IoT Intrusion Detection Competition using Machine Learning

Applied Machine Learning – Group Practical Project

Timothy Chan – *MSc in Data Science*, Ian Dickerson – *MSc in Advanced Computing Technologies*, Mike Jun Ming – *MSc in Computing for Financial Services*, Michelangelo Rubino – *MSc in Data Science* and Cosmin Stanciu – *MSc in Data Science*

STRUCTURE

This document has been subdivided into ten parts: an introduction, two group components, five individual components and two appendixes. They have been listed with the responsible and the word count:

I. Overview;
II. Planning – Group – 832 words;
III. Pre-processing – Michelangelo Rubino, Student No. 13140302 – 943 words;
IV. Selecting features – Ian Dickerson, Student No. 12407190 – 643 words;
V. Exploring and selecting ML algorithms - Timothy Chan, Student No. 13160525 – 660 words;
VI. Refining algorithms - Cosmin Stanciu, Student No. 13179534 – Missing;
VII. Evaluating model and analyzing the results - Mike Jun Ming, Student No. 13126984 – 557 words;
VIII. Future Work – Group – 300 words;
IX. Appendix A (Figures);
X. Appendix B (Tables).

I. OVERVIEW

THE Internet of Things (IoT) is growing continuously in the last years for a number of aspects: the fields it is applied to (healthcare, agriculture, cars...), the number of devices, the network traffic. At the same time the cyber-attacks against the 802.11 networks are increasing steeply and becoming more and more aggressive. In the most recent literature on the subject the cyber-attacks have been often subdivided into four categories, based on the execution mode [1]: injection, flooding, impersonation, passive attack. These are the reasons why the Intrusion Detection System (IDS) is one of the critical components of a wireless network: in the last decade the machine learning techniques, from more traditional ones [1] to neural networks [2], [3], have been applying to IDS as more static prevention systems have shown unsatisfying rates in detecting false positives and false negatives and in adapting to new threats. A clear explanation of the differences between IDSs and the other detection systems can be found in [4].

II. PLANNING

This work uses a reduced version of the AWID dataset and focuses only on the impersonation type of attack. This because the impersonation attacks have been identified as the most dangerous threats to the 802.11 networks [1]. The goal is to build a binary classifier able to identify real intrusions in the network traffic using the machine learning techniques.

It has been agreed to use a Gantt chart (**Fig. 1**) to schedule the workload, even though this kind of chart is more suitable to a traditional waterfall methodology. The Machine Learning (ML) projects instead use an iterative lifecycle: several iterations to find the best settings. The idea in this project is then to spend the first six weeks in building a model with the simplest techniques to have a baseline and then the following twenty days in testing a certain number of more complex algorithms and in adjusting parameters and

hyperparameters.

There are several challenges in this research area which are described in the following subsections along with the approach followed in this work.

A. Known threats vs Unknown threats

The traditional prevention systems are often reliable for the known threats but not for new ones. The method a threat is detected can be categorized into three main types [2]: misuse-based (or signature-based), anomaly-based, specification-based. The first group is considered reliable in detecting threats already known, though not the same can be said for the new ones. The third group relies on a set of rules manually written and updated frequently. These two methods have been discarded, then the chosen one for this work is the anomaly-based one, which uses the machine learning techniques to build the user behavior considered “normal”, while the others are labelled as “threat”.

B. Feature selection

This is one of the most delicate stages in the ML process. The general question is whether to use every feature in the dataframe to identify the threat or just part of them, typically those considered more relevant for the class. In the previous literature, the prevalent choice is to use only a certain number of features, like 12-22 in [2], 10-20 in [3] or 36 in [5]: the reasons are summarized in [6]: “The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data”. In this work it has been decided to use from 20 to 40 features based on a mix of criteria (correlation, Mutual Information, PCA).

C. The algorithm to use

Due to the limited amount of time it is not possible to test every machine learning technique, the focus will be on those more frequently used for binary classifiers.

D. Model Size

The traffic of networks 802.11 is generated by portable devices having a small memory where the IDS can be implemented, then the need for light models is crucial and is strictly linked to the feature selection and the algorithm to use. A light model seems more appropriate for this kind of projects for the reasons previously described, provided that this is not to detriment of the performances: not necessarily a lighter model has low performances and a more complex one has better performances, then it is necessary to find a trade-off between the complexity of the model and the results.

E. Precision vs Recall

It is commonly accepted that for binary classifiers the Accuracy is misleading and may lead to wrong conclusions about a model [7] as it detects only the correct predictions, positive or negative, while it says nothing about the false positives and negatives. For this reason, it is important to focus on two more relevant metrics, Precision and Recall. Precision and Recall show respectively the proportion of relevant instances out of the total number of instances and the proportion of relevant instances out of the total number of relevant instances. High Precision means a low false positive rate, while high recall means a low false negative rate. These metrics are like two opposing forces as increasing the first one decreases the second one and vice versa. This is known as the Precision/Recall trade-off and it is often necessary to establish the metric to focus on. For a binary classifier like the one proposed, the question should be: is it desirable to have a low false positive rate or a low negative rate? Is it desirable to see a lower number of normal behaviors wrongly classified as threats or a lower number of real threats wrongly classified as normal behaviors? It is undoubtedly better to focus on the recall as this would limit the number of real threats misclassified as non-threats.

