MTH 410 Data Mining for Cybersecurity Midterm

Dataset: CIC-CSE-IDS2018

Group Number: 23

Algorithms: Decision Tree Classifier - Voting Classifier - Isolation Forest

By:

Obadah Nidal Ghizawi 121200038

Hassan Fakih Osman 121200078

Khaled El Arabi El Azzouzi 120200125

Professor: Umut Mennan Güder

Github Link: https://github.com/Retsuko64/Anomaly-Detection-Using-CSE-CIC-2018

Submission Date: 20/05/2024

CSE-CIC-IDS2018: Midterm Assignment

1 - Introduction:

The CSE-CIC-IDS2018 dataset is a collaborative project between the Communications Security Establishment (CSE), and the Canadian Institute for Cybersecurity (CIC). The primary purpose of the CSE-CIC-IDS2018 dataset is "anomaly detection". Anomaly detection is an important area of study for many researchers as it is highly useful in detecting novel attacks [1]. Utilizing anomaly detection in real-world applications is easier said than done though. The real-world systems generally require a substantial amount of testing, evaluation, and tuning before deployment. To accomplish this, the best way is to run the system over real labeled network traces with a comprehensive and extensive set of intrusions and abnormal behavior [1]. However, this is quite challenging because the availability of datasets to help us perform testing and evaluation is rare, whether because the datasets are internal and cannot be shared, heavily anonymized, and do not reflect current trends, or they lack certain statistical characteristics, the perfect dataset does not exist, thus researchers are forced to use datasets that are “suboptimal” but due to changing network behaviors and patterns, and the evolution of intrusions, researchers find it important to move towards more dynamically generated datasets that not only reflect the traffic compositions and intrusions of that time, but are also modifiable, extensible, and reproducible. The purpose of this dataset lies in the overcoming of the shortcomings above. A systematic approach was devised to generate datasets primarily to analyze, test, and evaluate intrusion detection systems, with a focus on network-based anomaly detectors [1]. The dataset can be found in the following link: https://www.unb.ca/cic/datasets/ids-2018.html

CIC and CSE devised such a system based on the creation of user profiles that contain abstract representations of events and behaviors seen on the network. Then such profiles will be merged to generate various kinds of datasets each with unique features. The final dataset of CSE-CIC-IDS2018 contains 7 attack scenarios: Brute Force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside [1].

To extract the features for this study, they used a tool called "CICFlowMeter", a network traffic flow generator written in Java. This specific tool gave them advantages in terms of flexibility and better control throughout the flow timeout [1]. Additionally, it generates bidirectional flows, where the first packet determines the forward/backward directions. The 83 features in this study are statistical features like duration, number of packets, number of bytes, etc. which are calculated separately in the forward and backward direction. Primarily, the first set of features contains information regarding the flow duration, total packets in the forward/backward directions, total size of packets in both directions, minimum/maximum size of packets in both directions, standard deviation of size of packets in both directions, etc. Most of the information stored in features is relevant to the packets sent in both directions. Other features store different information, mostly regarding packet numbers, flow byte rates, ratios, mean idle times, etc., but features mainly lie in the first category. After they extract the features and create the CSV file, they label the data using their attack scenarios schedule and the IPs and ports of the source and destination along with the protocol name to label the data per flow [1]. These features are essential for anomaly detection and intrusion detection systems since they can help a model learn from tailored information through the CICFlowMeter network traffic flow generator.

Feature Name	Description
fl_dur	Flow duration
tot_fw_pk	Total packets in the forward direction
tot_bw_pk	Total packets in the backward direction
tot_l_fw_pkt	Total size of packet in forward direction
fw_pkt_l_max	Maximum size of packet in forward direction
fw_pkt_l_min	Minimum size of packet in forward direction
fw_pkt_l_avg	Average size of packet in forward direction
fw_pkt_l_std	Standard deviation size of packet in forward direction
Bw_pkt_l_max	Maximum size of packet in backward direction
Bw_pkt_l_min	Minimum size of packet in backward direction

Table 1.1: [1] Snippet of Features List

Our aim during this study is to build an anomaly detection model to identify network intrusion attacks through the CSE-CIC-IDS2018 dataset. To accomplish this, we will need to take a small subset of the dataset (since the entire dataset is > 400 GB), preprocess the dataset, select relevant features, perform exploratory data analysis (EDA), then fit our model and test it. The dataset of this study is found in the following link:

https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv

2 - Data Preprocessing:

