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

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. Different categorizations are possible based on different criteria. 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 the 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 (such as HIDS and IPS) 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 decided 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. 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.

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 two most delicate stages in the ML process, the other being the choice of the algorithm. 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 is 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". Another advantage to mention is that a limited number of features will avoid overfitting. To be completed after reading the feature selection part…

C. *The algorithm to use*

To be completed after reading the Selecting ML algorithms part…

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 several reasons previously mentioned in the section B, 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 which are computed through the formulas TP / [TP + FP] and TP / [TP + FN]. 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 clearly better to focus on the recall as this would limit the number of real threats misclassified as non-threats. What have we focused on?

F. ML knowledge vs Domain knowledge

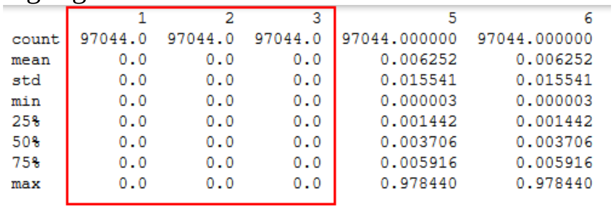
This is a wide topic and is out of the scope of this paper. It can be said however that a certain knowledge of the domain where the ML model is applied is beneficial to the accuracy ([8], [9]).

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 have 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
```

Fig. 1. Features with no values

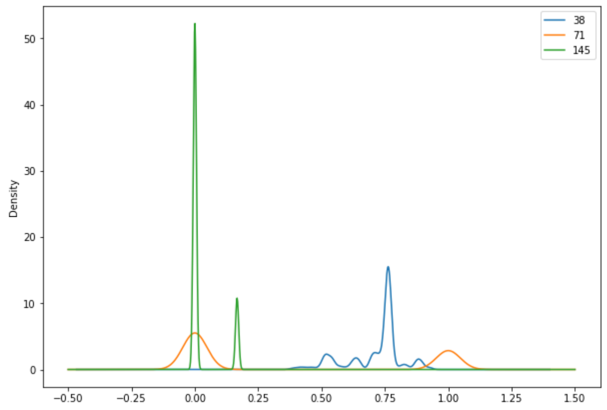
By combining the pandas dataframe functions loc() and duplicated() it is possible to detect duplicated features, having exactly the same values. These features can be removed as well, leading to a dataframe with only 64 features and the class. Decreasing the number of features is an important stage for a few 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 pandas nunique() function show that the features can be categorized into three main groups on the basis of the unique values:

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 equal or higher than 0 and equal or lower than 1, the data does not need to be rescaled. This term refers to the techniques used in machine learning 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 has a normal distribution:



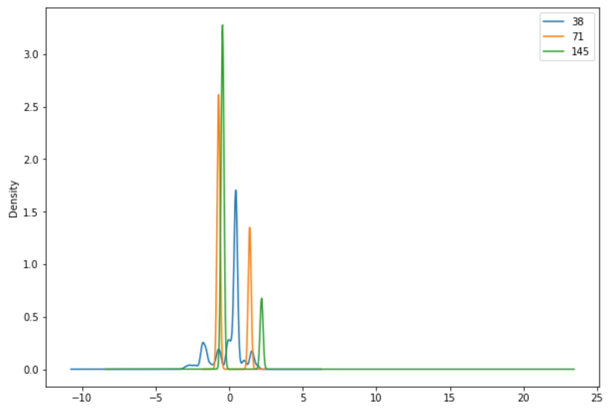
At this point, rather than using the correlation matrix plot or the scatter plot matrix, which are not easy to visualize due to the number of features, it has been computed the correlation among pairs of features and established a threshold of 0.9: for each pair having a correlation equal or higher than the threshold, the second feature will be removed. This will avoid multicollinearity: in [13] 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 is not dangerous and in some cases it may bring advantages to the model. There are several techniques used to avoid multicollinearity: for this project the simpler one has been chosen, that is removing the second feature of the highly correlated pairs. When dealing with dataframes showing a gaussian distribution, the Pearson correlation is typically 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 to analyze this dataframe for the reasons explained earlier in this section. A more detailed explanation of the non-parametric methods can be found in [12]. 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 from the dataframe:

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

After dropping the unnecessary features, the dataframe has 45 features and 1 class. 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: it has been decided 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:



Standardize every features or only the continuous ones?

