Evaluation of Classification Machine Learning Models in Network Intrusion Detection Systems (NIDS)

By Davian Ricardo Chin

09 January 2022

The full code for this assignment can be found at: https://github.com/dave2k77/drc-code-space

---

TABLE OF CONTENTS

---

1. Introduction and Overview

In today's big data-driven world, massive amounts of data are generated every second and passed over massive, complicated networks. However, with this advancement comes more sophisticated and challenging risks of unauthorised intrusion and malicious attacks against network systems. As a result, there arises an urgent need for effective network intrusion detection systems (Al-Daweri, Khairul, Salwani & Mohamad, 2020).

Network intrusion detection systems analyse network traffic over local area networks (LAN) to detect abnormal activities such as unauthorised access, propagation of malicious code, unauthorised manipulation of privileges, traffic interception and denial of service (DoS) attacks. NIDS rely on two mechanisms: (i) misuse detection and (ii) anomaly detection. Misuse detection techniques are sophisticated techniques that use advanced mathematical and statistical methods to search and find patterns in network traffic data that are characteristic of types of illegal intent. Anomaly detection techniques use advanced statistical methods and heuristic measurements of system characteristics. Anomaly detection techniques try to predict whether a particular network activity is regular or not. NIDS has benefited from the rise in big data technologies and artificial intelligence, and machine learning techniques in the modern computing environment (Dini & Saponara, 2021).

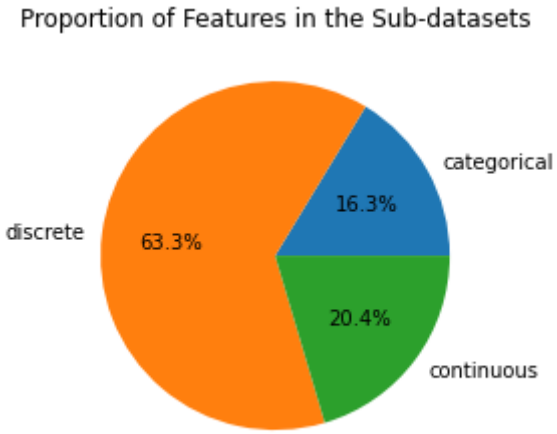
In this paper, we will focus on the anomaly detection aspect of NIDS, to determine whether we can derive a smaller subset of features from the original set from the UNSW-NB15 dataset that can predict the nature of the network activities on a LAN accurately. We propose that a smaller and more optimal set of features be selected through statistical processes to achieve an accuracy score of at least 95%. In addition, we predict that tree-based models will perform better for the predictive analytics we require.

This paper has five sections. The first section studies the dataset to understand its features by investigating the data types, basic statistics, and groupings or aggregations. Second, we will go through the ETLT process. Third, we scrutinise the data stored in each feature column and carry out data cleaning tasks such as removing duplication and trivial records, fixing poor formatting, and dealing with missing or uninformative data values. Third, we take a deep dive into some deeper statistical analysis to understand some summary statistics about features of the data set, uncover hidden relationships between features, and understand the data distribution in the features columns. Fourth, we use the insights from our statistical analysis to reduce the number of features before performing dimensionality reduction to produce five linearly independent features. The fifth and final stage is to train and test several machine learning models and compare their accuracy scores.

---

2. Deep Dive: Learning about the UNSW-NB15 Dataset and its Features

The UNSW-NB15 data is a popular dataset used for studying network intrusion detection systems. The dataset has 49 features with over 2.5 million records. There are 6 string-type (categorical), 10 floating-point (continuous) and 33 integer-type (discrete) features. On this basis, we divided the large dataset into three smaller datasets: "*categorical_features,*" "*discrete_features*" and "*continuous_features.*" The pie chart below shows the relative distribution of the features.



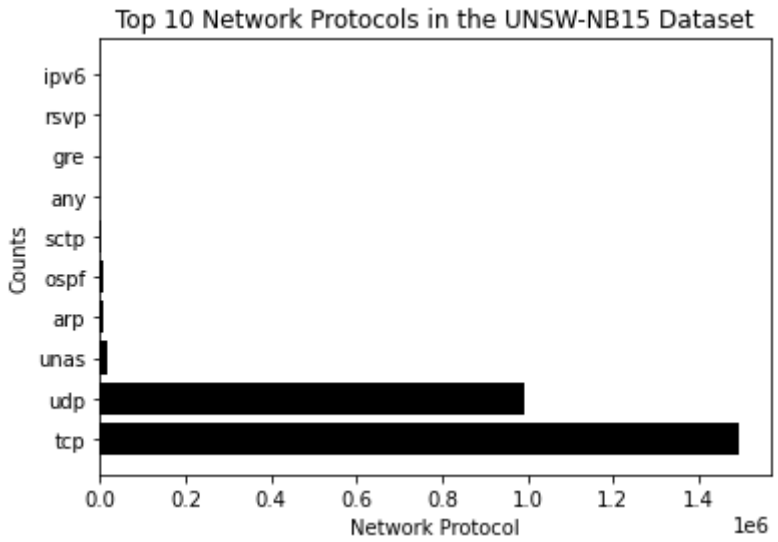
Let us start our deep dive with the categorical features. The "srcip" and "dstip" features store the IP addresses at the source and destination and points, respectively. Our analysis found 43 unique IP addresses at the source point and 47 at the destination point. The table below shows the top 5 most popular IP addresses at the source and destination, respectively.

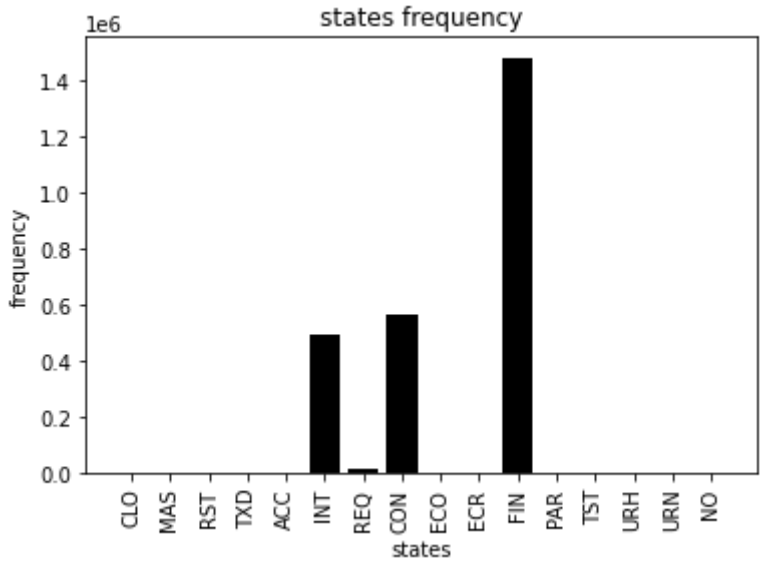
Source IP	Frequency	Destination IP	Frequency
59.166.0.4	197959	149.171.126.3	197771
59.166.0.1	197680	149.171.126.2	197648
59.166.0.5	197626	149.171.126.4	197639
59.166.0.2	197550	149.171.126.1	197525
59.166.0.0	197528	149.171.126.5	197000

The most popular source point IP address is 59.166.0.4, while the most popular destination point IP address is 149.171.126.3. We also found one instance of the loop-back IP 127.0.0.1. Therefore, we removed this trivial record.