F. ML knowledge vs Domain knowledge

This is a wide topic and is out of the scope of this paper. It can be said however that a certain knowledge of the domain where the ML model is

applied is beneficial to the accuracy ([8], [9]).

III. Pre-processing

AWID is a public collection of standard data sets often used to evaluate the performances of an IDS. For this project it will be used the reduced version AWID-CLS, comprising a training and a test set. The class is the column 155:

Dataset	Obs.	Features	Classes
AWID-CLS-Train	97,044	152	1
AWID-CLS-Test	40,158	152	1

After loading the data through the pandas library and creating a dataframe, it is possible to see that both the datasets are perfectly balanced, 50% of the observations classified as 1, 50% as 0. This is an important aspect as unbalanced datasets may lead to wrong predictions. The dataframe contains no nulls, then it is not necessary to handle them. The first exploration of the raw data can be done through the pandas functions info() and head(). The describe() function shows the descriptive statistics and the first thing to notice is that many features contains only zeros, then the features are useless. They can be safely removed (74), ending up with 78 from the original 152. In the following figure three features with no values have been highlighted:

	1	2	3	5	6
count	97044.0	97044.0	97044.0	97044.000000	97044.000000
mean	0.0	0.0	0.0	0.006252	0.006252
std	0.0	0.0	0.0	0.015541	0.015541
min	0.0	0.0	0.0	0.000003	0.000003
25%	0.0	0.0	0.0	0.001442	0.001442
50%	0.0	0.0	0.0	0.003706	0.003706
75%	0.0	0.0	0.0	0.005916	0.005916
max	0.0	0.0	0.0	0.978440	0.978440

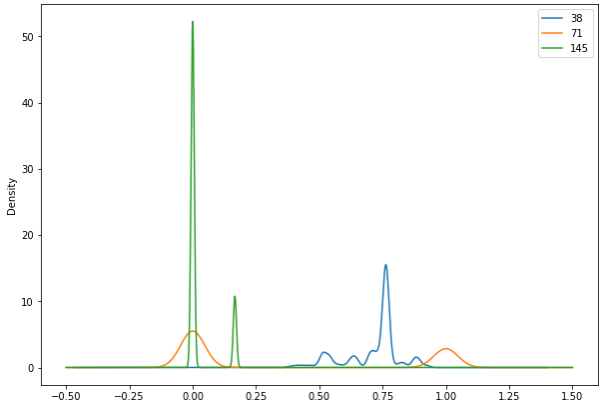
By combining the pandas dataframe functions loc() and duplicated() it is possible to detect duplicated features, having exactly the same values. These features can be removed as well, leading to a dataframe with only 64 features and the class. Decreasing the number of features is an important stage for some reasons:

- it reduces the processing time;
- it simplifies the model, avoiding overfitting;
- focusing only on relevant features, it improves the results.

The real selection of features will happen in the next stage, though removing useless features can be done in the preprocessing stage. The unique values show that the features can be categorized into three main groups:

1. continuous values between 0 and 1 (ex: 38);
2. only two values, 0 or 1 (ex: 71);
3. only a few values, integer or floating, between 0 and 1 (ex: 145).

As every feature show only values between 0 and 1, the data does not need to be rescaled. This term refers to the techniques used to give the features the same scale (typically between 0 and 1), which may give the model some improvements [10]. There is often confusion when using the terms scaling, standardizing and normalizing as authors tend to use them interchangeably, then the meanings explained in [11] will be used. The three abovementioned groups have different distributions and none is normal:



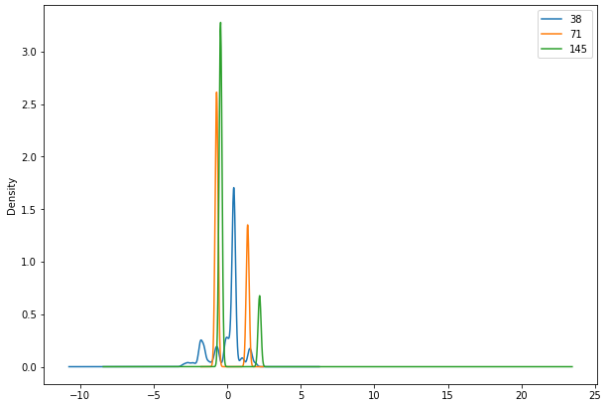
Rather than using the correlation matrix plot or the scatter plot matrix, which are not easy to visualize due to the number of features, it has been computed the correlation among pairs of features and established a threshold of 0.9: for each pair having a correlation equal or higher than the threshold, the second feature will be removed. This will avoid multicollinearity: in [12] multicollinearity is defined as “a phenomenon in which one feature variable in a regression model is highly linearly correlated with another feature variable” and this may affect the model and lead to

misleading results. Multicollinearity will never improve the results, on the contrary avoiding it may bring advantages to the model. There are several techniques used to avoid multicollinearity: for this project it will be simply removed the second feature of the highly correlated pairs. When dealing with dataframes showing a gaussian distribution, the Pearson correlation is preferred, here instead the non-parametric correlations will be used. The pandas library contains correlation methods different from Pearson's such as the Kendall rank correlation coefficient (Kendall's tau coefficient) or the Spearman rank correlation coefficient (Spearman's rho). These methods do not rely on the normal distribution assumption and seem more appropriate for this dataframe. A more detailed explanation of the non-parametric methods can be found in [13]. The Kendall and Spearman correlations are quite similar and indeed the features exceeding the 0.9 threshold are the same, apart from one. The 19 features in common will be removed (**Tab. 1**).