The dataset is made up of 10 files each captured on a specific day with different types of attacks as shown in the table below:

File	Type of Attacks
03-02-2018	Bot
02-28-2018 - 03-01-2018	Infiltration
02-22-2018 - 02-23-2018	Brute Force -XSS & -Web / SQL Injection
02-21-2018	DDOS attack-LOIC-UDP / DDOS attack-HOIC
02-20-2018	DDoS attacks-LOIC-HTTP
02-16-2018	DoS attacks-SlowHTTPTest / DoS attacks-Hulk
02-15-2018	DoS attacks-GoldenEye / DoS attacks-Slowloris
02-14-2018	FTP-BruteForce / SSH-Bruteforce

Figure 2.1: Table showing the different types of attacks present in each file of our dataset

Due to hardware limitations, we decided to cover some of the attacks available and not all of them. So, from the files available, we chose the ones on date ‘03-02-2018’ - ‘03-01-2018’ - ‘02-23-2018’ - ‘02-14-2018’, which covers Infilteration, Brute Force -XSS, Brute Force -Web, SQL Injection, Bot, FTP-BruteForce, and SSH-Bruteforce.

The dataset is of 3476825 instances made up of 80 features as discussed in the previous section. Two columns ‘Dst Port’ (Dst = Destination) and ‘Timestamp’ were dropped because they are just metadata and not to be used during classification.[5][2]

The ‘Protocol’ column is made up of 3 unique values [6, 0, 7], however since it is a categorical column, it was one-hot encoded and now there exist two columns ‘Protocol_6’ and ‘Protocol_17’. [5]

10,423 rows carry a null value which are present in column ‘Flow Byts/s’, which were dropped. Additionally, 12,156 rows carry values that are considered ‘infinity’ which are present in both ‘Flow Byts/s’ and ‘Flow Pkts/s’ columns, which were replaced with the maximum value in each column, 850,000,000 in ‘Flow Byts/s’ and 4,000,000 in ‘Flow Pkts/s’. The reason we had to deal with both ‘nan’ (Not a Number) and infinite values is that, if not dealt with, it would cause some problems when creating our models later on.

In the dataset, the target labels were encoded with values, because the models can’t learn from data of type ‘strings’ or ‘object’. So, Benign, Bot, Brute Force -Web, Brute Force -XSS, FTP-BruteForce, Infiltration, SQL Injection, and SSH-BruteForce were encoded to 0, 1, 2, 3, 4, 5, 6, 7 respectively, using the LabelEncoder from Scikit-Learn.

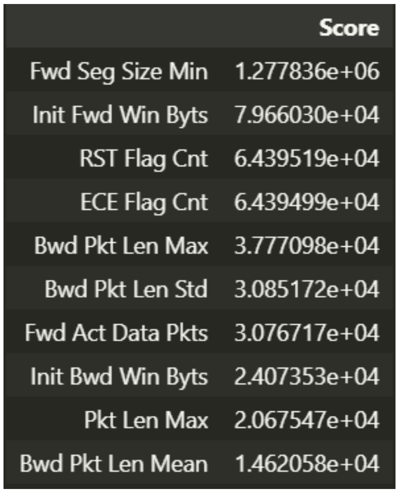
There are 407231 rows duplicates of other rows, and such duplicate rows can let our models give biased results, or even affect its performance, so it was all dropped.

All columns should carry positive values, however, certain instances carry negative values which were dropped.

Certain columns carry values of greater weight than the other columns. For example, in one of the rows, the ‘Flow Duration’ column contains a value of ‘115307855’, which is much bigger than the value 5 in column ‘Tot Fwd Pkts’, therefore the columns were normalized to values between [0, 1] using ‘MinMaxScaler’ from the Scikit-Learn library, making the columns equally weighted. The normalization formula is shown below:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Currently, there are 78 features, however, some features carry but little or no relevant information for the model to classify the target labels, which then would consider the features as noise for the model, lowering the accuracy and its overall performance. Thus we have to make feature selections, and only select the features of high importance. To do that, we used the ‘SelectKBest’ function from the Scikit-Learn library which selects the best features according to the K highest scores [3]. The score function used for ‘SelectKBest’ is the ‘f-classif’ which computes the ANOVA F-value for the provided sample. We decided to select the top 10 out of 78 features, which are: 'Bwd Pkt Len Max', 'Bwd Pkt Len Mean', 'Bwd Pkt Len Std', 'Pkt Len Max', 'RST Flag Cnt', 'ECE Flag Cnt', 'Init Fwd Win Byts', 'Init Bwd Win Byts', 'Fwd Act Data Pkts', 'Fwd Seg Size Min'. The feature scores are shown in the table in Fig. 2.2:



