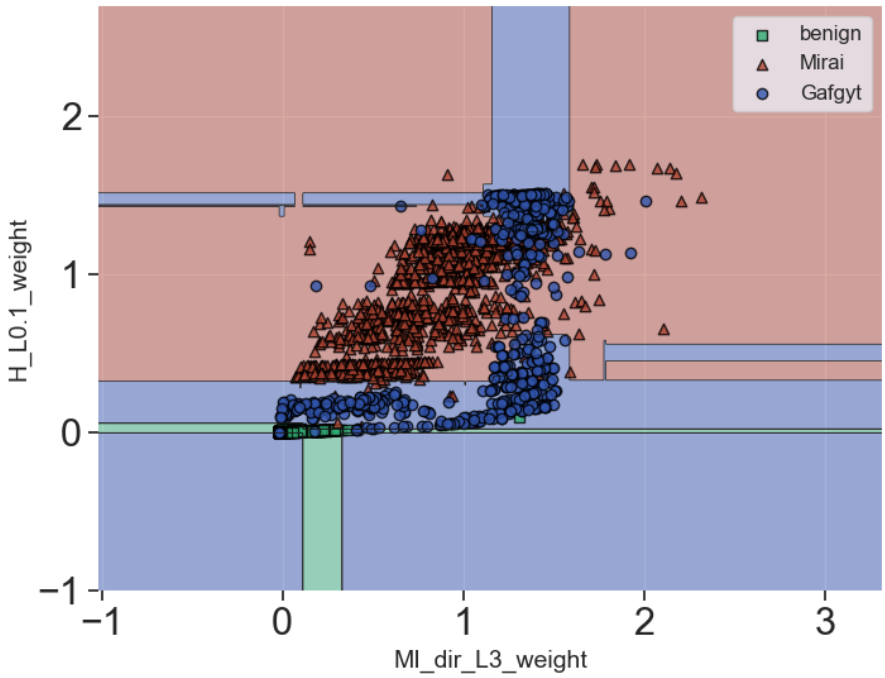

Figure 22: Decision boundary for random forest on the subset generated by ensemble model

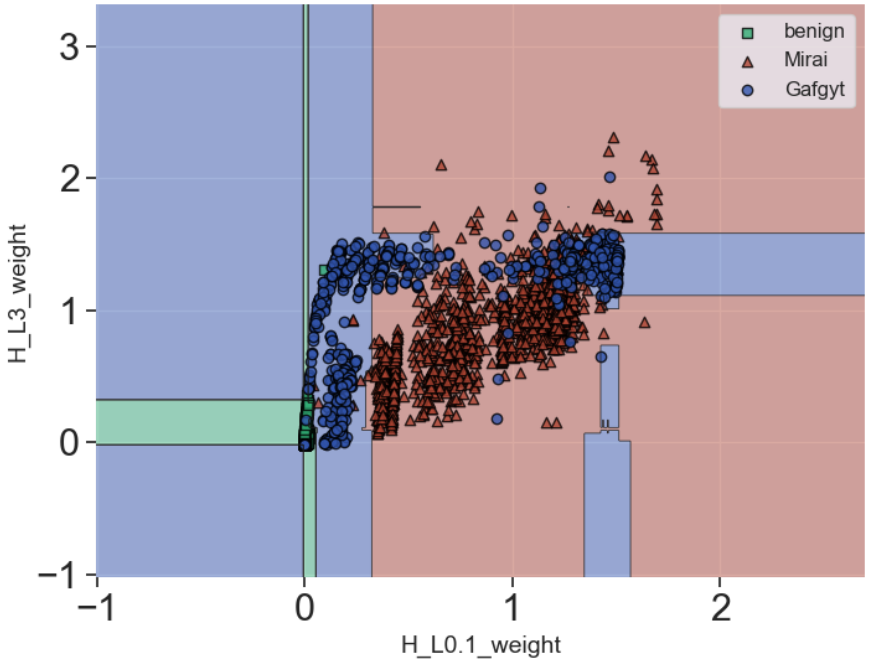

Figure 23: Decision boundary for random forest on the subset generated by ensemble model