The "proto" feature column lists network protocol used in transactions over the network, and the service feature column lists the network services used by network activity. A detailed inspection of the service column revealed several rows with the string "-" recorded in the service feature column. After the data cleaning process, the "-" strings were replaced by "unused". As a result, we found that 1246089 rows of the dataset contain the string "-" in the service feature column representing approximately 49% of the dataset.

We found that there are 134 unique protocols and 13 unique services listed. The most popular network protocol used was TCP with a frequency of 1495074, and most of the activities recorded used a service that is not commonly used ("-") with a frequency of 1246089. The least used protocol was RTP (7), and the least used service was IRC (31). The bar chart s below shows the relative popularity of the top 10 network protocols and the states used.





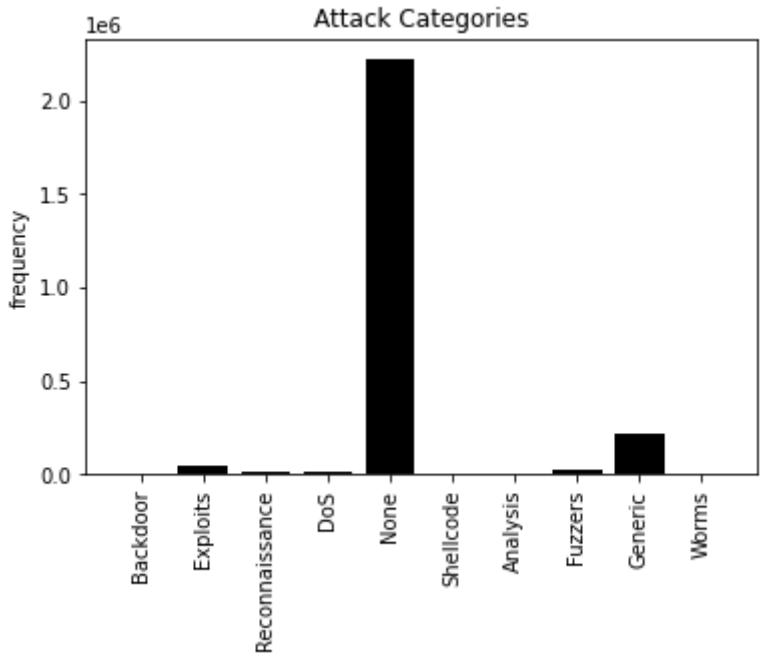
The "*attack_cat*" column is one of our label columns used to develop our machine learning models. However, if we look at the number of distinct values here, we find something surprising; we have several duplications due to poor formatting, blank values, and inconsistencies in spelling.

Attack Category	**Label**	**Frequency**
	0	2218456
Backdoor	1	1795
Exploits	1	44525
Reconnaissance	1	12228
Backdoors	1	534
DoS	1	16353
Shellcode	1	1288
Analysis	1	2677
Fuzzers	1	5051
Fuzzers	1	19195
Generic	1	215481
Reconnaissance	1	1759
Shellcode	1	223
Worms	1	174

At the top is a row of blank values that account for a large amount of the data. We found out that there is a one-to-one association between the blank *"attack_cat"* values and a Label of 0 and between attack categories and a Label of 1, suggesting that the blank values are the attack category value for regular activities on the network. Therefore, we replaced all the blank values with the more informative value "None" during the cleaning process. The table below shows the cleaned *"attack_cat"* column.

Attack Category	Frequency
None	2218456
Backdoor	2329
DoS	16353
Exploits	44525
Fuzzers	24246
Generic	215481
Analysis	2677
Reconnaissance	13987
Shellcode	1511
Worms	174

The bar chart below gives a visual perspective of our observations.



In terms of the attack categories, generic attacks are far more common than all others, followed by exploits and fuzzers.

Let us now look at our *"discrete features*". Let us start with the *"stime"* and *"ltime"* columns with time data. Next, we create a new feature called *"duration"* ("ltime" - "stime") to use as an aggregated feature for both features. The table below shows the top 5 records of our newly created feature.

Record Start Time	Record End Time	Duration
1421970774	1421970775	1
1421970775	1421970775	0
1421970775	1421970775	0
1421970772	1421970775	3
1421970773	1421970775	2

There are 2008635 records where the start and end times are the same, giving a total duration of 0, representing approximately 79% of the dataset. The table below shows some additional statistics of the “duration” feature.

Max. Duration	Min. Duration	Mean Duration	Variance
8896	0	0.7640690637896256	566.9555975929928

Next, we aggregate the `sbytes`, `dbytes`, `sload` and `dload` features using the following formulas:

$$\text{Let S} = \text{sbytes and D} = \text{dbytes. We can then define the following new features.}$$

$$dS = \frac{S}{T_s}\ (\text{source transmission rate})$$

$$dD = \frac{D}{T_d}\ (\text{destination transmission rate})$$

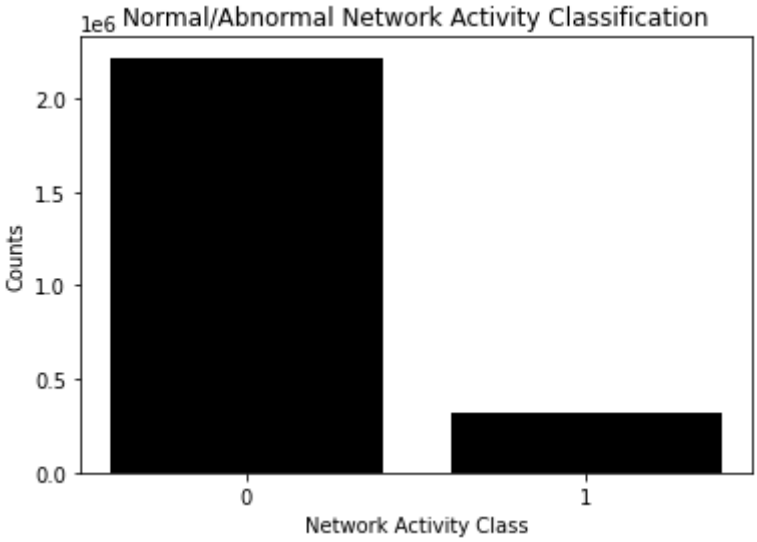
The table below shows the top 5 values for `STRate` and `DTRate` features.