	Score
Fwd Seg Size Min	1.277836e+06
Init Fwd Win Byts	7.966030e+04
RST Flag Cnt	6.439519e+04
ECE Flag Cnt	6.439499e+04
Bwd Pkt Len Max	3.777098e+04
Bwd Pkt Len Std	3.085172e+04
Fwd Act Data Pkts	3.076717e+04
Init Bwd Win Byts	2.407353e+04
Pkt Len Max	2.067547e+04
Bwd Pkt Len Mean	1.462058e+04

Figure 2.2: Table showing the selecting features scores in descending form

The process we decided to follow for preprocessing is shown in Fig 2.3, which was made by László Göcs, and Zsolt Csaba Johanyák [2]. However, with the exception that we didn’t delete infinite values and, as stated previously, exchanged them with the maximum value of their columns, and additionally, we dropped duplicate rows.

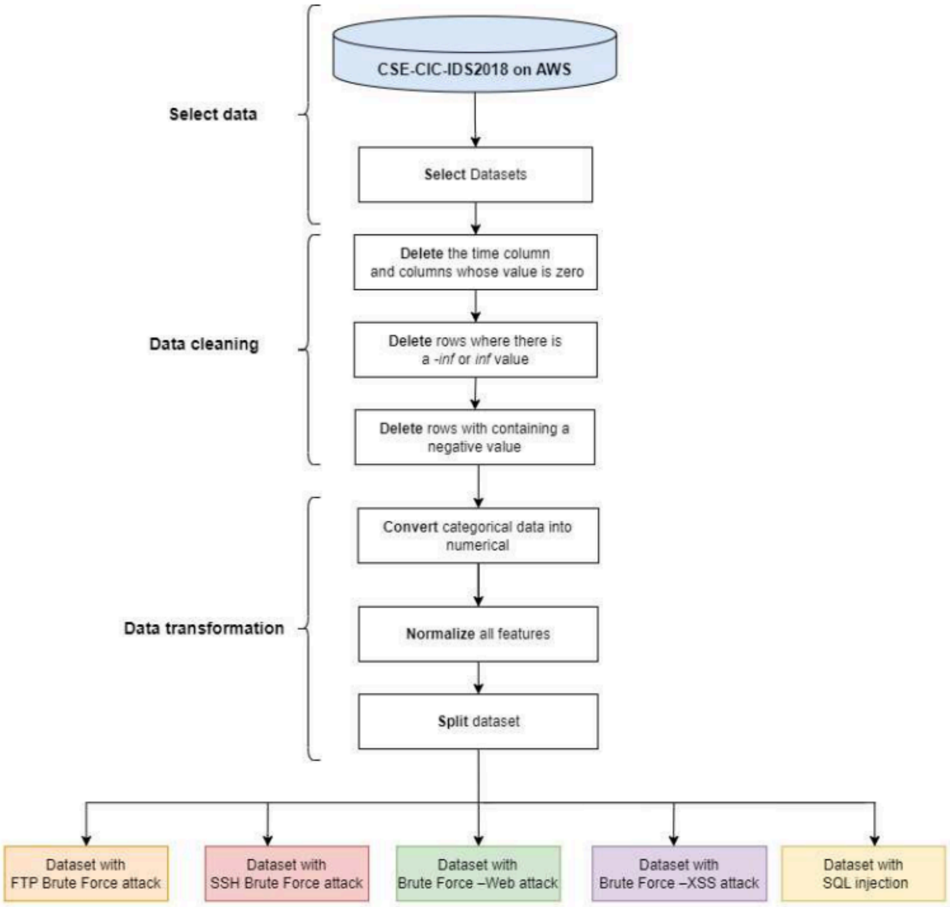

CSE-CIC-IDS2018 on AWS
Select data
Select Datasets
Data cleaning
Delete the time column and columns whose value is zero
Delete rows where there is a -inf or inf value
Delete rows with containing a negative value
Data transformation
Convert categorical data into numerical
Normalize all features
Split dataset
Dataset with FTP Brute Force attack
Dataset with SSH Brute Force attack
Dataset with Brute Force –Web attack
Dataset with Brute Force –XSS attack
Dataset with SQL injection

Figure 2.3: [2] Steps followed for data preprocessing

3 - Exploratory Data Analysis (EDA):

The class distributions are shown in Figure 3.1 below. The major problem noticed here is the huge class imbalance where we have approx. 80% Benign values and 20% Malicious data, which could affect the performance of the models later on.