10 Feature selection results based on unsupervised learning

Comparison of unsupervised wrapper model output subset performance on the separate test subset with 3 classifiers is shown in the Table 11. Stratified cross-validation of generated subset with length 5 has proven that it is possible to select optimal feature subset from the training data that contains benign instances only. Optimal subset outperforms on test set with k-NN (k = 5) and random forest classifiers. Surprisingly, classes distribution for features belonging to this subset is now skewed towards benign data (see the Figure 17 In chapter 8.2). What is more interesting, this subset does not overlap with subsets generated earlier by filter, supervised learning based hybrid, and ensemble model outputs.

Wrapper model	**Features subset**	**Classifier used in wrapper model**	**Cross-validation accuracy on the test set**
Greedy forward selection + LOF	HH_L1_mean, HH_L0.1_std, HH_L0.1_pcc, HH_jit_L5_mean, HpHp_L0.1_mean	K-NN, k = 5	0.9608 (see the Figure 24)
		Random forest	0.9442 (see the Figure 25)
		Extra Trees	0.8369

Table 11: Comparison of predictions for optimal subset generated by unsupervised wrapper model

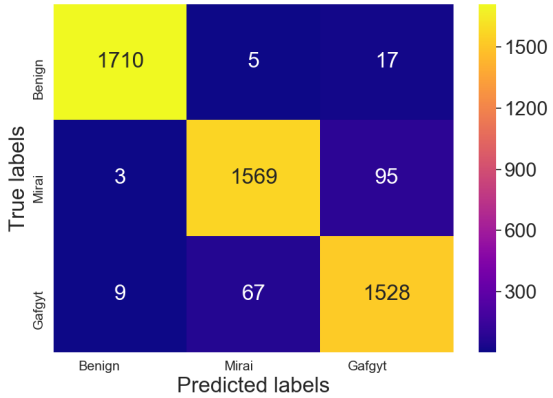

Figure 24: Confusion matrix for k-NN classifier on unsupervised wrapper model output subset

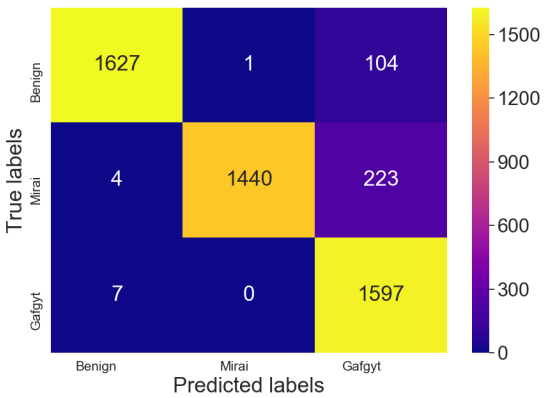

Figure 25: Confusion matrix for random forest on unsupervised wrapper model output subset

11 Predictions interpretation with LIME

11.1 Methodology

Final stage of the research represents interpretation of classification results. The motivation is to check whether optimal feature subsets contain those attributes that have the stronger impact on predicting whether a single data point can be mapped to a particular class or not. Attribute importance weights calculation strategy is specific for each ML model and strongly affected by the data bias, thus it is significant to check what features are having stronger impact in making predictions treating classifier as a black box. Predictions interpretation stage is crucial before making a decision on choosing the most appropriate and trustworthy model for the future deployment.

11.2 LIME technique

LIME is predictions interpretation technique that can be applied to any model in *agnostic* manner, i.e. treating it as a black box. Features importance values are calculated based on explanation matrix that is constructed for the interpretable components of each instance. Attributes with higher representativeness value have stronger impact on predicting all instances, i.e. LIME algorithm picks features that cover most important components and avoids selecting features with analogous explanations [8].