After dropping the unnecessary features, the dataframe has 45 features and 1 class (**Tab. 2-3**). Plotting the features and computing their skewness confirm that no features have a normal distribution: the histogram, the density and the box and whisker plots of the features can be seen in the Appendix A (**Fig. 2-4**). To improve the data distribution some techniques have been tested using the scikit-learn functions: yet, it has been arranged to use only the standardization through the StandardScaler function, which removes the mean and scales to unit variance. In the following table a short explanation for each technique and why it has been discarded:

Function	**Description**	**Explanation**
Binarizer	Continuous values are converted into only two values, 0 and 1, based on if less or more than 0.50	No improvement, on the contrary the results are slightly worse.
Power Transformer	It helps the data to look more Gaussian-like by using two methods: Box-Cox or Yeo-Johnson	Similar to StandardScaler
MinMaxScaler	It scales data to values between 0 and 1	Not necessary as the feature values are all included between 0 and 1
RobustScaler	Similar to MinMaxScaler, in addition it is useful to deal with outliers	Not necessary
Normalizer	It helps the data to look more Gaussian like	It works at row level. Not recommended [11]

The standardization through the StandardScaler will help the dataframe to show a distribution closer to the normal one (mean = 0, standard deviation = 1). It will not be a perfect normal distribution, but this is better than the original one as shown by the following figure with the same features previously analyzed:



The four steps for preprocessing the dataset have been included in the function preproc_df(), which can be found in the .ipynb file.

IV. Selecting Features

It is possible to train a classification model on a large number of features, but this has clear disadvantages. The time taken for training can be prohibitive. And there is a danger that the model will fit the noise in irrelevant features, reducing generalizability. A selection of fewer, demonstrably relevant features should improve classification performance and processing time while in some cases also yielding better interpretability.

To avoid feature selection becoming a processing bottleneck in this project, a broadly filter-based approach was adopted. That is, testable characteristics of individual features were used as the primary selection method. A logistic regression classifier, trained and tested on a 60/40 split of the training data, was used to sense-check the feature selection and fine tune the thresholds used. The

recall of the classifier was used as the principal metric, as the priority was to make sure that as many intrusions are detected as possible, but in practice precision was equally high. The disadvantage here is that a set of features that leads to good recall from a logistic regression model might not transfer well to a more complex classification model like Random Forest or a neural network. The filter methods employed were:

- *Statistical hypothesis testing* - If the distribution of values in a feature does not change under the two classes, then the feature will be of no value for the classifier model. Independent t-tests are a computationally fairly cheap process to eliminate such features. Tests were carried out for all the features remaining after preprocessing and the features ordered by test statistic. Since almost all of the features showed significantly -- that is, with vanishingly low p-value -- different distributions under the two conditions, this method was of limited value in the present case;
- *Correlation* - A second common and quite efficient method for assessing relevance is to measure the strength of correlation between feature and target classification. Either a strong positive or strong negative correlation with the target suggests that the feature will have predictive value. The linear correlation measures employed here was Kendall's tau and Pearson's rho which gave sufficiently different results to warrant the inclusion of both. There are two obvious limitations to these measures: firstly, that as statistical measures they are not designed to be applied to a binary variable, and secondly that they measure linear correlation, and may miss a relationship with the target variable that is not linear. The first weakness appears not to have too great an impact in practical applications. To account for the second possibility, another measure of relationship is needed.
- *Mutual Information* - To capture relationships with the target variable that are not linear, or otherwise not captured by correlation, the Mutual Information between individual features and the target class was calculated, following Parker et al. One possible weakness of this approach may be that the highest-ranked features for MI may share information with each other as well as the target and that lower ranked features may encode genuinely different information. In the present case, measurement of MI between so many features was found to be impractical.
- *PCA* - To compensate for the possible weaknesses in MI and linear correlation identified above, Principal Component Analysis was used to capture any remaining information from features that had not so far been identified. This unsupervised technique captures as much variance as possible from high dimensional data into a lower dimensional space.

Good recall was found by exploring combinations of A) five to ten features with high absolute value of Kendall's tau or Pearson's rho with the target, B) five to ten features with highest mutual information with the target, and C) the first ten to twenty principle components of the continuous features. A Python function was written to allow the developer of the model to easily vary the thresholds to obtain slightly different feature sets for training.

V. Exploring and Selecting ML Algorithms

A. Random Forests

Previous research [14] and alert detection studies [15] suggests trees perform well for classification with mixed data. RandomForests provide a model with enough flexibility without making assumptions about the data, with tools to reduce high variance on large data sets.