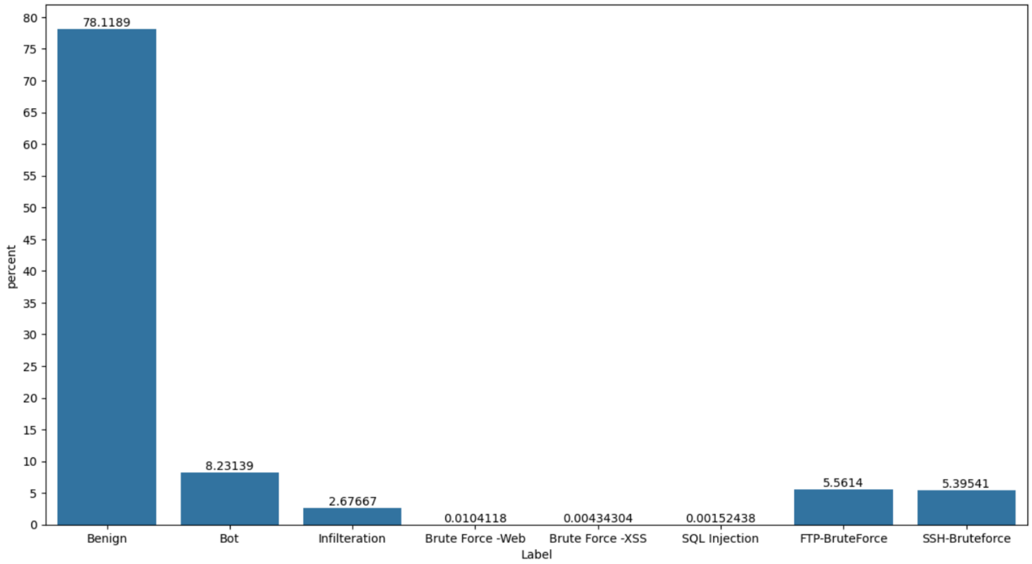


Figure 3.1: Histogram showing the frequency (%) of the classes in the dataset.

Figure 3.2 shows a violin plot that represents the numerical distributions of the 'Fwd Seg Size Min' column which is the minimum segment size in the forward direction for both benign and malicious packets, note that 'Malicious' is the union of all the attacks shown in Figure 3.1. We can see how there are similar minimum sizes of 10 and 20 in both Malicious and Benign classes, however, there seems to be an abnormal level of minimum sizes between 30 and 40 for malicious packets, which is absent in benign packets. Such observation could be a good indicator that a packet is malicious if its minimum segment size is larger than usual.

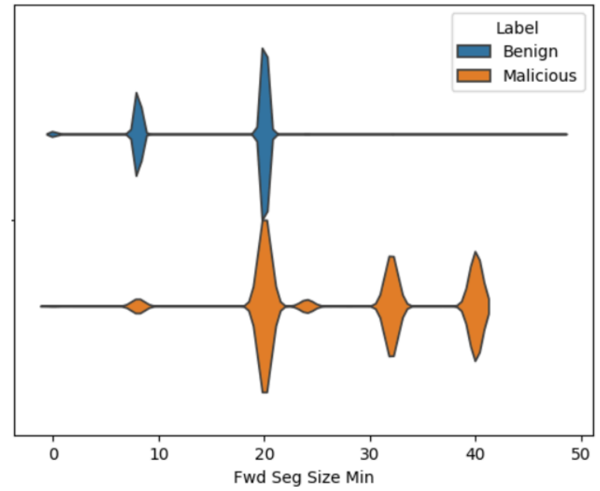


Figure 3.2: Violin plot showing the numerical distributions of the minimum forward segment size.