11.3 LIME interpretation results

Random instance was taken from the same set that was earlier used for filter models and for supervised classification based feature selection models. The set was splitted into train (80%) and test (20%) splits using random sub-sampling. *LimeTabularExplainer* class of LIME library was used here in solving 3-class classification task for all 115

features using random forest classifier as estimator. In the Table 12, Table 13, and Table 14 are 10 most important features from significance rank that have the most strong impact compared with remaining features on predicting whether a random sample belongs to particular class or not. According to LIME interpretation, classifier is 100% certain that an instance belongs to Mirai class (prediction probability value 1.00).

Number in importance rank	**Feature**	**Weight value**	**Importance value**	**Mirai?**
1	H_L0.01_weight	1.54	0.07	Yes
2	MI_dir_L0.01_weight	1.54	0.06	Yes
3	H_L0.1_weight	0.72	0.05	Yes
4	MI_dir_L0.1_weight	0.72	0.04	Yes
5	HH_jit_L1_mean	1.00	0.03	Yes
6	MI_dir_L5_weight	0.76	0.03	Yes
7	H_L1_weight	0.66	0.02	Yes
8	HH_jit_L0.1_mean	1.00	0.02	Yes
9	HpHp_L3_covariance	0.00	0.02	Yes
10	MI_dir_L1_weight	0.66	0.02	Yes

Table 12: LIME explanation for predicting random instance belonging to Mirai class with random forest classifier

Number in importance rank	**Feature**	**Weight value**	**Importance value**	**Benign?**
1	H_L0.01_weight	1.54	0.03	Yes
2	MI_dir_L0.01_weight	1.54	0.02	Yes
3	H_L0.01_variance	1.80	0.02	No
4	HH_jit_L0.1_mean	1.00	0.02	Yes
5	H_L0.1_mean	1.18	0.02	No
6	MI_dir_L0.01_mean	1.52	0.02	No
7	MI_dir_L0.1_variance	2.01	0.02	No
8	HH_L5_magnitude	-0.42	0.02	No
9	H_L0.1_variance	2.01	0.01	No
10	HH_jit_L5_mean	1.00	0.01	Yes

Table 13: LIME explanation for predicting random instance belonging to benign class with random forest classifier

Number in importance rank	Feature	Weight value	Importance value	Gafgyt?
1	H_L0.01_weight	1.54	0.10	No
2	MI_dir_L0.01_weight	1.54	0.08	No
3	H_L0.1_weight	0.72	0.05	No
4	MI_dir_L0.1_weight	0.72	0.05	No
5	HH_jit_L1_mean	1.00	0.04	No
6	HpHp_L3_covariance	0.00	0.03	No
7	HH_jit_L0.01_mean	1.00	0.03	No
8	HH_jit_L0.1_mean	1.00	0.03	No
9	H_L0.1_mean	1.18	0.03	Yes
10	HH_jit_L5_mean	1.00	0.02	Yes

Table 14: LIME explanation for predicting random instance belonging to Gafgyt class with random forest classifier

LIME interpretation results for predicting a random instance belonging to Mirai class with probability value 1.00 have shown that there are several features that were also found by filter models, such as 20 best features from Fisher score ranking. What is more interesting, first four features that have the strongest influence on correctly classifying random instance as Mirai are the same as 4 best features from Fisher score ranking:

1) H_L0.01_weight,
2) MI_dir_L0.01_weight,
3) MI_dir_L0.1_weight,
4) H_L0.1_weight.

This subset has been earlier cross-validated on a separate test set and gave accuracy 0.9782 for the random forest estimator with low false positive and false negative rates (see the Table 8 in section 8.2).