Untrimmed trees tend to overfit and produce many leaf nodes. RandomForests improve over traditional trees, through tools to reduce overall error and variance; bagging and random feature sampling. Generating multiple trees, through

random sampling with replacement, of subsets of test data and features. This reduces the overall reliance of prediction on one single model.

RandomForests benefit from identifying feature importance, unlike other models, but are relatively blackbox in nature; there is little scope to amend parameters.

B. K-Nearest Neighbors (K-NN)

K-NN was selected for its relative simplicity compared to other blackbox models. The algorithm works by grouping datapoints, based on other datapoints which are 'close' to one another'. Each new datapoint is classified by its K nearest partners. The model therefore only requires setting a distance calculator and K.

Due to classification being solely reliant on a simple distance calculation, there is a likelihood that the model is not flexible enough for a large data set, with a large feature space. A small change in a binary feature across data rows, may lead to only a small change in distance between datapoints. This binary feature, however, may be critical in the true categorization model.

The initial use of Euclidean distance poses issues in measuring distance between continuous and categorical variables. Selection of a distance calculator such as a Gower [16], may reduce the disadvantages outlined.

C. Binary Linear Logistic Regression

LR was selected due to its probabilistic nature. The calculation of logits makes the model relatively easy to interpret and highlight feature importance.

The assumptions within LR however, poses issues for the large data set with a high number of features. LR requires continuous independent variables to have linear relationships with their logits. Secondly, there cannot be multi-collinearity between independent variables. Both can lead to high standard errors, and therefore poor prediction reliability [17].

Due to the data set being unlabeled, and assumptions made about the types data contained within the set, I is not possible to ensure that these assumptions are met.

D. MLP Neural-Network (Stochastic Gradient Descent) [18]

A NN was selected due to its flexibility on large data sets with a higher number of features. NN and Deep NN has been particularly successful in classification accuracy tasks [19]. Further and recent research has shown the increased capabilities of Convolutional Neural Networks (CNN) and Artificial Neural Networks (ANN) in intrusion detection [20].

Neural-Networks complexity, and computational cost lead to significant disadvantages. Owing to backpropagation, the algorithm classifies based on patterns identifying patterns between layers, and patterns between neurons. Many neurons and layers can lead to a computationally expensive model, with a tendency to overfit. Owing to how these patterns are formed, NN are blackbox; little information can be gained from viewing these patterns.

The ability to balance model overfitting, variance and computational cost, therefore, requires time spent iteratively finetuning the selection of the number of layers and neurons used in each layer.

E. Support Vector Machine (SVM)

SVM was selected due to its relative simplicity compared to blackbox models. SVM transforms the data into N dimensional space (N=Number of features). The transformed data is separated by the maximal margin hyperplane, with threats and non-threats falling on either side.

SVM works under the assumption the transformed data is separable by a hyperplane. This has benefits; It is easy to understand and tends to not overfit.

However, difficulties arise in a large data set with selecting the correct kernel (shape) of the hyperplane and gamma (smoothness of the hyperplane) – requiring numerous iterations to find the correct parameters. The assumption of the data being separable, in itself, also limits the models flexibility compared to blackbox models; RF and NN.

A model initialization and parameter

justification can be found in **Fig. 5**.

VI. Refining algorithms

Missing…

VII. Evaluating model and analyzing the results

Different method was considered when evaluate the overall performance, like accuracy, precision, F1 score, false alarm, true positive, false negative and even the time use to build/train the model. The component used is Windows 10 based system 64 bits system with a CPU of Intel Core i5 2.6 GHZ and with memory card of 8 Gb. When validate the model, validation and test. I kept the training data and test data separate, so that the model was produced and generated based on the train dataset during this procedure. Test data was kept away to prevent any incorrect estimate and error. Cross validation used while on the first stage validation process, and the result of accuracy is for different model are shown at the table below.

As the accuracy figures are all very simpler, but there is no distinction between classes, this might be due to the misclassification. It has been decided to use a confusion matrix, as it can provide more information on correct or incorrect on each class. The number shows the misclassification number is quite high at logistic regression model, but classifier behavior seems positive as the ROC score is 99%. It is not possible to see much difference on per class accuracy either, means the variation of accuracy is low for each class. From the result the micro average in this case are identical to the accuracy, expected as the data are balanced. For each model it has been also provided the precision, recall and f1 metrics, this is important during evaluation, f1 is high so both precision and recall are high, all the models have pretty high percentage of result that are relevant and shows the result is classified by the algorithm correctly. However, it has been discovered that the time taken to build this svm and neural network model will need between 20-30% extra during this process. While the rest of the model was measured below 3 seconds which indicate the computation pressure power that take into consideration at later stage.