IV. Selecting features

Ian here…

V. Exploring and selecting ML algorithms

A. *Random Forests MISSING REFERENCES*

Previous research and alert detection studies suggests trees perform well for classification with mixed data. A basic tree works by iteratively splitting the features into partitions, starting with the feature split that reduces the overall misclassification rate, and then continuing down the branches. An untrimmed tree will repeat these splits until each leaf node is its own category, which results in overfitting – leading to high variance on test data. To reduce overcome this trees are typically trimmed to a maximum depth, therefore, only the most important features are selected. This can lead to less accuracy but lower variance.

RandomForests were selected over other trees as they benefit from bagging, and random feature sampling. Bagging trains multiple trees, on sub-sections of the test data (with replacement). As such multiple trees predict a category for each instance of data, reducing the likelihood of overfitting. RandomForests train each tree with a random subset of features, therefore reducing the likelihood of overreliance on a specific feature. This overall leads to a reduced overall error, and additionally, a more flexible but lower variance model.

As such RandomForests provide a model with enough flexibility without making assumptions about the data, with tools to reduce high variance.

B. *K-Nearest Neighbors (K-NN)*

K-NN was selected for its relative simplicity compared to other blackbox models (Neural-Network and RandomForest). The algorithm works by grouping datapoints, based on other datapoints which are ‘close’ to one another’. Each new datapoint is considered grouped by its K nearest partners.

One of the issues for the dataset is the selection and method of calculating distance, and the selection of the K. The base model calculates distance/closeness for each data point, via a vector method, Euclidean distance. Intuitively Euclidean distance isn’t a statistically correct method for judging distance in categorical, and particularly non-ordinal categorical data. Measuring distance using Gower – which does not rely on vector distance, would be an area of future improvement.

Due to classification being solely reliant on a simple distance calculation (whether it is Euclidean or another method Gower), there is a likelihood that the model is not flexible enough for a large data set, with a large feature space. For example, a small change in a binary feature across data rows, may lead to a small change in distance and as such changes in grouping, but it may impact on classification.

C. *Binary Linear Logistic Regression*

Binary logistic regression provides a good base case for classification. Binary logistic regression is a probabilistic model – where features which have the highest log odds of binary classifying a data point have the largest impact. As such, the model is relatively easy to interpret compared other models selected.

A binary logistic regression model relies on the underlying assumption that the data is separable by a line or a plane, in an N-feature space (a boundary can be drawn). Due to the dataset being high dimensionality, this can pose a problem as it can be hard to visualize this plain. Which makes the selection of a linear, quadratic or parabolic model difficult. Selecting an incorrect model will lead to higher misclassification rates.

Logistic Regression benefits from increased flexibility of K-NN with the same benefits of lower computational cost (time and memory). Due to the dataset containing a large number of datapoints and features, if the data can be separate with a hyperplane, it would avoid the computational and difficult to interpret flexible models of NN and RF.

D. *Neural Network*

A simple neural network was selected due to its flexibility on large data sets with a higher number of features. NN has been particularly successful in classification accuracy tasks.

The disadvantage of Neural networks is their blackbox nature. Owing to backpropagation, the algorithm classifies based on patterns identifying patterns between layers, and patterns between neurons. The number of patterns increases with the number of layers and neurons, resulting in a difficulty in interpreting these patterns in line with the underlying features.

This additionally presents difficulties when, selecting the number of layers and neurons used for a large dataset with a large number of features. Many layers and neurons in each layer, would mean more scope to identify patterns in the underlying data. This would increase model accuracy and complexity but be extremely computationally expensive due to backpropagation. Additionally, it would increase model variance, similar to untrimmed trees.

Balancing model overfitting, variance and computational cost, therefore, requires 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 (where N=Number of features), which is separated by the maximal margin hyperplane; with the closest points in each class being the support vectors (the most important for hyperplane calculation). As such, although a non-probabilistic model, SVM relies on the data being separable by a hyperplane, similar to Logistic Regression.