12 Discussion and future work

To conclude, primary goals of this thesis have been achieved – optimal attribute subsets that were found by several FSAs have shown good performance on validation set that is comparable with deep learning approach accuracy values for the same dataset. Classical ML algorithms performances boosting deduce consectary that it is possible to select optimal sets of attributes with consuming less computational resources.

Filter, ensemble, supervised learning based hybrid, and unsupervised learning based wrapper models were constructed for selecting most optimal subsets with number of attributes not more than 10 elements. Optimal subsets generated by mentioned models provided prediction performance with best accuracy values in range 0.94 – 0.99.

Surprisingly, feature selection designed for unsupervised anomaly detection also boosted classifiers performances - wrapper model based on LOF estimator and greedy forward feature selection has generated output subset that has demonstrated cross-validation accuracy 0.96 with k-NN classifier. Generated subset differs from those sets of attributes that were chosen based on data set containing all three classes instances, so it may be helpful to explore the impact of hybrid models on classical ML models performance when selecting most promising features from normal traffic data.

Cross-validation of all FSAs have shown that random forest and k-NN classifiers performances were boosted by data reduction, whereas Extra Trees classifier performance was reduced by certain methods – this also needs more investigation.

Final interpretation of results based on LIME algorithm applied to random forest 3-class classification have demonstrated that 10 features with stronger impact mostly coincide with features selected by filter model based on Fisher score, especially first 4 most significant features that coincide with 4 best features from Fisher scoring rank. This research outcome leads to the conclusion that current filter model implementation is trustworthy to some degree and may be used in further series of research related to anomaly detection.

13 Summary

Primary goal of this thesis was to find optimal subsets of not more than 10 attributes from the data set containing 115 features describing network traffic data and to demonstrate that data reduction based on several feature selection algorithms boost performance of classical ML models, thus it is possible in future to deploy less complex ML models with consuming less computational resources on algorithms training.

Research is based on network traffic data with benign instances and anomalous instances belonging to two families of botnet attacks – Mirai and Gafgyt.

Novel combination of feature selection methods was applied for selecting optimal sets of attributes for boosting performances of classical ML models. Proposed solution is filter model, hybrid model (combination of filter and wrapper models), ensemble model based on several output subsets intersection.

Novelty of this research is provided by applying wrapper feature selection model on unsupervised anomaly detection, where LOF algorithm is preliminarily fit on normal data that contains no anomalous contamination.

Cross-validation accuracy values for outperforming feature selection models are in range 0.94 – 0.99. Optimal subsets with minimal number of features contain 2 attributes and provide cross-validation accuracy 0.96 with random forest classifier.

Interpretation of 3-class classification predictions based on LIME algorithm have demonstrated that optimal subsets generated by models that contain filter approach based on Fisher score ranking have some degree of trust.

Achieved results indicate that main tasks have been successfully completed, those are:

- select optimal subsets containing not more than 10 attributes;

- boost performances of classical ML algorithms by achieving trustworthy and interpretable results (high AUC area, low FN and FP rates, prediction accuracy with value higher than running on all 115 features or at least 0.90 in case of high prediction accuracy with running on the whole data set);
- compare several FSAs, such as filter, wrapper, hybrid, ensemble models;
- apply wrapper model to unsupervised classification and compare achieved results with FSAs that are based on supervised learning (hybrid, ensemble);
- interpret classifier predictions using LIME algorithm and compare earlier generated attributes subsets (wrapper, filter, hybrid, ensemble models outputs) with those that were selected based on model agnostic approach.

Present research can be dedicated to applying feature selection methods with training on pure benign data and to developing other combinations of feature selection models with other estimators.

References

[1] Globerson, Amir & Tishby, Naftali. (2003). Sufficient Dimensionality Reduction.. Journal of Machine Learning Research. 3. 1307-1331. 10.1162/153244303322753689.

[2] Douligeris, Christos & Mitrokotsa, Aikaterini. (2004). DDoS attacks and defense mechanisms: Classification and state-of-the-art. Computer Networks. 44. 643-666. 10.1016/j.comnet.2003.10.003.