	RF	BLLR	k-NN	NN	SVM
validation	0.99922	0.9792	0.998722	0.99897	0.99835
misclass	19	514	31	25	40
Pre-class accuracy	0.99922		0.998715	0.998964	
test	0.5	0.49993	0.49990	0.5	0.5

As the model is built and trained by the features and PCA (Principal Component Analysis) value, PCA are built from all the pre-processed columns in test, so 5 pcas value is different then the one use to build the model with standard scaler applied. This is due to the variables is converted from correlated to uncorrelated one through some statistical procedure. And then joint this with the features identified from the selection stage when used from training and to build the model. From the accuracy result on test dataset, they are all simpler. To increase the accuracy, can be achieved by increasing variance, for SVM increasing parameter = 'C' can put up variance, and reducing gamma.

It has been picked the Neural Network by looking at different dimension, it is not the best on misclassification, but it can do more neutron and layers depend on the data, Tree that can go more depth or maybe the cost is higher on SVM and variance increase is possible. In combination of lightweight and accurate models plus the maintenance might need in the future, Neural network performed best out of the 5 model have been trained.

VIII. Future Work

In this paper a binary classifier for detecting intrusions in the 802.11 networks has been proposed. After preprocessing the dataset, the number of features to use (from 20 to 40) has been selected based on a three-filter method (correlation, MI, PCA). The five algorithms analyzed have all strengths and weaknesses: based on performances, the best model seems to be the Neural Network, even though it requires from 20 to 30% of additional time (along with the SVM). However, due to the limited amount of time and to the absence of the Refining algorithms part, the accuracy using the test dataset is not very high (around 50%). It would be necessary in the future

to test the models changing parameters and hyperparameters in order to obtain a high Recall: this metric indeed is the one to focus on when working with this type of binary classifier. A high Recall would output a low number of real threats wrongly classified as non-threats.

Also, recent trends show the development of the use of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) in threat detection. The use of Long Short Term Memory (LSTM) networks, with historical layer learning, has been shown to improve upon the MultiLayer Perceptron (MLP) classifier in isolation [21]. CNNs have been adapted to threat detection and used in a multi algorithmic approach with LTSM methods, with better overall results [22].

The use of MLP deep learning as an unsupervised first step has shown to increase threat detection over 30% in accuracy, over an MLP NN in isolation. When an MLP NN unsupervised with a final layer of neurons is higher than the number of dependent categories, is then passed through a logistic regressor [22].

The use of CNN and RNN, and NN as an unsupervised tool should be considered.

IX. Appendix A (Figures)

GANTT CHART

PROJECT TITLE	Impersonation Detection		MODULE NAME	AML		
OWNER	Michelangelo Rubino		DATE	18/11/19		

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Team Forming**					100
1.1	Project Kick-off	Group	18/11/19	25/11/19	7	100%
2	**Planning**					
2.1	Scope, resources, tasks of the projects	Group	25/11/19	29/11/19	4	100%
2.2	Define responsibles of the main tasks	Group	29/11/19	30/11/19	1	100%
2.3	Define goals and trade-offs	Group	30/11/19	02/12/19	2	100%
3	**Pre-processing**					
3.1	Load data and use descriptive statistics	MR	02/12/19	03/12/19	1	100%
3.2	Investigate data	MR	03/12/19	04/12/19	1	100%
3.3	Show data graphically	MR	04/12/19	05/12/19	1	100%
3.4	Use correlation to discard features	MR	05/01/20	06/01/20	1	100%
3.5	Use transformation techniques (binarization...)	MR	06/12/19	09/12/19	3	100%
4	**Selecting Features**					
4.1	Analyze the techniques to apply	ID	09/12/19	10/12/19	1	100%
4.2	Choose a small number of techniques to test	ID	10/12/19	11/12/19	1	100%
4.3	Test the techniques	ID	11/12/19	14/12/19	3	100%
4.4	Select the best one	ID	14/12/19	16/12/19	2	100%
5	**Exploring and Selecting ML Algorithms**					
5.1	Analyze different algorithms	TC	16/12/19	18/12/19	2	100%
5.2	Choose a small number of algorithms	TC	18/12/19	19/12/19	1	100%
5.3	Test the algorithms	TC	19/12/19	22/12/19	3	100%
5.4	Select the best one	TC	22/12/19	23/12/19	1	100%
6	**Refining ML Algorithms**					
6.1	Analyze components (layers, units per layers...)	CS	23/12/19	25/12/19	2	100%
6.2	Analyze hyperparameters	CS	25/12/19	27/12/19	2	100%
6.3	Iterate and optimize	CS	27/12/19	29/12/19	2	100%
7	**Evaluating model and analysing the results**					
7.1	Evaluate performances	CS	29/12/19	31/12/19	2	100%
7.2	Interpret the results	CS	31/12/19	02/01/20	2	100%
7.3	Compare results with benchmarks	CS	02/01/20	04/01/20	2	100%
8	**Refine the model**					
8.1	Improve the performances tweaking parameters	Group	04/01/20	14/01/20	10	100%
9	**Documentation revision**					
9.1	Merge the individual parts into a single document	Group	08/01/20	18/01/20	10	100%
10	**Future Work**					
10.1	Final chapter	Group	17/01/20	19/01/20	2	100%
11	**Submission**	Group	19/1/2020	19/1/2020		