S (sbytes)	D (dbytes)	STTL (Ts)	DTTL (Td)	STRate (dS)	DTRate (dD)
7812	16236	31	29	252.0	559.8620689655172
4238	65628	31	29	136.70967741935485	2263.0344827586205
2750	29104	31	29	88.7097741935483	1003.5862068965517
12476	395734	31	29	337.93548387096774	13646.0
13350	548216	31	29	430.64516129032256	18904.0

Looking at the label column, we can generate a frequency distribution shown below.

label	counts
0	2218455
1	321283

The bar chart below shows a graphical distribution of the "label" column. Most of the activities observed on the network are normal activities. Normal activities account for about 87% of the dataset.



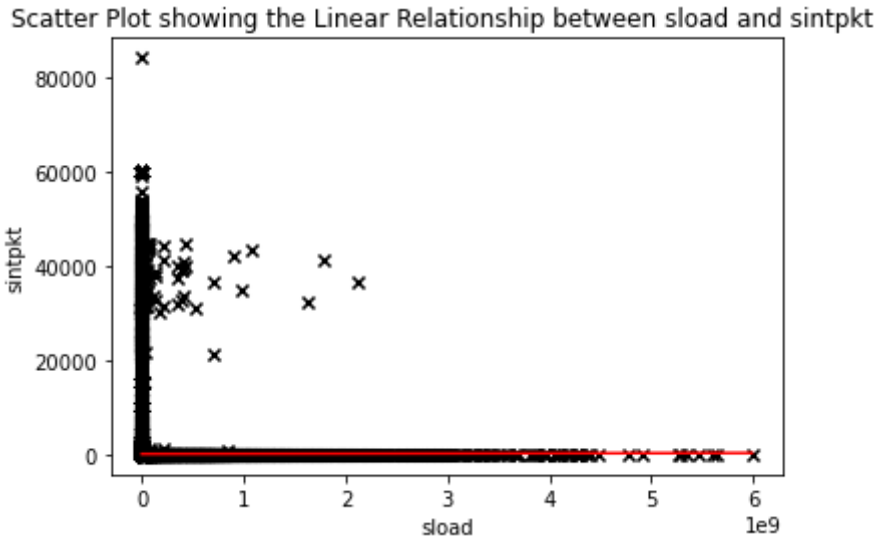
Lastly, we look at some of the continuous features. First, let us look at the average inter-packet arrival times at the source and destination points.

Average sources inter-packet arrival time (sintpkt)	**Average destination inter-packet arrival time (dintpkt)**	**Standard Deviation (sintpkt)**	**Standard Deviation (dintpkt)**
193.04662647027246	78.83435459342978	2778.582999823837	1433.27807008653

The result shows that inter-arrival times at the source are twice if that at the destination. It may also be interesting to see if there is any correlation between the inter-packet arrival times and the bit rate at the source and destination:

sintpkt-sload Correlation	**dintpkt-dload Correlation**
-0.020239609022803283	-0.03125328769397401

The correlation between the inter-packet arrival times and bit rates is weak and negative, suggesting no solid linear relationship exists between the two features. The scatter plot below summarises our observations.



The scatter plot shows a line of best fit (red line) that is flat, indicating the weak linear association between “sload" and “sintpkt", consistent with the near-zero correlation value obtained above. The relationship between "dintpkt” and "dload" showed a similar result.

Finally, we look at the "tcprtt", "synack" and "ackdat" features. The dataset documentation shows that "tcprtt" is an aggregation of "synack" and "ackdat". Therefore, we can look at the correlation among the features to get more details.

tcprtt - synack Correlation	tcprtt-ackdat Correlation	synack-ackdat Correlation
0.9313053020510753	0.9188987920093781	0.7120852419489705

We find a strong correlation between "synack" and "ackdat" with "tcprtt". "synack" and "ackdat" are moderately correlated. Therefore, including "tcprtt" in the final dataset features would be equivalent to adding "synack" and "ackdat".

After the data cleaning process, we arrive at a new set of features.

Feature ID	Feature Name	Feature ID	Feature Name	Feature ID	Feature Name
F1	srcip	F16	swin	F31	ct_state_ttl
F2	sport	F17	dwin	F32	ct_flw_http_mthd
F3	dstip	F18	stcpb	F33	is_ftp_login
F4	dsport	F19	dtcpb	F34	ct_ftp_cmd
F5	proto	F20	smeansz	F35	ct_srv_src
F6	state	F21	dmeansz	F36	ct_srv_dst
F7	strate	F22	trans_depth	F37	ct_dts_ltm
F8	dtrate	F23	res_bdy_len	F38	ct_src_ltm
F9	sloss	F24	sjit	F39	ct_srv_src
F10	dloss	F25	djit	F40	ct_src_dport
F11	service	F26	duration	F41	ct_dst_sport
F12	sload	F27	sintpkt	F42	ct_dst_src_ltm
F13	dload	F28	dintpkt	F43	attack_cat
F14	spkts	F29	tcprtt	F44	label
F15	dpkts	F30	is_sm_ips_ports		

3. Advanced Analytics and Statistics on the UNSW-NB15 Dataset

3.1 Advance Statistical Analysis of the UNSW-NB15 Dataset

Observing the dataset's schema makes it clear that we can sub-divide the whole dataset into sub-datasets based on the data types of the columns, as shown below.

Categorical (string-type)	Discrete (integer-type)	Continuous (floating-type)
F1, F3, F5, F6, F11, F43	F2, F4, F9, F10, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F26, F30, F31, F32, F33, F34, F35, F36, F37, F38, F39, F40, F41, F42, F44	F7, F8, F12, F13, F27, F28, F29

Dividing the dataset by data types have the advantage of simplifying the analytics processes by reducing the overheads of selecting similar feature types and applying statistical estimators and transformers (in many cases, require all the data operated on to be of the same data type, e.g., categorical data for Chi-square Testing) to the data. It also simplifies planning for data visualisation.