Figure 3.3 shows another violin plot representing the numerical distributions of the number of bytes sent in the initial window in the forward direction [1] for both benign and malicious packets. We can see similar distributions between 0 to 10,000 bytes in both classes, however, there seems to be a big number of values around 20,000 and 30,000 in the malicious packets, which could be another indicator of malicious packets if the number of bytes sent in the initial window in the forward direction is large.

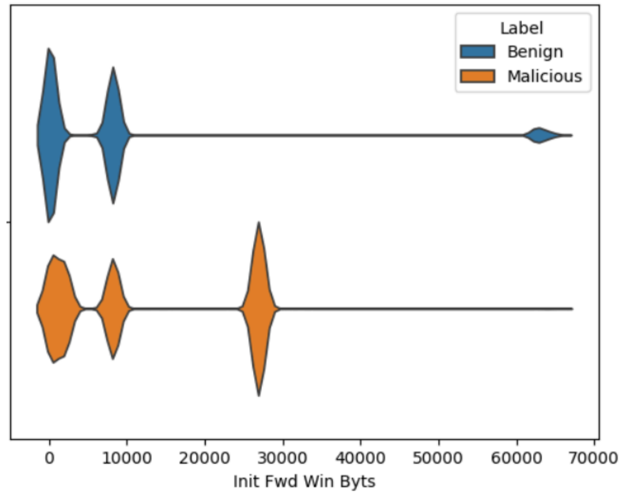
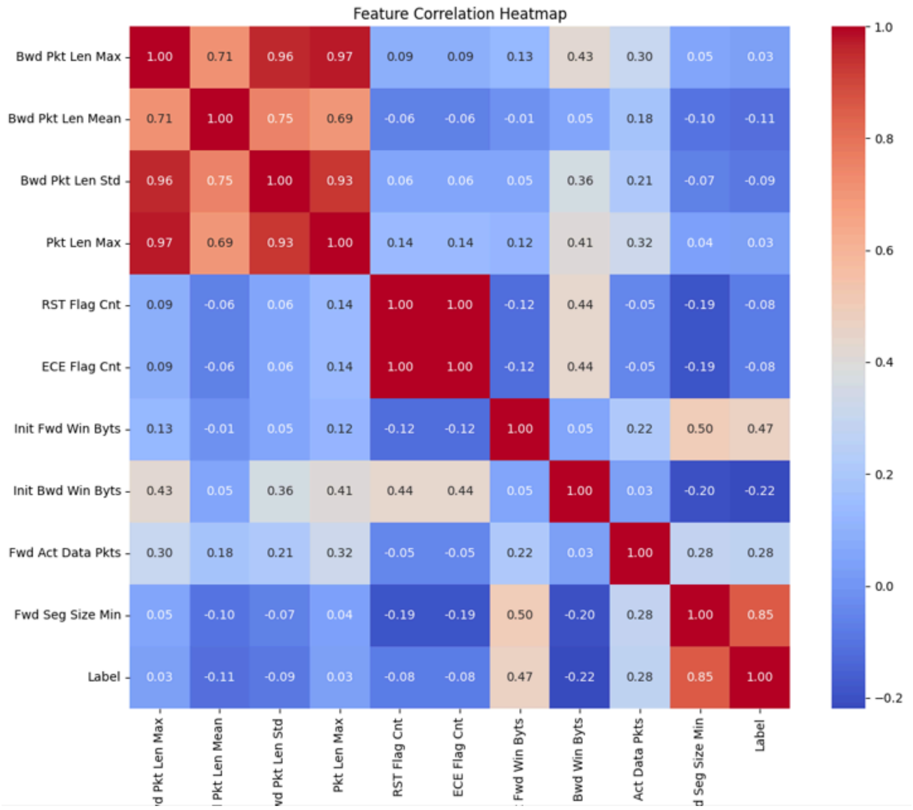