Fig. 1

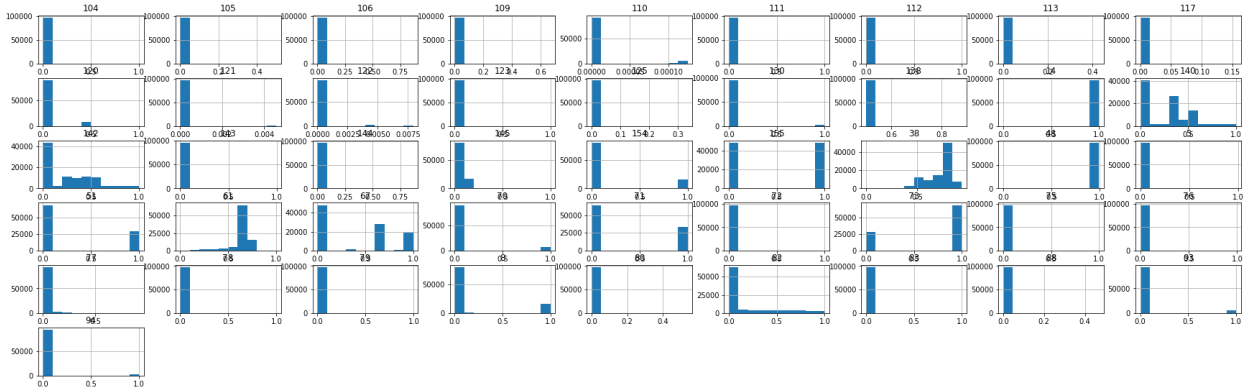


Fig. 2

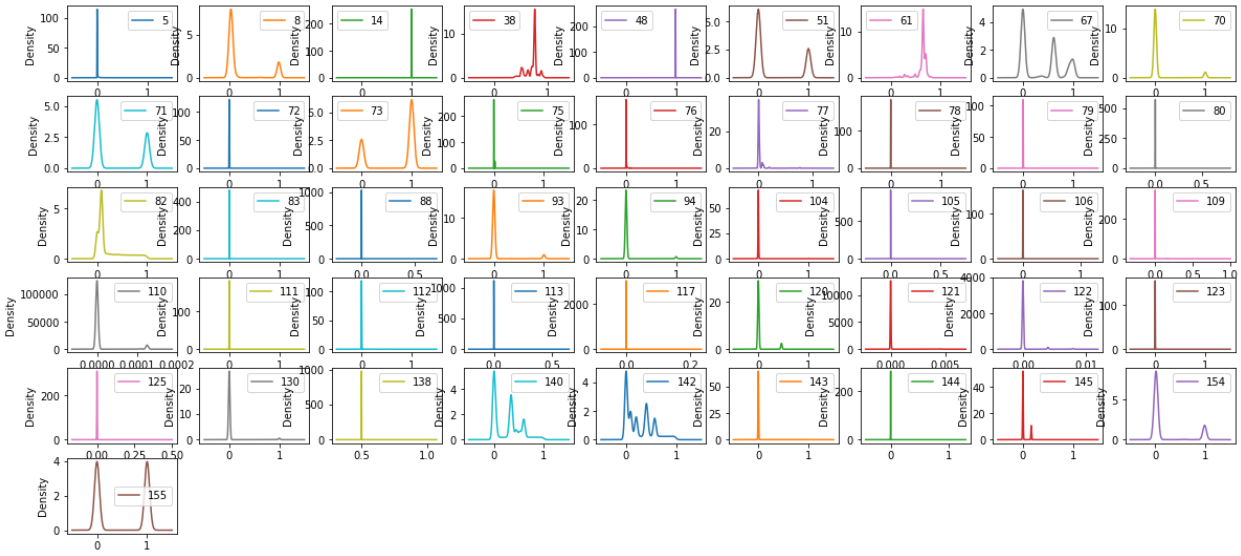

Density
5
8
14
38
48
51
61
67
70
71
72
73
75
76
77
78
79
80
82
83
88
93
94
104
105
106
109
110
111
112
113
117
120
121
122
123
125
130
138
140
142
143
144
145
154
155

