IoT Intrusion Detection Competition using Machine Learning

Applied Machine Learning – Group Practical Project

Timothy Chan – *Birkbeck, University of London*, Ian Dickerson - *Birkbeck, University of London*, Mike Jun Ming - *Birkbeck, University of London*, Michelangelo Rubino - *Birkbeck, University of London* and Cosmin Stanciu - *Birkbeck, University of London*

***Abstract*—If not needed, remove it…**

***Index Terms*— AWID, IDS, Machine Learning, Neural Networks**

STRUCTURE

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

I. Overview;
II. Planning – Group;
III. Pre-processing – Michelangelo Rubino, No. 13140302 – 600 words;
IV. Selecting features – Ian Dickerson, No. 13140302 – 600 words;
V. Exploring and selecting ML algorithms - Timothy Chan, No. 13140302 – 600 words;
VI. Refining algorithms - Cosmin Stanciu, No. 13140302 – 600 words;
VII. Evaluating model and analysing the results - Mike Jun Ming, No. 13140302 – 600 words;
VIII. Future Work – Group;
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 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: 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 (IE. firewalls) have shown unsatisfying rates in detecting false positives and false negatives and in adapting to new threats.

II. PLANNING

This work uses a reduced version of the AWID dataset (maintained by the University of Aegean) 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] (EG. the attacks named as EvilTwin, Honeypot, Hirte). 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 waterfall methodology. The Machine Learning projects instead use an iterative lifecycle: several iterations to find the best setting. The idea in this project is then to test a certain number of algorithms and parameters for each stage to find the right model. Yet, due to the limited amount of time, around two months, it is not possible to test every machine learning technique. The focus is on those which are more frequently used for binary classifiers. The person accountable for each stage should agree the choices with the person accountable for the following stage, nevertheless the final decision is to be taken by the responsible for that stage.

Generally speaking, there are several challenges in this research area:

- known threats vs new threats (outliers),
- computational issues,
- accuracy vs speed,
- precision vs recall
- ML knowledge vs domain knowledge

To be continued…

III. Pre-processing

AWID is a public collection of standard data sets often used to evaluate the performances of an IDS, in particular for this project it will be used the reduced version AWID-CLS, comprising a training and a test set where two features have been removed as not useful for predictions. 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 feature as unbalanced datasets may lead to wrong predictions. The dataframe contains no nulls, then it is not necessary to handle them, anyway they would be identified as “NaN”. The describe() function shows the descriptive statistics and the first thing to notice is that many features have mean and standard deviation equal to zero: this means that the values are always zero and the features are not useful. 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

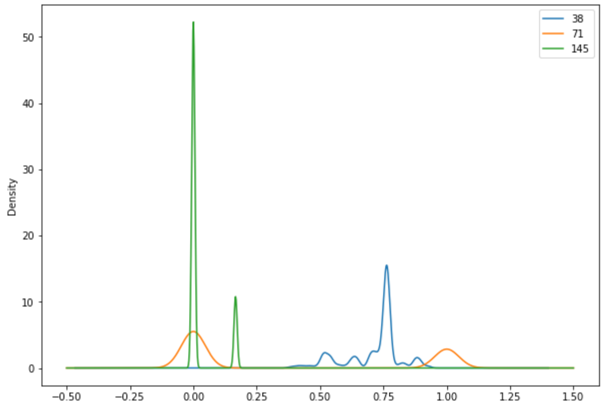
By combining the pandas dataframe functions loc() and duplicated() it is possible to detect duplicated features, having exactly the same values. These features can be removed as well, leading to a dataframe with only 64 features and the class. Decreasing the number of features is an important stage for 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 preprocessing in order to have a clean dataframe. The pandas describe() and nunique() functions 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).