3.2 Pearson's Chi-Square-Test for Independence

We used the Pearson's Chi-Square Test for independence to determine the statistical association or dependence of the categorical variables relative to the target variable. The test measures the degree to which the data points cluster around a straight line. The p-value associated with a Chi-square test proposes a null hypothesis that maintains that the degree to which the features and the label are correlated is due to random phenomena. In contrast, the alternate hypothesis proposes that the association between the feature and the label is unlikely due to random phenomena but to statistically significant factors.

We will use the categorical features data to test against the "attack_cat_index" target variable. See the results below.

	F1	F3	F5	F6	F11	F43
Degrees of Freedom	378	414	1197	135	9	81
p-value	0.0	0.0	0.0	0.0	0.0	0.0
Test Statistic	2513377.4825	3347042.4671	1527033.6743	1063339.3489	242205.7156	22857651.0

The result suggests rejecting our null hypothesis and accepting the alternate hypothesis; in other words, there is not enough evidence to support the position that the observed linear association between each feature and the label is due to random factors. On this basis, we maintain that it is statistically sound to keep all the feature columns.

3.3 Distinct Categories Count

Some categorical features have many distinct groupings, which does not work well with machine learning algorithms. Therefore, we set a threshold of a maximum of 32 distinct groupings. The table below shows the number of distinct groupings in each categorical feature column and the action taken.

Feature	Number of Distinct Groupings	Action
F1	43	Drop
F3	47	Drop
F5	134	Drop
F6	16	Keep
F11	2	Keep
F43	10	Keep

3.4 Correlation Analysis

Correlation analysis computes Pearson's correlation coefficient for pairwise features, taking values in the interval [-1, 1]. First, we computed the correlation matrix and filtered out the label column that shows the correlation values of all the features with the label.

We then filtered the resulting dataset to remove all features with correlation in the interval (-0.15, 0.15), that is, all features close to 0. Correlation values in the intervals [-0.70, -0.15] and [0.15, 0.70] are deemed as moderate so we keep these values.

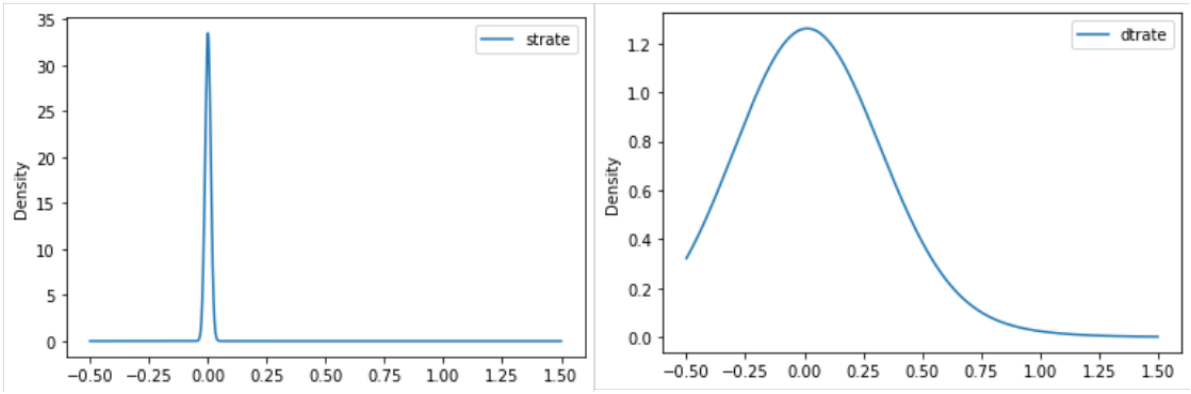
Feature	F44
F31	0.873695
F36	0.38299
F37	0.386492
F38	0.339446
F39	0.343228
F40	0.396737
F41	0419045
F42	0.439904
F44	1

3.5 Descriptive Analytics and Kernel Density Estimation

We now look at the continuous features, observing some basic statistics about the features.

Summary	F7	F8	F12	F13	F27	F28	F29
count	2529291	2033969	2539738	2539738	2539738	2539738	2539738
mean	123.1886	1546.7618	3.6949E+7	2451159.4030	193.0466	78.8344	0.00618
standard deviation	740.4791	5973.0297	1.1860E+8	4225033.4492	2778.5835	1433.2784	0.0462
minimum	0.0	0.3413	0.0	0.0	0.0	0.0	0.0
maximum	187599.6935	58164.8056	5.9880003E+9	1.2876E+8	84371.49	59485.32	10.0375

F7 and F8, representing "strate" and "dtrate", are two aggregate features from the original dataset representing the bytes rate at the source and destination, respectively. The statistical summary table shows that the "dtrate" feature's standard deviation is much larger than the "strate" feature, indicating that the "dtrate" feature has a higher variance than the "strate" feature. The density plot of the "strate" and "dtrate" features shows the observations about the "strate" and "dtrate" features.



The "dtrate" feature has a distribution that is closer to being normally distributed than the "strate" feature. The density plots show that the "strate" has a low variance. Low variance tends to introduce bias (errors) into models, while high variance introduces complexity (low consistency). According to Sharda, Delen and Turban (2020), for analytics, the best models show have low bias and low variance; however, this is an ideal situation rather than the norm. There will always be a trade-off. A large amount of training data will always help to reduce the impact of high variance, so we will, on this basis, drop the "strate" and "tcprtt" features with low variance in favour of admitting a high variance.

The analysis done so far has reduced the features in each sub-dataset. We have the following features remaining.

Feature ID	Feature Name
F6	state
F8	drate
F11	service
F12	sload
F13	dload
F27	sintpkt
F28	dintpkt
F31	ct_state_ttl
F36	ct_srv_dst
F37	ct_dts_ltm
F38	ct_src_ltm
F39	ct_srv_src
F40	ct_src_dport
F41	ct_dst_sport
F42	ct_dst_src_ltm
F43	attack_cat
F44	label

---

4. Data Preparation and Feature Selection for the UNSW-NB15 Dataset

4.1. Feature Selection techniques

Feature selection is the process of selecting the most relevant features of a dataset to predict or classify the inevitable consequences of creating the data. In deciding which features to include for training and testing our prediction model, we consider the "relevance" of the features to the decision process. We consider a feature to be strongly relevant if removing this feature from the dataset causes the predictive model to perform more poorly than when present in the dataset. A feature is weakly relevant if, on its own, its removal does not reduce the performance of the predictive model, but the removal of a finite subset of features containing that feature reduces the performance of the predictive model. Therefore, a feature is irrelevant if it is neither strongly nor weakly relevant (Mahfouz et al., 2020).