[3] Mirkovic, Jelena & Reiher, Peter. (2004). A taxonomy of DDoS attack and DDoS Defense mechanisms. ACM SIGCOMM Computer Communication Review. 34. 10.1145/997150.997156.

[4] Bilge, Leyla & Balzarotti, Davide & Robertson, William & Kirda, Engin & Kruegel, Christopher. (2012). Disclosure: Detecting botnet command and control servers through large-scale NetFlow analysis. ACM International Conference Proceeding Series. 129-138. 10.1145/2420950.2420969.

[5] M. Breunig, Markus & Kriegel, Hans-Peter & Ng, Raymond & Sander, Joerg. (2000). LOF: Identifying Density-Based Local Outliers.. ACM Sigmod Record. 29. 93-104. 10.1145/342009.335388.

[6] S. Gibson "DRDoS: Description and analysis of a potent, increasingly prevalent, and worrisome internet attack," Gibson Research Corporation (2002). [Online]. Available: https://goo.gl/zH26gj

[7] Paxson, Vern. (2001). An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. Computer Communication Review. 31. 10.1145/505659.505664.

[8] Tulio Ribeiro, Marco & Singh, Sameer & Guestrin, Carlos. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. 97-101. 10.18653/v1/N16-3020.

[9] Ali, Aida & Shamsuddin, Siti Mariyam & Ralescu, Anca. (2015). Classification with class imbalance problem: A review. 7. 176-204.

[10] Alshawabkeh, Malak & Jang, Byunghyun & Kaeli, David. (2010). Accelerating the local outlier factor algorithm on a GPU for intrusion detection systems. International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS. 104-110. 10.1145/1735688.1735707.

[11] Gupta, B B & Chandra Joshi, Ramesh & Misra, Manoj. (2009). Defending against Distributed Denial of Service Attacks: Issues and Challenges. Information Security Journal: A Global Perspective. 18. 244-247. 10.1080/19393550903317070.

[12] Bischl, Bernd & Mersmann, O & Trautmann, Heike & Weihs, Claus. (2012). Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation. Evolutionary computation. 20. 249-75. 10.1162/EVCO_a_00069.

[13] Bolón-Canedo, Verónica & Alonso-Betanzos, Amparo. (2018). Ensembles for feature selection: A review and future trends. Information Fusion. 52. 10.1016/j.inffus.2018.11.008.

[14] Bredeche, Nicolas & Shi, Zhongzhi & Zucker, Jean-daniel. (2003). Perceptual learning and abstraction in machine learning. 18- 25. 10.1109/COGINF.2003.1225946.

[15] Kolias, Constantinos & Kambourakis, Georgios & Stavrou, Angelos & Voas, Jeffrey. (2017). DDoS in the IoT: Mirai and other botnets. Computer. 50. 80-84. 10.1109/MC.2017.201.

[16] Carlos Molina, Luis & Belanche, Lluís & Nebot, Àngela. (2002). Feature Selection Algorithms: A Survey and Experimental Evaluation.. Second IEEE International Conference on Data Mining (ICDM'02). 4. 306-313. 10.1109/ICDM.2002.1183917.

[17] CHANDOLA, VARUN & Kumar, Vipin. (2009). Outlier Detection : A Survey. ACM Computing Surveys. 41.

[18] Charu C. Aggarwal “Data Mining: The Textbook”, 2015

[19] Douligeris, Christos & Mitrokotsa, Aikaterini. (2004). DDOS Attacks and Defense Mechanisms: a Classification. 190 - 193. 10.1109/ISSPIT.2003.1341092.

[20] Tax, David & Duin, Robert. (2004). Support Vector Data Description. Machine Learning. 54. 45-66. 10.1023/B:MACH.0000008084.60811.49.