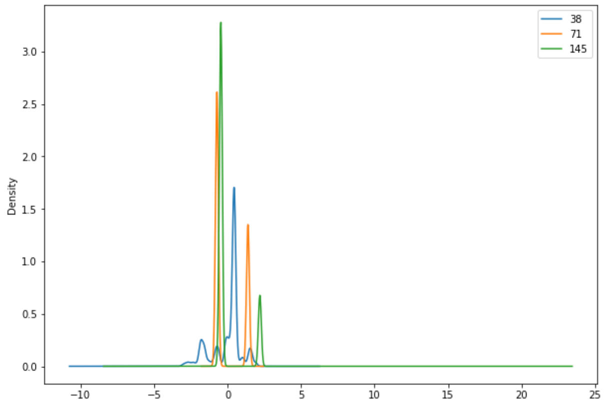
As every features show only values equal or higher than 0 and equal or lower than 1, the data does not need to be rescaled. This term refers to the techniques used in machine learning to give the features the same scale (typically between 0 and 1), which can give the model some improvements [4]. There is often confusion when using the terms scaling, standardizing and normalizing as authors tend to use them interchangeably, then the meanings explained in [5] will be used. The three abovementioned groups have different distributions and none has a normal distribution:



The density, the box and whisker plots and the histograms of every features can be seen in the Appendix A (Figg. 2-4). 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 [6] 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. Generally speaking, multicollinearity will never improve the results, on the contrary avoiding it is not dangerous and in some cases 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 pair. When dealing with dataframes showing a normal distribution, the Pearson correlation is typically used, 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 (commonly known as Kendall's tau coefficient) or the Spearman rank correlation coefficient (Spearman's rho). Differently of the most common correlation, 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 [6]. 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. To improve the distribution some techniques have been tried using the scikit-learn functions (Binarizer, PowerTransformer…), others have been discarded because not necessary (RobustScaler, MinMaxScaler) or because considered not useful (Normalizer as explained in [5]). It has been decided in the end to use only the standardization through the StandardScaler function, which removes the mean and scales to unit variance. This means that this transformation 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