In practice, we carry out feature selection in three ways:

1. Using prior knowledge and direct inspection of the dataset features to remove uncorrelated features to the label or uncorrelated to other features.
2. Using the filtering method includes statistical tests to evaluate feature relevance.
3. Using formal methods where a set of relevant features are selected using an inductive algorithm.

(Mahfouz et al., 2020)

In this experiment, we used methods 1 and 2 to determine the relevance of the features of the UNSW-NB15 dataset. We used the definitions of strong and weak relevance stated above to select a set of features with the help of *Pearson's Chi-Square Test for Independence* and *Correlation Analysis*. Additionally, we dropped features with internal groupings greater than 32 because they caused computational errors when trying to use them in training. For instance, the "proto" feature had 134 distinct groupings, which caused errors in training as the algorithms required that the number of groupings not exceed a maximum value of 32.

Method 3 is more effective at feature selection but can be time-consuming and impractical for large datasets (Mahfouz et al., 2020). We did not use this method for this reason.

4.1. Data Normalisation

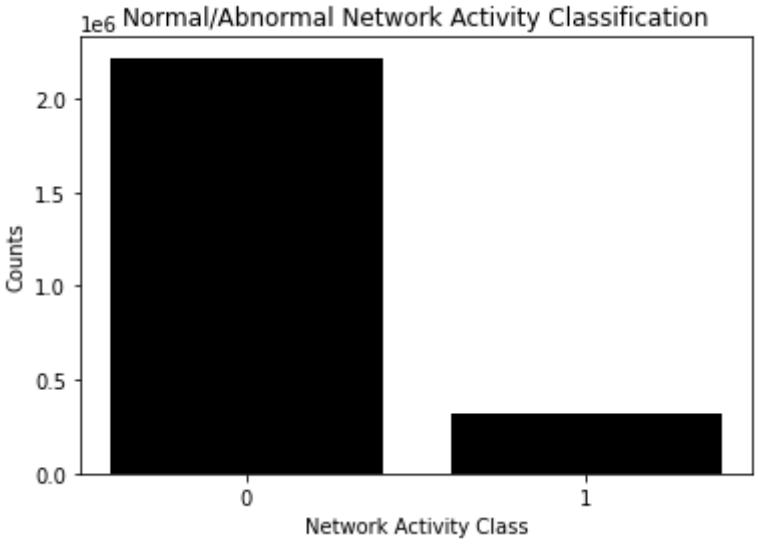
Data normalisation (or standardisation) aims to map data values to a value in the range [0, 1] without changing its statistical distribution. Normalisation helps to ensure that those features are compared using the measurement units and on the same basis. This can help to ensure that the results obtained from models are proportionate and without bias. In our experiment, the *Standard Scaler* was used to scale the data values with data values centred around the mean and variance set to 1. A *unit-vector* based scaler defined by:

$$\hat{x} = \frac{x}{|x|}, \text{for } x \neq 0.$$

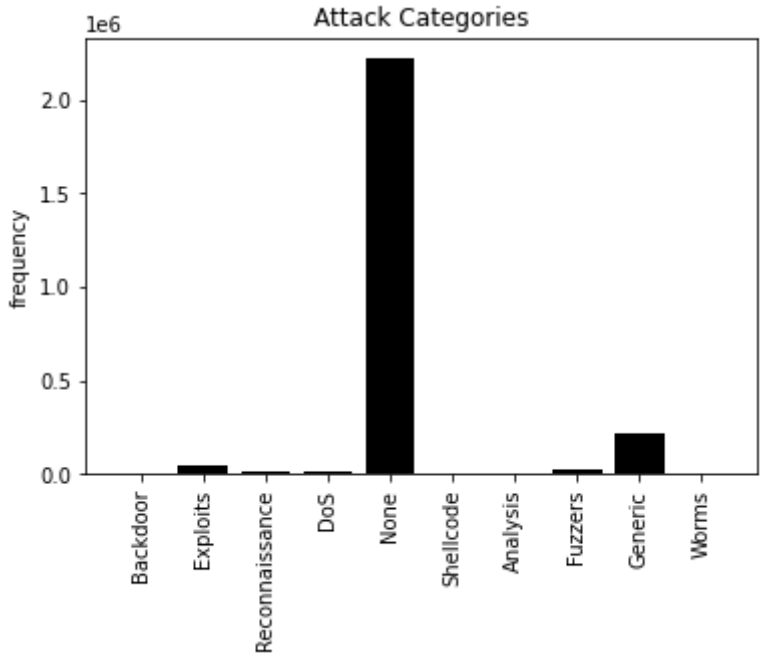
was also used to normalise features during the analytics process.

4.2. Addressing Dataset Imbalance

An imbalanced dataset is defined as a dataset where the number of examples across the dataset is not equally distributed. For example, the UNSW-NB15 dataset is imbalance when you consider the proportion of normal activities and attack incidents shown below.



This imbalance will have dire consequences for machine learning algorithms as most are designed to assume equal numbers of examples per class label. This problem will be further exaggerated if individual cases are comparing the normal activities, as those numbers will be even lower as seen below.



According to Mahfouz et al. (2020), random sample can be an effective technique to counter imbalance in the dataset be excluding majority of the values from the dominant class label.

In this experiment, we chose to do an under-sample where the number of examples in the class label of 0 was reduced to match the number of examples in the class label of 1. We defined a function *under_sampling_function()* that took a dataframe as an argument and returned a new dataframe in which each class label has an equal number of examples.

4.3. Dimensionality Reduction

Our final task is to use a dimensionality reduction technique to reduce the number of features in the dataset without losing generality. We used the Principal Component Analysis (PCA) to do this. The PCA estimator technique takes *n* features of a dataset and reduces them to *m* linearly independent features where $n > m$.

---

5. Machine Learning Models for Predictive Analytics

5.1 The Computing Environment

Hardware Model	Lenovo ThinkPad E480
Memory	31.3 GiB
Processor	Intel® Core™ i5-8250U CPU @ 1.60GHz × 8
Graphics	Mesa Intel® UHD Graphics 620 (KBL GT2)
Disk Capacity	1.3 TB

OS Name	Ubuntu 21.10
OS Type	64-bit

The figure above describes the technical specifications of the computer used to carry out the experiments for this paper. The computer also runs Hadoop 3.3.1 on YARN for big data storage and processing, and Hive with MySQL metastore to query and pre-process the data stored in HDFS.