[21] Davis, Jesse & Goadrich, Mark. (2006). The Relationship Between Precision-Recall and ROC Curves. Proceedings of the 23rd International Conference on Machine Learning, ACM. 06. 10.1145/1143844.1143874.

[22] Eid, Heba & Hassanien, Aboul Ella & Kim, Tai-Hoon & Banerjee, Soumya. (2013). Linear Correlation-Based Feature Selection For Network Intrusion Detection Model. Communications in Computer and Information Science. 381. 10.1007/978-3-642-40597-6_21.

[23] Galar, Mikel & Fernández, Alberto & Barrenechea, Edurne & Sola, Humberto & Herrera, Francisco. (2012). A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on. 42. 463 - 484. 10.1109/TSMCC.2011.2161285.

[24] Gao, Lianli & Song, Jingkuan & Liu, Xingyi & Shao, Junming & Liu, Jiajun & Shao, Jie. (2015). Learning in High-Dimensional Multimedia Data: The State of the Art. Multimedia Systems. 10.1007/s00530-015-0494-1.

[25] Geurts, Pierre & Ernst, Damien & Wehenkel, Louis. (2006). Extremely Randomized Trees. Machine Learning. 63. 3-42. 10.1007/s10994-006-6226-1.

[26] Guo, Gongde & Wang, Hui & Bell, David & Bi, Yaxin & Greer, Kieran. (2004). An kNN Model-Based Approach and Its Application in Text Categorization. 559-570. 10.1007/978-3-540-24630-5_69.

[27] Guyon, Isabelle & Elisseeff, André. (2003). An Introduction of Variable and Feature Selection. J. Machine Learning Research Special Issue on Variable and Feature Selection. 3. 1157 - 1182. 10.1162/153244303322753616.

[28] Wei, Hua-Liang & Billings, S.A.. (2005). Feature subset selection and ranking for data dimensionality reduction.

[29] Nguyen, Ha-Nam & Ohn, Syng-Yup. (2006). DRFE: Dynamic Recursive Feature Elimination for Gene Identification Based on Random Forest. 4234. 1-10. 10.1007/11893295_1.

[30] Bahsi, Hayretdin & Nõmm, Sven & Benedetto La Torre, Fabio. (2018). Dimensionality Reduction for Machine Learning Based IoT Botnet Detection. 1857-1862. 10.1109/ICARCV.2018.8581205.

[31] He, Xiaofei & Cai, Deng & Niyogi, Partha. (2005). Laplacian Score for Feature Selection.. proceeding of Advances in Neural Information Processing Systems. Vol. 18.

[32] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016

[33] Guyon, Isabelle & Weston, Jason & Barnhill, Stephen & Vapnik, Vladimir. (2002). Gene Selection for Cancer Classification Using Support Vector Machines. Machine Learning. 46. 389-422. 10.1023/A:1012487302797.

[34] Kevin P. Murphy "Machine Learning. A Probabilistic Perspective", The MIT Press, 2012.

[35] Kloft, Marius & Brefeld, Ulf & Düssel, Patrick & Gehl, Christian & Laskov, Pavel. (2008). Automatic feature selection for anomaly detection. Proceedings of the ACM Conference on Computer and Communications Security. 71-76. 10.1145/1456377.1456395.

[36] Kohavi, Ron & John, George. (1997). Wrappers for Feature Subset Selection. Artificial Intelligence. 97. 273-324. 10.1016/S0004-3702(97)00043-X.

[37] L. Ladla and T. Deepa “Feature Selection Methods And Algorithms”, International Journal on Computer Science and Engineering (IJCSE), vol.3(5), 2011, pp. 1787-1797.

[38] Sun, Li & Du, Qinghe. (2018). A Review of Physical Layer Security Techniques for Internet of Things: Challenges and Solutions. Entropy. 20. 730. 10.3390/e20100730.

[39] Liaw, Andy & Wiener, Matthew. (2001). Classification and Regression by RandomForest. Forest. 23.