Difficulties a rise in SVM like Logistic Regression – selecting the correct shape or underlying model of the separating plane. SVMs provide a better method of selecting the shape, and smoothness of the hyperplane via a kernel and gamma. As such, being more flexible than logistic regression. Similarly, overfitting can be controlled by adjusting a cost/loss function, limiting/increasing the number of points that can cross the boundary and be misclassified.

Due to the dataset containing many features and datapoints the fine tuning and selection of cost, kernels and gamma (smoothness of the hyperplane), can be computationally expensive. Particularly if the data is not separable by a hyperplane.

VI. Refining Algorithms

Cosmin here…

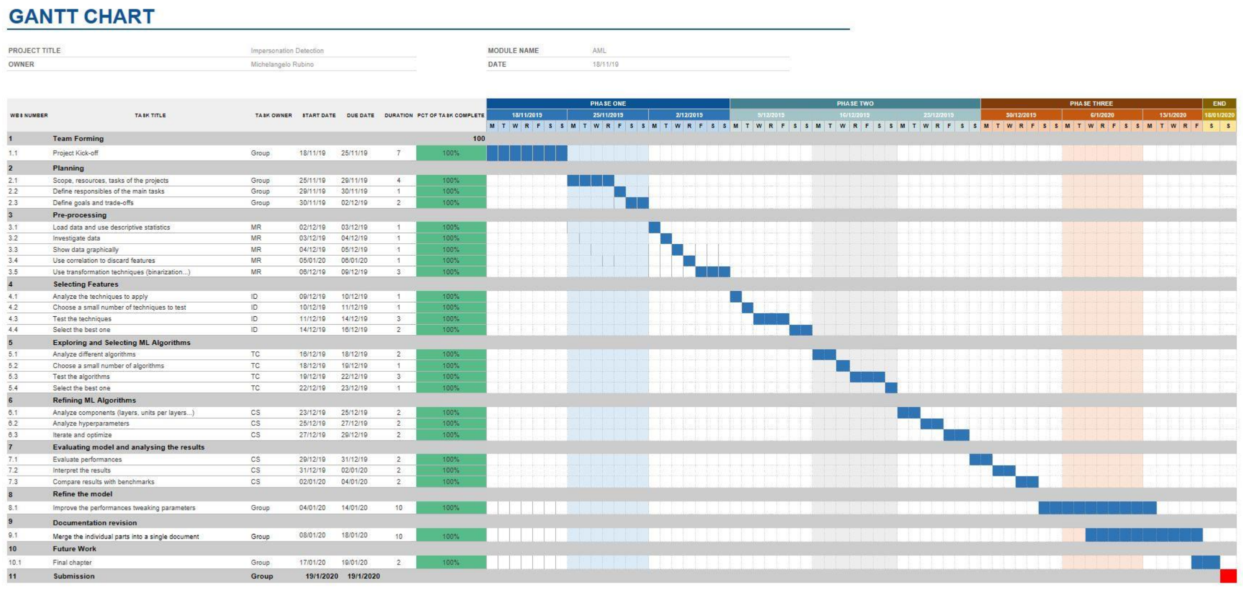
VII. Evaluating Model and Analysing the Results

Mike here…

VIII. Future Work

Data vs Model… more data means better accuracy?