Fig. 3

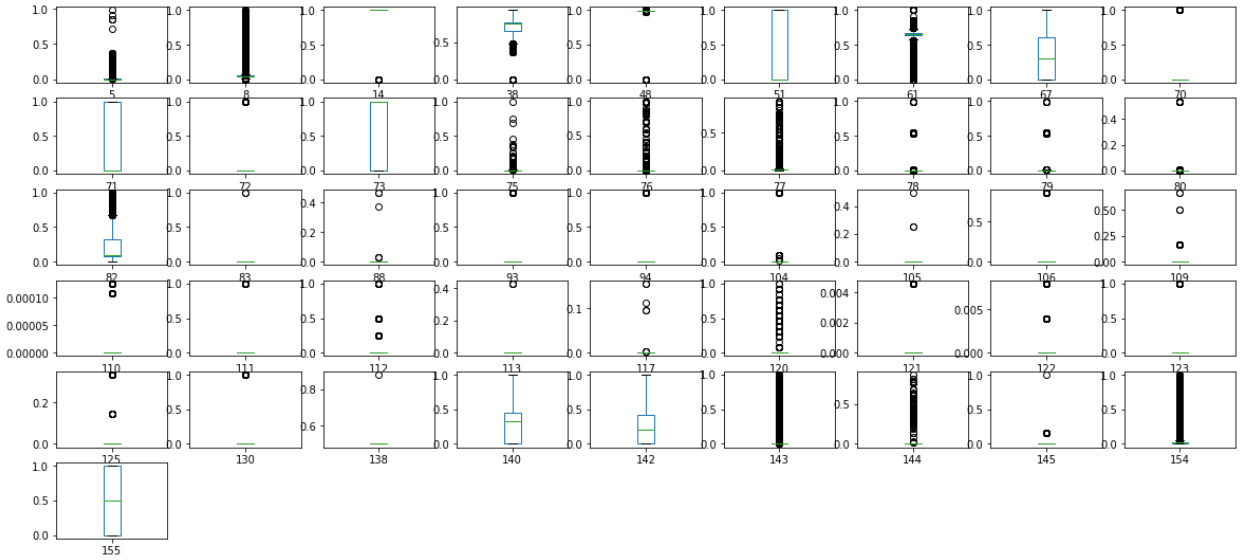

5
8
14
38
48
51
61
67
70
71
72
73
75
76
77
78
79
80
82
83
88
93
94
104
105
106
109
110
111
112
113
117
120
121
122
123
125
130
138
140
142
143
144
145
154
155

Fig. 4

Model	Parameters	Justification
Random Forest	(1) Untrimmed, (2) Depth limited to 6	(1) Test the expectation of a very accurate model and explore feature importance. (2) To initially access how trimming reduces the precision and recall.
K-NN	Euclidean Distance, 5 neighbours	Euclidean distance for it's simplicity. 5 Neighbours, square root of number of features.
Binary Logistic Regression		Explore feature feature importance.
Neural Network	2 Layers, 5 Neurons. 1000 Iterations of BackPropogation	Small number of layers and neurons to access suitability and reduce overfitting.
SVM	Kernel: RBF, Gamma = 1/n, Cost = 1	Flexible kernel due to high dimensionality. Low cost to assess separability of data and low gamma to reduce flexbility of separating hyperplane

Fig. 5

X. Appendix B (Tables)

TABLE 1

Features having high Kendall and Spearman correlation to be dropped

Feature	Kendall corr >= 0.9	Spearman corr >= 0.9
47	Yes	Yes
50	Yes	Yes
64	Yes	Yes
66	Yes	Yes
67	No	Yes
68	Yes	Yes
84	Yes	Yes
86	Yes	Yes
90	Yes	Yes
97	Yes	Yes
98	Yes	Yes
107	Yes	Yes
108	Yes	Yes
118	Yes	Yes
119	Yes	Yes
126	Yes	Yes
127	Yes	Yes
128	Yes	Yes
129	Yes	Yes
141	Yes	Yes

TABLE 2

Train dataframe features after the preprocessing stage (45 + 1 class)

5	frame.time_delta
8	frame.len
14	radiotap.length
38	radiotap.mactime
48	radiotap.channel.freq
51	radiotap.channel.type.ofdm
61	radiotap.dbm_antsignal
67	wlan.fc.subtype
70	wlan.fc.retry
71	wlan.fc.pwrmgt
72	wlan.fc.moredata
73	wlan.fc.protected
75	wlan.duration
76	wlan.ra
77	wlan.da
78	wlan.ta
79	wlan.sa
80	wlan.bssid
82	wlan.seq
83	wlan.bar.type
88	wlan.ba.bm
93	wlan_mgt.fixed.capabilities.privacy
94	wlan_mgt.fixed.capabilities.preamble
104	wlan_mgt.fixed.listen_ival
105	wlan_mgt.fixed.current_ap
106	wlan_mgt.fixed.status_code
109	wlan_mgt.fixed.aid
110	wlan_mgt.fixed.reason_code
111	wlan_mgt.fixed.auth.alg
112	wlan_mgt.fixed.auth_seq
113	wlan_mgt.fixed.category_code
117	wlan_mgt.fixed.sequence
120	wlan_mgt.ds.current_channel
121	wlan_mgt.tim.dtim_count
122	wlan_mgt.tim.dtim_period
123	wlan_mgt.tim.bmapctl.multicast

125	wlan_mgt.country_info.environment
130	wlan_mgt.rsn.akms.type
138	wlan_mgt.tcprep.trsmt_pow
140	wlan.wep.iv
142	wlan.wep.icv
143	wlan.tkip.extiv
144	wlan.ccmp.extiv
145	wlan.qos.tid
154	data.len
155	class

TABLE 3
Test dataframe features after the preprocessing stage (43 + 1 class)