[40] Mukkamala, Srinivas & Sung, Andrew & Abraham, Ajith. (2005). Cyber Security Challenges: Designing Efficient Intrusion Detection Systems and Antivirus Tools.

[41] Khalid, Samina & Khalil, Tehmina & Nasreen, Shamila. (2014). A survey of feature selection and feature extraction techniques in machine learning. Proceedings of 2014 Science and Information Conference, SAI 2014. 372-378. 10.1109/SAI.2014.6918213.

[42] Hoang Vu, Nguyen & Gopalkrishnan, Vivekanand. (2010). Feature Extraction for Outlier Detection in High-Dimensional Spaces.. Journal of Machine Learning Research - Proceedings Track. 10. 66-75.

[43] Niwattanakul, Suphakit & Singthongchai, Jatsada & Naenudorn, Ekkachai & Wanapu, Supachanun. (2013). Using of Jaccard Coefficient for Keywords Similarity.

[44] Omar, Salima & Ngadi, Md & H Jebur, Hamid & Benqdara, Salima. (2013). Machine Learning Techniques for Anomaly Detection: An Overview. International Journal of Computer Applications. 79. 10.5120/13715-1478.

[45] Powers, David & , Ailab. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. J. Mach. Learn. Technol. 2. 2229-3981. 10.9735/2229-3981.

[46] D. Shanbhogue, Rahul & M. Beena, B. (2017). Survey of Data Mining (DM) and Machine Learning (ML) Methods on Cyber Security. Indian Journal of Science and Technology. 10. 1-7. 10.17485/ijst/2017/v10i35/118951.

[47] Forrest, Stephanie & Hofmeyr, Steven & Somayaji, Anil & Longstaff, Thomas. (2000). A Sense of Self for Unix Processes. Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy.

[48] Lei, Shang. (2012). A Feature Selection Method Based on Information Gain and Genetic Algorithm. Proceedings - 2012 International Conference on Computer Science and Electronics Engineering, ICCSEE 2012. 2. 10.1109/ICCSEE.2012.97.

[49] Mansfield-Devine, Steve. (2016). DDoS goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare. Network Security. 2016. 7-13. 10.1016/S1353-4858(16)30104-0.

[50] Sedgwick, Philip. (2012). Pearson's correlation coefficient. BMJ. 345. e4483-e4483. 10.1136/bmj.e4483.

[51] Shao, Chenhui & Paynabar, Kamran & Kim, Tae Hyung & Judy) Jin, Jionghua & Jack Hu, S & Patrick Spicer, J & Wang, Hui & A. Abell, Jeffrey. (2013). Feature selection for manufacturing process monitoring using cross-validation. Journal of Manufacturing Systems. 32. 550–555. 10.1016/j.jmsy.2013.05.006.

[52] Spognardi, Angelo & De Donno, Michele & Dragoni, Nicola & Giaretta, Alberto. (2017). Analysis of DDoS-Capable IoT Malwares. 807-816. 10.15439/2017F288.

[53] Tharwat, Alaa. (2016). Principal component analysis - a tutorial. International Journal of Applied Pattern Recognition. 3. 197. 10.1504/IJAPR.2016.079733.

[54] Tsimbalist, Sergei (2019) "Detecting, Classifying, and Explaining IoT Botnet Attacks Using Deep Learning Methods Based on Network Data", https://digi.lib.ttu.ee/i/?12113 (05.03.2019)

[55] Türk, Ahmet & Ozkan, Kemal. (2015). Pre-Processing Methods for Imbalanced Data Set of Wilted Tree. 10.13140/RG.2.1.2204.1684.

[56] Upton, Graham; Cook, Ian (1996). Understanding Statistics. Oxford University Press. p. 55. ISBN 0-19-914391-9.