Figure 3.3: Violin plot showing the numerical distributions of the 'Init Fwd Win Byts' column



	d Pkt Len Max	l Pkt Len Mean	vd Pkt Len Std	Pkt Len Max	RST Flag Cnt	ECE Flag Cnt	: Fwd Win Byts	Bwd Win Byts	Act Data Pkts	d Seg Size Min	Label
Bwd Pkt Len Max	1.00	0.71	0.96	0.97	0.09	0.09	0.13	0.43	0.30	0.05	0.03
Bwd Pkt Len Mean	0.71	1.00	0.75	0.69	-0.06	-0.06	-0.01	0.05	0.18	-0.10	-0.11
Bwd Pkt Len Std	0.96	0.75	1.00	0.93	0.06	0.06	0.05	0.36	0.21	-0.07	-0.09
Pkt Len Max	0.97	0.69	0.93	1.00	0.14	0.14	0.12	0.41	0.32	0.04	0.03
RST Flag Cnt	0.09	-0.06	0.06	0.14	1.00	1.00	-0.12	0.44	-0.05	-0.19	-0.08
ECE Flag Cnt	0.09	-0.06	0.06	0.14	1.00	1.00	-0.12	0.44	-0.05	-0.19	-0.08
Init Fwd Win Byts	0.13	-0.01	0.05	0.12	-0.12	-0.12	1.00	0.05	0.22	0.50	0.47
Init Bwd Win Byts	0.43	0.05	0.36	0.41	0.44	0.44	0.05	1.00	0.03	-0.20	-0.22
Fwd Act Data Pkts	0.30	0.18	0.21	0.32	-0.05	-0.05	0.22	0.03	1.00	0.28	0.28
Fwd Seg Size Min	0.05	-0.10	-0.07	0.04	-0.19	-0.19	0.50	-0.20	0.28	1.00	0.85
Label	0.03	-0.11	-0.09	0.03	-0.08	-0.08	0.47	-0.22	0.28	0.85	1.00

Figure 3.4: Correlation Heatmap with the selected features

This correlation heatmap in Figure 3.4 shows us the correlations between our selected features and the label. These selected features are the most relevant to the label, which was selected through the ANOVA F-value test conducted in the “SelectKBest” function [3]. For example, the ‘Fwd Seg Size Min’ feature positively correlates greatly with the label (0.85). So that means as the minimum size increases, it is more likely to be a malicious packet, which also agrees with the observation found in Figure 3.2.

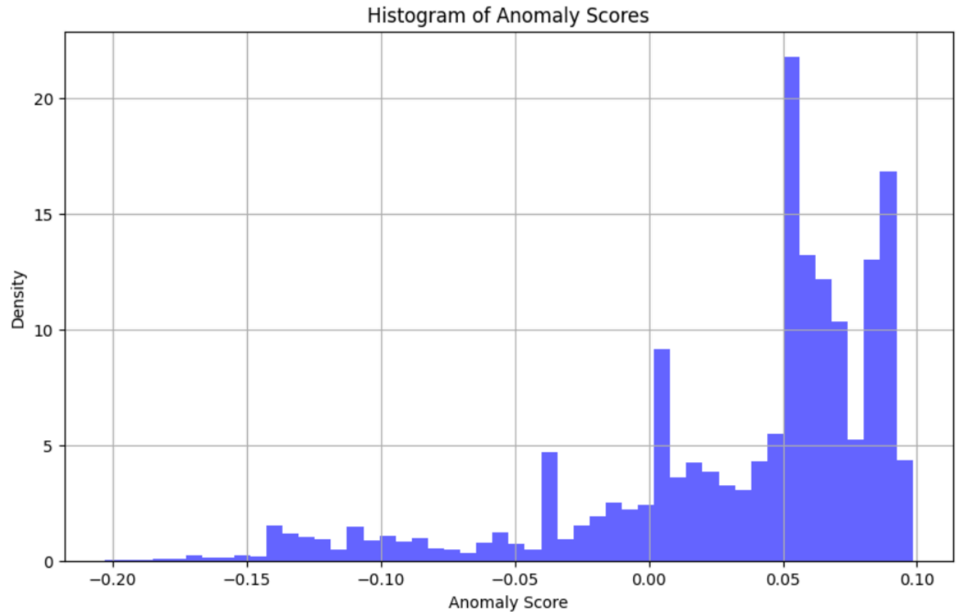


Figure 3.5: Histogram of Anomaly Scores

This Histogram of anomaly scores in Figure 3.5 shows us where the majority of our anomaly scores (calculated by the Isolation Forest classifier shown in section 4.3) lie. This helps us realize where our normal points lie (as there will be a higher density of these points since approximately 80% of our points are benign), as well as our anomaly points (a.k.a. 'Malicious' points).

4 - Data Mining Techniques and Applications:

To start building our models, we have to split our data into two. One for training, and the other for testing. The testing data has to be completely separated from the training process or else it gives us biased results in the end. In this study, we split our data into 70% training data and 30% testing data.