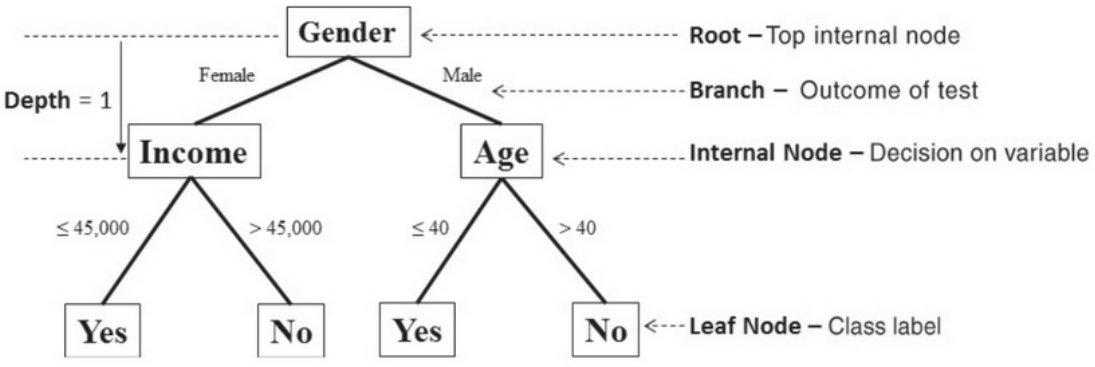
We used Spyder-IDE 4.2.1 (Scientific Python Development Environment), Python 3.9.7 and IPython 7.20.0 to do all the coding work involved in developing, training, testing, and evaluating the machine learning models. In addition, we used Pyspark ML/MLLIB framework and Python to develop the machine learning models and visualisations.

5.2 Overview of Classification Machine Learning Models

5.2.1 Decision Tree Classifier

A decision tree is a computational model of decisions and consequences. It uses a tree-like structure to model the relationship between decisions made and the consequences that come as a result.

The figure below shows an example of a decision tree used to model customers' decision to buy or not to buy a product.



(Adapted from EMC Educational Services (2015, p. 193))

A tree node represents decision points where a binary choice of "Yes" or "No" must be made. "Yes" and "No" are values that we are interested in predicting for a given set of inputs for "Gender", "Income", and "Age". In machine learning, these inputs represent *features*, and the values we want to predict are called *labels*. Therefore, this problem is a *Binary Classification* problem as the label consists of only two values.

The general algorithm for a decision tree attempts to create a tree T from a set of training data S. The tree T has the following attributes:

- Tree depth (integer-type): the minimum number of steps required to reach an internal node starting from the root node.
- Impurity (entropy, Gini): measures the impurity of a feature. For entropy, this parameter takes values True or False.

Decision tree classifiers are computationally efficient and perform well on categorical and numerical data that may be redundant or highly correlated; however, they are susceptible to irrelevant features, so care must be taken to remove these features before training decision tree classifiers.

5.2.2 Naive Bayes Classifier

The "Naive Bayes Classifier" is a probabilistic classifier that works based on *Bayes Theorem*:

$$P(c_i|A) = \frac{P(a_1, a_2, \ldots, a_m|c_i) \cdot P(c_i)}{P(a_1, a_2, \ldots, a_m)}, i = 1, 2, \ldots n$$

(Adapted from EMC Educational Services (2015, p. 214))

The *Naive Bayes* classifier works with categorical data, but we can tune its parameters to allow it to work with continuous data through discretisation; that is, data transformation converting numerical data into categorical data (Sharda, Delen and Turban, 2020). Furthermore, the Naive Bayes classifier assumes the independence of the features in predicting the class label of data. As a result, it works well against missing and irrelevant values in the dataset, handles high-dimensional datasets very well and is more resistant to over-fitting due to its adjustable smoothing parameter (EMC Educational Services, 2015). It is, however, susceptible to highly correlated values due to its fundamental assumption that the features are statistically independent (EMC Educational Services, 2015).

The general *Naive Bayes* algorithm work in this way: in observing new data, it associates the new data with class labels containing sample data most like the new data and then assigns the new data to the most representative class label (Sharda, Delen and Turban, 2020).

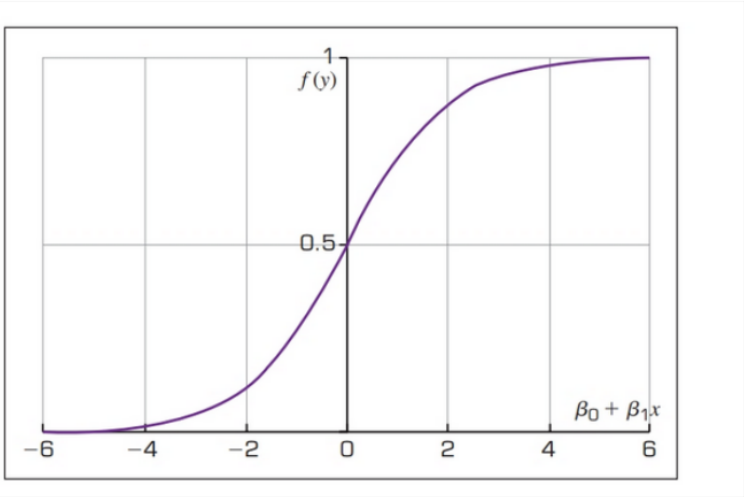
The Naive Bayes classifier has the following attributes:

- smoothing (float-type): value in [0, 1]

5.2.3 Logistic Regression

Logistic regression models are commonly used to develop probabilistic models for binary and multiclass classification problems. It is a probabilistic model to explain the relationship between the dataset's features and the class label. We describe the mathematical formulation for a logistic regression problem below.

$$f(y) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$



(Adapted from EMC Educational Services (2015, p. 192))

The Logistic Regression classifier has the following parameters:

- regParam (float-type): values greater than or equal to 0
- elasticNetParam (float-type): values in [0, 1]
- maxIter (intger-type): integers greater than or equal to 0.