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

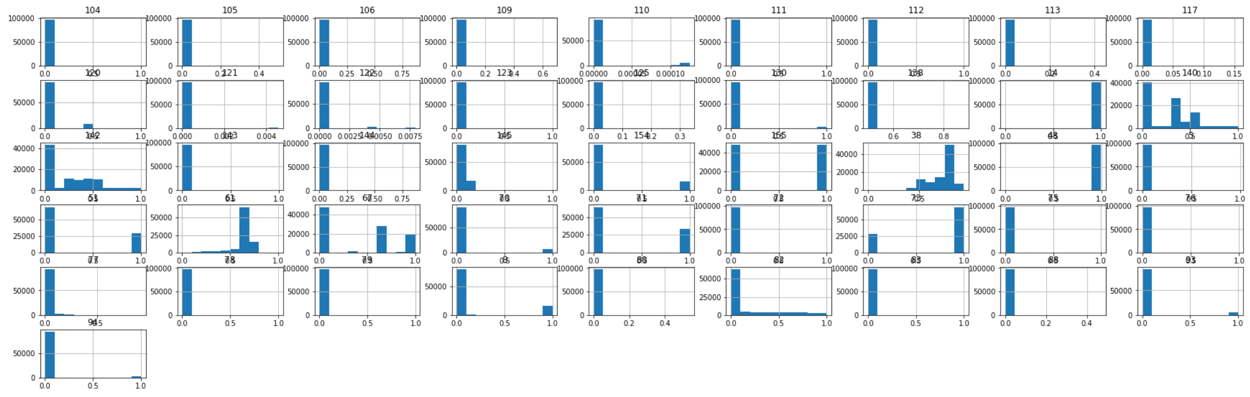


Fig. 2

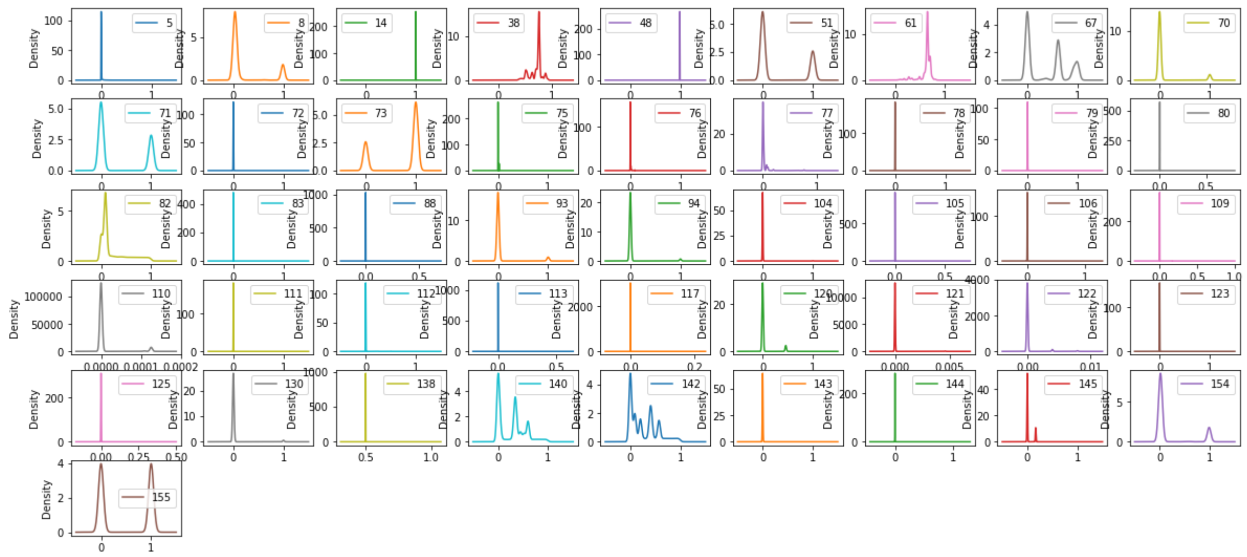

5
8
14
38
48
51
61
67
70
71
72
73
75
76
77
78
79
80
82
83
88
93
94
104
105
106
109
110
111
112
113
117
120
121
122
123
125
130
138
140
142
143
144
145
154
155
Density

Fig. 3

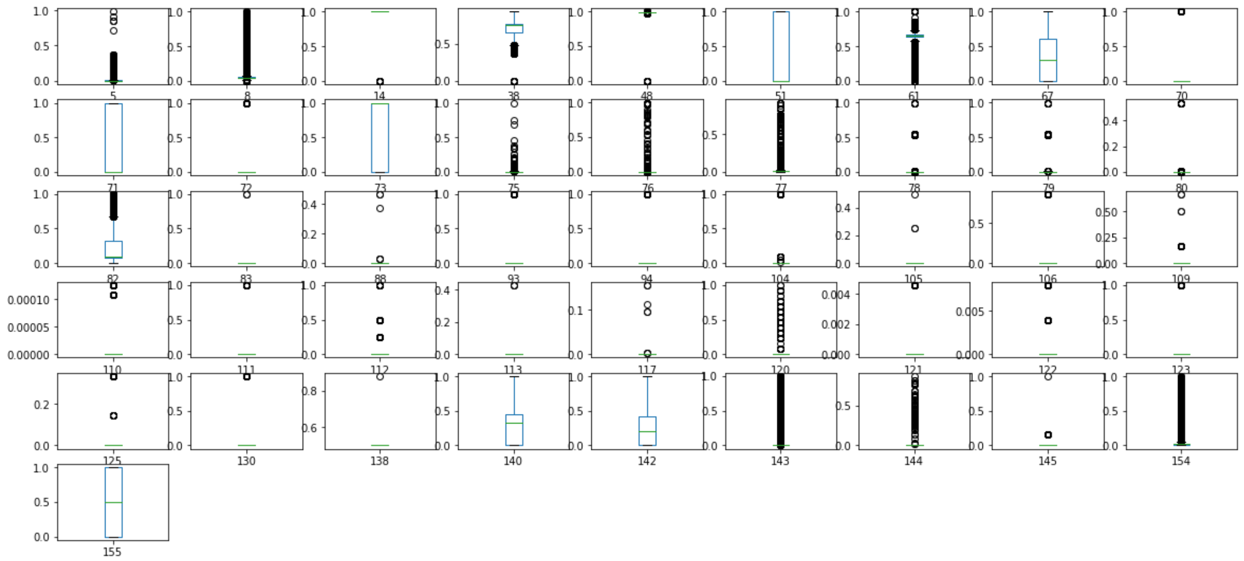

5
8
14
38
48
51
61
67
70
71
72
73
75
76
77
78
79
80
82
83
88
93
94
104
105
106
109
110
111
112
113
117
120
121
122
123
125
130
138
140
142
143
144
145
154
155

Fig. 4

X. Appendix B (Tables)

TABLE 1

Features sorted by Kendall correlation

79	0.725743
71	0.708561
50	0.652165
68	0.52784
5	0.517106
73	0.477183
80	0.452988
38	0.44858
66	0.420697
78	0.269324
140	0.24599
142	0.230346
14	0.014712
154	0.006752
111	0.005809
48	0.003032
97	-0.00321
138	-0.00321
83	-0.00454
105	-0.00556
113	-0.00556
84	-0.00786
88	-0.00786
108	-0.00877
86	-0.00908
117	-0.00908
119	-0.01171
120	-0.01314
64	-0.01967
123	-0.02487
144	-0.02872
106	-0.02873
72	-0.02955
98	-0.02997
90	-0.03385
125	-0.03921
118	-0.04226
93	-0.05423
109	-0.0579
104	-0.05973
112	-0.06228
121	-0.06841
143	-0.10554
141	-0.10583
61	-0.10681
8	-0.11996