5	frame.time_delta
8	frame.len
14	radiotap.length
38	radiotap.mactime
47	radiotap.datarate
48	radiotap.channel.freq
51	radiotap.channel.type.ofdm
61	radiotap.dbm_antsignal
64	wlan.fc.type_subtype
67	wlan.fc.subtype
70	wlan.fc.retry
71	wlan.fc.pwrmgt
72	wlan.fc.moredata
73	wlan.fc.protected
75	wlan.duration
76	wlan.ra
77	wlan.da
79	wlan.sa
80	wlan.bssid
81	wlan.frag
82	wlan.seq
94	wlan_mgt.fixed.capabilities.preamble
104	wlan_mgt.fixed.listen_ival
105	wlan_mgt.fixed.current_ap
106	wlan_mgt.fixed.status_code
109	wlan_mgt.fixed.aid
110	wlan_mgt.fixed.reason_code
111	wlan_mgt.fixed.auth.alg
112	wlan_mgt.fixed.auth_seq
120	wlan_mgt.ds.current_channel
121	wlan_mgt.tim.dtim_count
122	wlan_mgt.tim.dtim_period
123	wlan_mgt.tim.bmapctl.multicast
125	wlan_mgt.country_info.environment
128	wlan_mgt.rsn.pcs.count
140	wlan.wep.iv
141	wlan.wep.key
142	wlan.wep.icv
143	wlan.tkip.extiv
144	wlan.ccmp.extiv
145	wlan.qos.tid
148	wlan.qos.ack
154	data.len
155	class
5	frame.time_delta
8	frame.len

XI. REFERENCES

[1] C. Kolias, G.Kambourakis, A.Stavrou and S. Gritzalis, "Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, 2015.

[2] M.E.Aminanto, R.Choi, H.C.Tanuwidjaja, P.D.Yoo, "Deep Abstraction and Weighted Feature Selection for Wi-Fi Impersonation Detection" *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, 2018.

[3] L.R.Parker, P.D.Yoo, T.A.Asyhari, L.Chermak, Y.Jhi, K.Taha, "DEMise: Interpretable Deep Extraction and Mutual Information Selection Techniques for IoT Intrusion Detection" in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, Article no. 98, 2019.

[4] Rao U.H., Nayak U. – "Intrusion Detection and Prevention Systems" in *The InfoSec Handbook*. Apress, Berkeley, CA, 2014.

[5] S. Rezvy, M. Petridis, T. Zebin, Y. Luo, A. Lasebae "An efficient deep learning model for intrusion classification" in *53rd Annual Conference on Information Sciences and Systems (CISS)*, 2019.

[6] I. Guyon, A. Elisseeff – "An Introduction to Variable and Feature Selection" in Journal of Machine Learning Research 3, 2003.

[7] Y. Jankay – "The 3 pillars of Binary Classification: Accuracy, Precision & Recall" (2018, April). https://medium.com/@yashwant140393/the-3-pillars-of-binary-classification-accuracy-precision-recall-d2da3d09f664. [Online].

[8] Anand – "Why domain knowledge is important in Data Science" (2019, March). https://medium.com/@anand0427/why-domain-knowledge-is-important-in-data-science-anand0427-3002c659c0a5. [Online].

[9] T. Blanchard – "The importance of domain knowledge – A healthcare data science perspective" (2017, November). https://data-science-blog.com/blog/2017/11/10/the-importance-of-domain-knowledge-a-healthcare-data-science-perspective/. [Online].

[10] S. Raschka – Department of Statistics, University of Wisconsin-Madison. "About Feature Scaling and Normalization" (2014, July). *https://sebastianraschka.com/Articles/2014_about_feature_scaling.html*. [Online] - e-mail: sraschka@wisc.edu.

[11] J. Hale. "Scale, Standardize or Normalize with Skikit-Learn" (2019, March). https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02 [Online].

[12] J. Saslow. "Collinearity – What it means, why it is bad and how does it affect other models?" (2018, July). https://medium.com/future-vision/collinearity-what-it-means-why-its-bad-and-how-does-it-affect-other-models-94e1db984168 [Online].

[13] J. Brownlee. "How to calculate Nonparametric Rank Correlation in Python" (2018, July). https://machinelearningmastery.com/how-to-calculate-nonparametric-rank-correlation-in-python [Online].

[14] Intrusion Detection in 802.11.

[15] Deep Abstraction and Weighted Feature Selection.

[16] Gower, J. (1971). A General Coefficient of Similarity and Some of Its Properties. Biometrics, 27(4), pp.857-871.

[17] Stoltzfus, J. (2011). Logistic Regression: A Brief Primer. Academic Emergency Medicine, 18(10), p.1101.

[18] Kingma, D. and Ba, J. (2017). Adam: A Method for Stochastic Optimization.

[19] Moradi, M. and Zulkernine, M.,(2014). A Neural Network Based System for Intrusion Detection and Classification of Attacks. Conference: *Proceedings of 2004 IEEE International Conference on Advances in Intelligent Systems - Theory and Applications*.

[20] H., B. and Tindo, G. (2017). Network Intrusion Detection Systems based Neural Network: A Comparative Study. International Journal of Computer Applications, 157(5), pp.42-47.

[21] Lee, B., Amaresh, S., Green, C. and Eagles, D., (2018). Comparative Study of Deep Learning Models for Network Intrusion Detection. SMU Data Science Review, 1(1).

[22] Wu, P. and Guo, H., (2019). LuNet: A Deep Neural Network for Network Intrusion Detection.