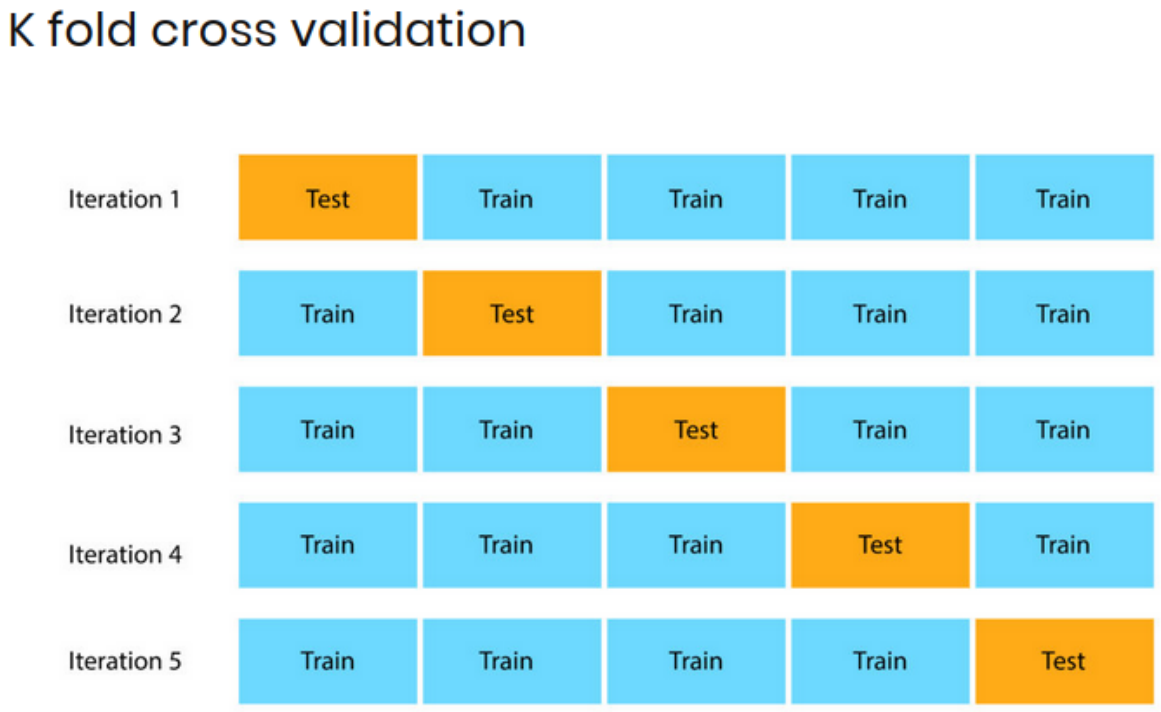
5.2.4 Ensemble Models

Ensemble models combine the strengths of multiple models to form a "super" model with much higher performance and thus increase its accuracy rate. They combine the outcomes of two or more analytical models into a single output that is usually more accurate and reliable than the individual models involved (Sharda, Delen and Turban, 2020). According to Sharda, Delen and Turban (2020, p. 331), ensemble techniques "can also improve model robustness, stability, and, hence, reliability". Bagging, boosting, and stacking are the three main frameworks for ensemble models (Mahfouz et al., 2020). Examples of ensemble models are *Random Forests* and *Gradient-Boosted Trees*.

5.2.5 Hyperparameter Optimisation and Cross-Validation

Hyperparameter optimisation is the process of selecting the "best" (most optimal) external parameter configuration for a model. The process involves testing and selecting the most optimal combination of parameters that will reduce some predefined loss function for the dataset. This process usually forms part of the cross-validation process. The hyperparameter optimisation technique is usually a grid search, random search, or Bayesian Optimisation algorithm (Mahfouz et al., 2020).

Cross-validation is a commonly used technique to divide a data sample into several disjointed subsets. Then, we iteratively train, test and evaluate the subsets against each other before computing an average score. Cross-validation usually includes Hyperparameter optimisation. The diagram below gives a graphical illustration of the cross-validation process.



Adapted from Great Learning (2020)

Cross-validation reduces the possibility of bias and over-fitting in predictive models, making them more reliable. However, one main disadvantage is that it can be computationally expensive.

5.3 Model Selection and Evaluation

5.3.1 Selecting a machine learning model

In choosing the models we used in our experiment, we considered what we know of the dataset through our analytics and descriptive statistics efforts and some background knowledge of specific machine learning models. To summarise a few details, the correlation matrix revealed that the data is moderately correlated, with mixed variable types. Furthermore, the dataset is high dimensional. On this basis, the EMC Educational Services (2015, p. 229) recommends the Naive Bayes, Logistic Regression and Decision Tree models. Additionally, ensemble models tend to work well where tree models are appropriate and may boost performance.

5.4 Classification Results

5.4.1 Binary Classification

Binary Classifier	Test Accuracy Score (%)	C.V. Accuracy Score (%)	Test Precision Score (%)	C.V. Precision Score (%)
Naive Bayes Classifier	99.37	98.81	99.37	98.62
Logistic Regression Classifier	98.82	99.37	98.62	99.37
Gradient Boosted Tree Classifier	99.70	99.74	99.44	99.48

The results show that the *Gradient Boosted Tree Classifier* performed the best of the three classifiers trained and test. The *Naive Bayes Classifier* performed the worst.

5.4.2 Multiclass Classification

Multiclass Classifier	Test Wgt Precision Score (%)	C.V. Wgt Precision Score (%)	Test F1-Score (%)	C.V. F1-Score (%)	Test Accuracy Score (%)	C.V. Accuracy Score (%)
Naive Bayes Classifier	93.83	93.83	92.98	92.98	93.95	93.95
Decision Tree Classifier	97.45	99.05	97.31	99.04	97.40	99.06
Random Forest Classifier	97.10	99.18	97.08	99.16	97.28	99.17

The results from the multiclass classification experiments show that the *Decision Classifier* performed the best of the three classifiers tested. Again, the *Naive Bayes Classifier* performed the worst.

---

6. Alternative Technologies for Big Data Storage and Analytics

In this paper, we used Hadoop Distributed File system with MapReduce for storing the UNSW-NB15 big data file, Hive to complete data mining, warehousing and pre-processing tasks, and Spark and Python for advanced data analytics and visualisation. These were excellent and essential tools that provided the capabilities to work with the big UNSW-NB15 data. However, these are not the only tools available for big data analytics. This section discusses some popular alternatives to the tools used in this project.

NoSQL databases such as Mongo and Cassandra databases are known to be "highly scalable, fault-tolerant and specifically designed to house semi-structured and unstructured data" (Erl, Khattak & Buhler, 2015, p. 94). Mongo is very efficient at storing a "wide variety of data at high volumes" (EDUCBA, n.d.). Cassandra features a distributed architecture that is highly scalable, fault-tolerant, schema-less, eventual consistency and others (Erl, Khattak & Buhler, 2015) and has built-in support for MapReduce (EDUCBA, n.d.).

Apache Kafka and Splunk are popular data analytics tools used by large companies like LinkedIn, Yahoo and Spotify. Kafka can ingest and perform analytics tasks on real-time streaming data. Splunk can correlate and index real-time streaming data from searchable repositories and generate visualisations, dashboards, and reports (EDUCBA, n.d.).