[57] Visa, Sofia & Ramsay, Brian & Ralescu, Anca & Knaap, Esther. (2011). Confusion Matrix-based Feature Selection.. CEUR Workshop Proceedings. 710. 120-127.

[58] Meidan, Yair & Bohadana, Michael & Mathov, Yael & Mirsky, Yisroel & Shabtai, Asaf & Breitenbacher, Dominik & Elovici, Yuval. (2018). N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE Pervasive Computing. 17. 12-22. 10.1109/MPRV.2018.03367731.

[59] Mirsky, Yisroel & Doitshman, Tomer & Elovici, Yuval & Shabtai, Asaf. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection.

[60] Yang, Shengping & Berdine, Gilbert. (2017). The receiver operating characteristic (ROC) curve. The Southwest Respiratory and Critical Care Chronicles. 5. 34. 10.12746/swrccc.v5i19.391.

[61] Zhang, Dell & Wang, Jun & Zhao, Xiaoxue. (2015). Estimating the Uncertainty of Average F1 Scores. 10.1145/2808194.2809488.

Appendix 1 – Sequential Forward Feature Selection

Greedy sequential forward feature selection method.

```
def select_forward(self, log_dir, features_subset_dir,
                   clf_name, save_output=False):
    current_set_of_features = []
    best_subset = (0, [])
    for i in range(0, len(self.X.columns)):
        feature_to_add = -1
        best_score = 0
        for kth_feature in range(0, len(self.X.columns)):
            if kth_feature not in current_set_of_features and
            \len(current_set_of_features) < 5:
                score = self.calculate_cross_val_f1_score\
                    (current_set_of_features[:], kth_feature)
                set_of_features = current_set_of_features[:]
                set_of_features.append(kth_feature)
                # score upper bound
                if score >= best_score and
                  \len(current_set_of_features) < 5:
                    best_score = score
                    feature_to_add = kth_feature
        current_set_of_features.append(feature_to_add)
        if best_subset[0] >= best_score:
            pass
        else:
            best_subset = (best_score,\
            current_set_of_features[:])
```

Figure 26: Sequential forward feature selection

Appendix 2 – Sequential Backward Feature Selection

Greedy sequential backward feature selection method.

```
def select_backward(self, log_dir, features_subset_dir,
                    clf_name, save_output=False):
    current_set_of_features = []
    for i in range(0, len(self.X.columns)):
        current_set_of_features.append(i)
    best_subset = (self.calculate_cross_val_f1_score
                   (current_set_of_features[:]),
                   current_set_of_features[:])
    for i in range(0, len(self.X.columns)):
        feature_to_remove = -1
        best_score = 0
        if len(current_set_of_features) > 5:
            for kth_feature in range(0, len(self.X.columns)):
                if kth_feature in current_set_of_features:
                    temp_features = current_set_of_features[:]
                    temp_features.remove(kth_feature)
                    score = self.calculate_cross_val_f1_score\
                        (temp_features)
                    if score > best_score and \
                            len(current_set_of_features) > 5:
                        best_score = score
                        feature_to_remove = kth_feature
            current_set_of_features.remove(feature_to_remove)
        if best_subset[0] <= best_score and \
                len(current_set_of_features) <= 5:
            best_subset = (best_score,
                           current_set_of_features[:])
```

Figure 27: Sequential backward feature selection

Appendix 3 – Heuristics Calculation For Wrapper Method

Heuristic calculation based on stratified k-fold cross-validation F1-score.

```
def calculate_cross_val_f1_score(self, current_set,
feature_to_add=None):
    if feature_to_add is not None:
        current_set.append(feature_to_add)
    X_reduced = self.X.iloc[range(0, len(self.X)),
      current_set]
    score = cross_val_score(self.estimator, X=X_reduced,
      y=self.Y, cv=self.skf,
      scoring=make_scorer(f1_score, average='macro'))
    return np.average(score)
```

Figure 28: Heuristic calculation based on stratified k-fold cross-validation F1-score