128	-0.14825
127	-0.14836
126	-0.14836
129	-0.14836
130	-0.14836
94	-0.1682
122	-0.19312
107	-0.20429
82	-0.23428
110	-0.25602
70	-0.28538
77	-0.28997

145	-0.45347
75	-0.62075
47	-0.6505
51	-0.65183
76	-0.74737
67	-0.7832

TABLE 3

Features sorted by Spearman correlation

79	0.800862
71	0.708561
50	0.652165
5	0.632906
38	0.549358
68	0.539139
73	0.477183
80	0.463607
66	0.43509
140	0.285182
78	0.280235
142	0.267047
14	0.014712
154	0.007297
111	0.005809
48	0.003033
97	-0.00321
138	-0.00321
83	-0.00454
105	-0.00556
113	-0.00556
84	-0.00786
88	-0.00786
108	-0.00877
86	-0.00908
117	-0.00908
119	-0.01185
120	-0.01315
64	-0.02145
123	-0.02487
144	-0.02872
106	-0.02873
72	-0.02955
98	-0.02997
90	-0.03385
125	-0.03921
118	-0.04226
93	-0.05423
109	-0.0579
104	-0.05973
112	-0.06232
121	-0.06841
143	-0.10584
141	-0.10584
61	-0.12261
8	-0.13387
128	-0.14836
126	-0.14836
127	-0.14836
129	-0.14836
130	-0.14836
94	-0.1682

122	-0.19381
107	-0.20639
110	-0.25671
82	-0.27956
70	-0.28538
77	-0.31502
145	-0.45347
75	-0.64472
51	-0.65183
47	-0.6791
76	-0.80979
67	-0.83998

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

[13] 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]