Regarding data mining and warehousing, Rapid Miner, Orange, Mahout, KNIME, and Teradata provide predictive analytics, business intelligence, machine learning, database connectivity, and SQL support in processing big data. Alpine Miner is a popular choice for creating analytic workflows for data mining jobs and clustering and descriptive statistics on big data, while Open

Refine and Data Wrangler are open-source examples that provide an interactive GUI for cleaning and transforming messy data (EMC Educational Services, 2015). Pharmaceutical research and financial companies use KNIME for working with big data. Elasticsearch and Presto are commercial examples used by large organisations like Netflix, Facebook and Accenture, and Rapid Miner is popular in academic institutions (EDUCBA, n.d.).

On the aspect of data modelling and predictive analytics, popular commercial tools are SAS, SPSS, Alpine Minor and STATISTICA/MATHEMATICA. These tools offer descriptive and predictive modelling capabilities for big data with highly interactive interfaces. R, Python, Octave and WEKA are popular free or open-source alternatives to the commercial packages commonly used in academia. For example, many universities and colleges use Octave as an alternative to Matlab as it has some of the functionalities of Matlab (EMC Educational Services, 2015).

Lastly, we consider data visualisation tools. Tableau is one of the fastest-growing big data visualisation tools on the market. Tableau is prevalent in large companies such as QlikQ, Oracle Hyperion and Cognos. Plotly is another visualisation tool that is up and coming. Plotly has many APIs and libraries that integrate popular data science tools such as Python, Matlab, and R (EDUCBA, n.d.).

---

7. Conclusion

This paper aims to derive a smaller and more optimal set of features from training, test and evaluating a binary and multiclass classifier to detect and classify network intrusion threats on LANs accurately and achieve an accuracy of at least 95%. To do this, we needed to go through the data analytics process, which included: data discovery, data preparation, model planning, model building, communicating results and operationalising, as described by the EMC Educational Services (2015, p. 29-30).

In the data discovery phase, we explored the dataset's structure, looked specifically at some of the features and viewed some descriptive statistics. Our efforts in this phase led to the formulation of the problem for investigation and our initial hypothesis. We used visualisations to give graphical insights about underlying patterns in the data.

In the data preparation phase, we took a deep dive into exploring the data more granularly. We zoomed in to observe the characteristics of the data stored in the columns and carried out data cleaning and transformation (ELT) tasks to get the dataset in a state that will facilitate more advanced analytics work. Such tasks include normalisation, balancing, and vectorisation.

In the model planning phase, we carried out in-depth research to learn about the technical aspects of machine learning models and used the knowledge gained about the dataset to select some models for our experiment. We further carried out the advanced analytical task to investigate the relationship between features and determine their relevance in a process called feature selection and engineering.

We split the resulting dataset into training and testing sets for our machine learning models. In the model-building phase, we built out our models, trained, tested, and evaluated them, and carried out cross-validation and hyperparameter optimisation to ensure that we got the highest performance from our models.

In the communication phase, we tabulated and discussed the results.

Finally, the source codes were uploaded to a GitHub repository and made available in the operational phase.

The results of our experiment showed that it was possible to derive a smaller and more optimal set of features that can achieve an accuracy score of at least 95%. For our binary classification model, the Gradient Boosted Tree Classifier gave the best performance recording an accuracy score across all the metrics of approximately 99.6%. For the multiclass classification, the Decision Tree Classifier performed the best with an average accuracy score of 98.22% across all metrics, followed closely by the Random Forest Classifier with an average of 98.16%.

Overall, the tree-based methods have out-performed the probabilistic methods (Naive Bayes and Logistic Regression) in detecting and classifying network intrusion threats.

---

8. References

1. Al-Daweri, Muataz S., Khairul A. Zainol Ariffin, Salwani Abdullah, and Mohamad F.E. Md. Senan. 2020. "An Analysis of the KDD99 and UNSW-NB15 Datasets for the Intrusion Detection System" *Symmetry* 12, no. 10: 1666. https://doi.org/10.3390/sym12101666
2. Apache Spark (n.d.). PySpark Documentation. [Online]. Available at: https://spark.apache.org/docs/latest/api/python/index.html
3. Aven, J. (2018). Data Analytics with Spark Using Python. USA: Pearson Technology Group.
4. Dini, P.; Saponara, S. (2021) Analysis, Design, and Comparison of Machine-Learning Techniques for Networking Intrusion Detection. *Designs 2021, 5, 9. https://doi.org/10.3390/designs5010009*
5. EDUCBA (n.d.). Big Data Technologies. [Online]. Available at: https://www.educba.com/big-data-technologies
6. EMC Educational Services (2015). Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data. Indiana, IN, USA: Wiley Professional Development (P&T).
7. Erl, T., Khattak, W. and Buhler, P. (2015). Big Data Fundamentals. Indiana, IN, USA: Pearson Technology Group.
8. Great Learning (2020). What is Cross Validation in Machine learning? Types of Cross Validation. [Online]. Available at: https://www.mygreatlearning.com/blog/cross-validation/
9. Hadoopsters (n.d.). Apache Sparke Starter Guide from Hadoopster. [Online]. Available at: https://hadoopsters.com/spark/
10. Kasango, S. M. and Sun, Y. (2020). Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *Journal of Big Data 7 (105).* https://doi.org/10.1186/s40537-020-00379-6
11. Mahfouz, A., Abuhussein, A., Venugopla, D., and Shiva, S. (2020). Ensemble Classifiers for Network Intrusion Detection Using a Novel Network Attack Dataset. *Future Internet 12 (180).* https://doi:10.3390/fi12110180
12. Mendelevitch, O., Stella, C. and Eadline, D. (2016). Practical Data Science with Hadoop and Spark. USA: Pearson Technology Group.
13. Sharda, R., Delen, D. and Turban, E. (2020). Systems for Analytics, Data Science, & Artificial Intelligence: Systems for Decision Support, eBook, Global Edition. 11th edition. London, UK: Pearson International Content.
14. Shaw, S., Vermeulen, A., Gupta, A. and Kjerrumgaard, D. (2016). Practical Hive. New York, NY, USA: Springer Nature.
15. Spark By Examples | Learn Spark Tutorial with Examples (n.d.). [Online]. Available at: https://sparkbyexamples.com/
16. Towards Data Science (2018). Multi-class text Classification with Pyspark. [Online]. Available at: https://towardsdatascience.com/multi-class-text-classification-with-pyspark-7d78d022ed3