Tim here…

VI. Refining algorithms

Cosmin here…

VII. Evaluating model and analysing the results

Mike here…

VIII. Future Work

IX. APPENDIX A (FIGURES)

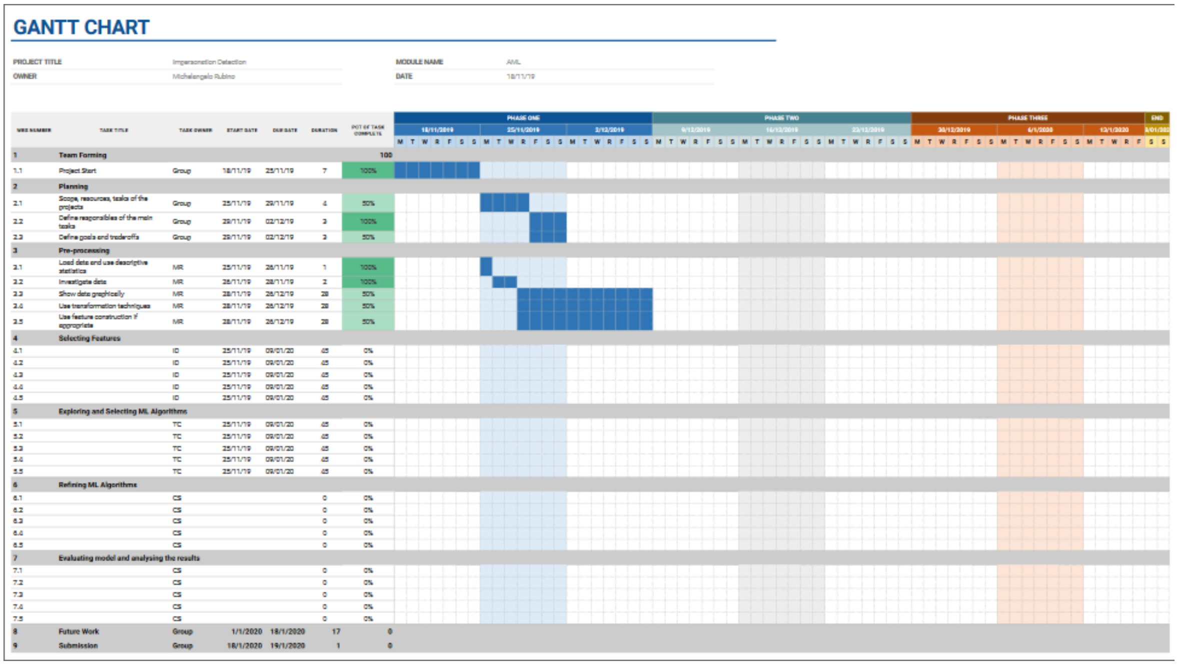


Fig. 1. GANTT chart

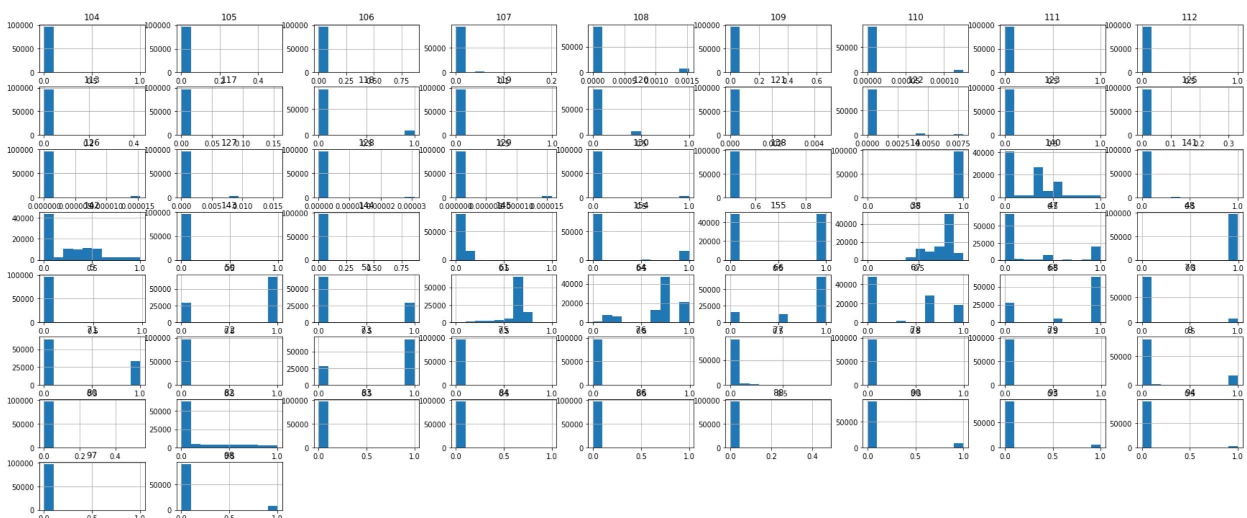


Fig. 2. Histograms for the df65

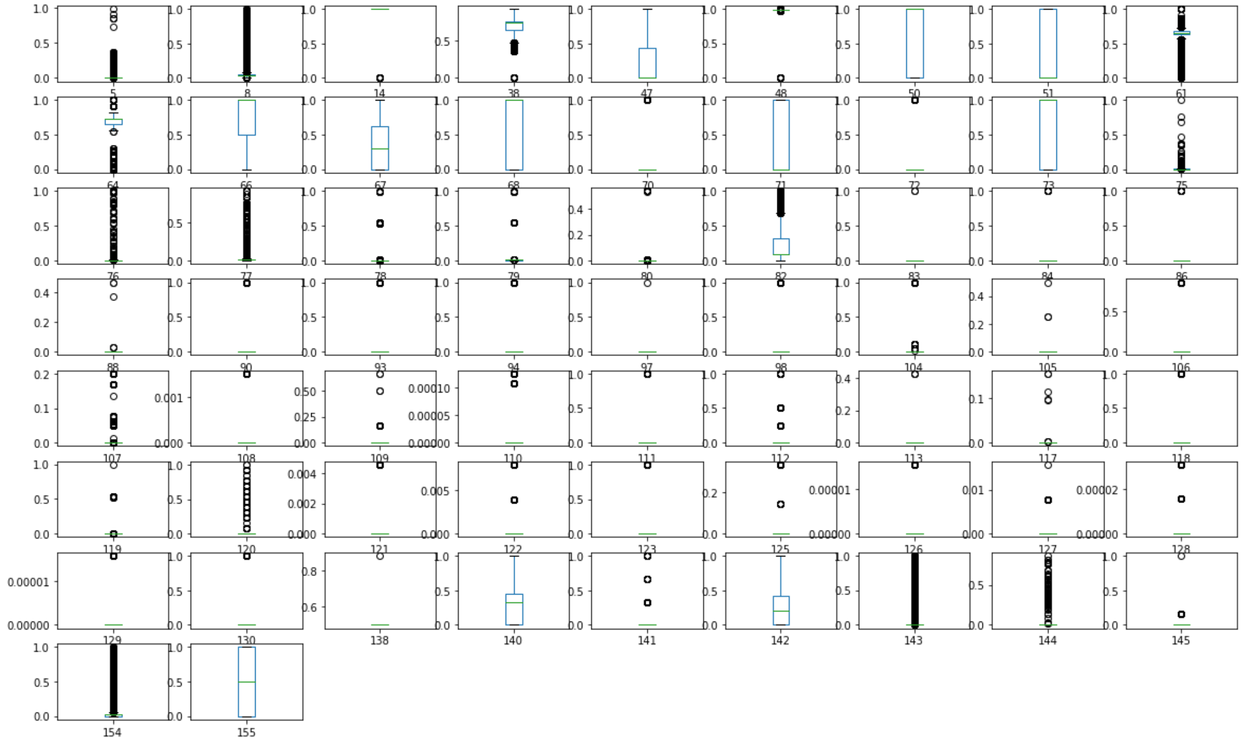


Fig. 3. Box and Whisker plot for the df65

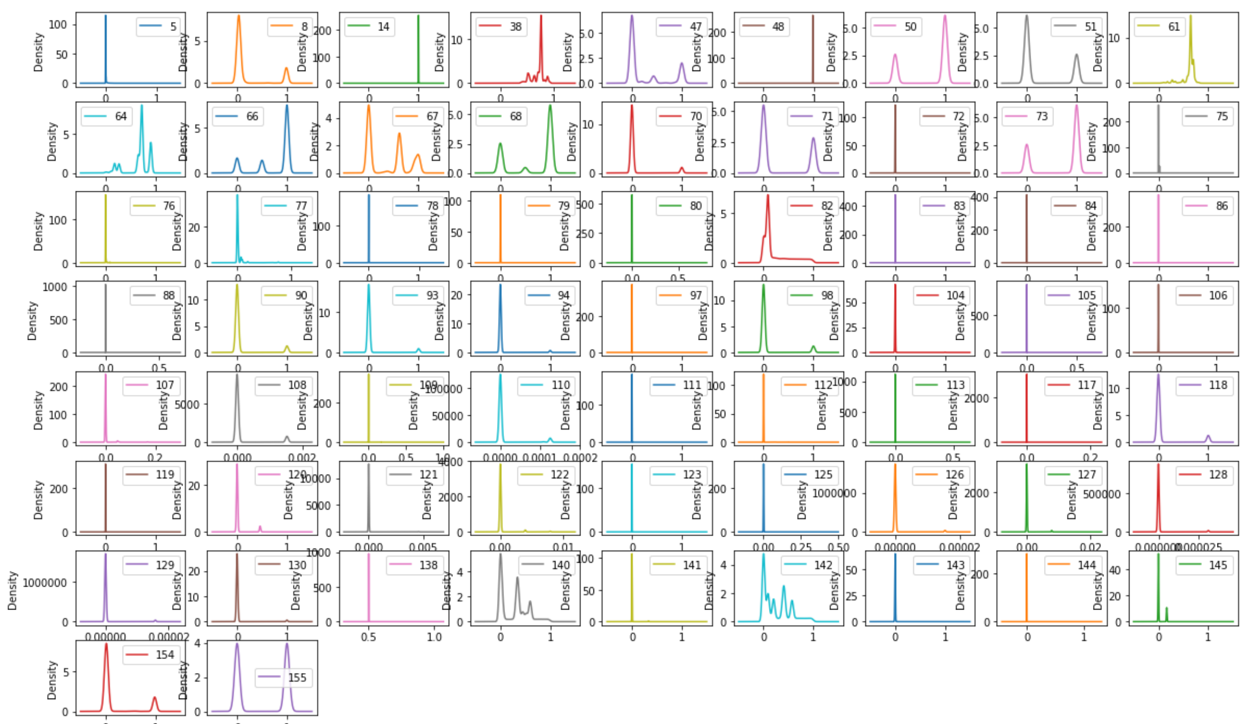


Fig. 4. Density plot for the df65

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] 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.

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

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

[7] 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]