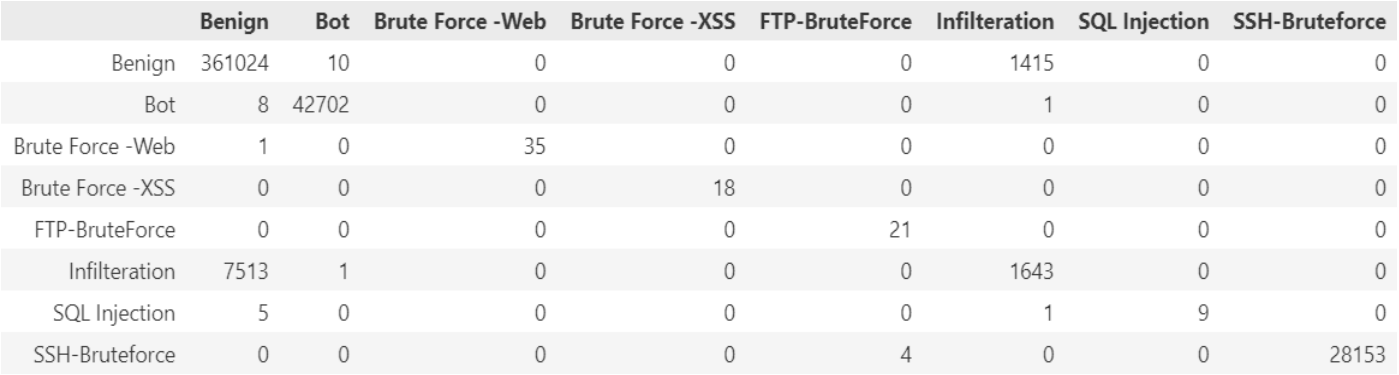
We also relied on 4 different metrics which are accuracy, recall, precision, and F2 score to evaluate the models.

4.1 - Decision Tree Classifier

Decision Trees is a supervised machine learning algorithm, useful for both Classification and Regression problems. It predicts the output label by learning simple decision rules inferred from the input features. In other words, it is like a series of If-Then-Else tests on the input data that would then lead to the final output. A tree can be seen as a piecewise constant approximation [6].

The criterion used is the “Gini”, and such criterion measures the quality of a split. The Gini index or Impurity measures the probability of a random instance being misclassified when chosen randomly. The lower the Gini index, the more pure a split is, and the lower the likelihood of misclassification [8].

After training the decision tree classifier with the training data, and evaluating it using the testing data, the results are shown in the table below:



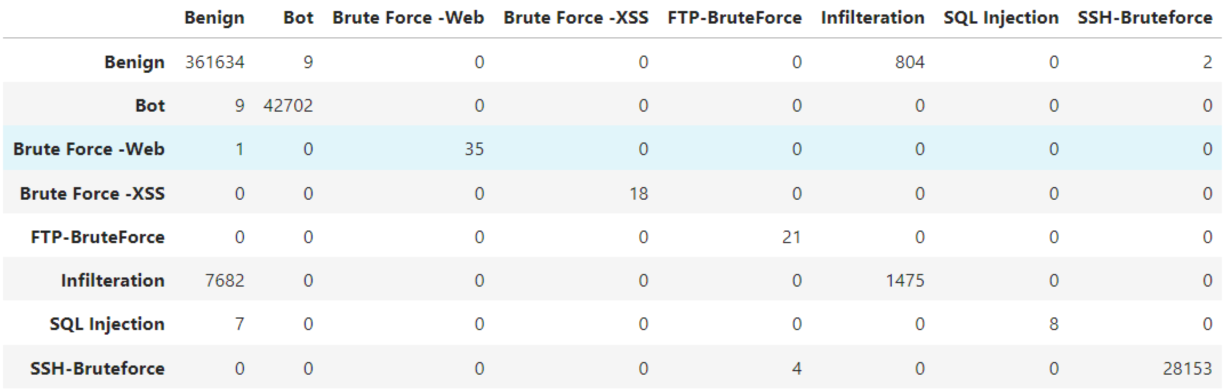
	Benign	Bot	Brute Force -Web	Brute Force -XSS	FTP-BruteForce	Infilteration	SQL Injection	SSH-Bruteforce
Benign	361024	10	0	0	0	1415	0	0
Bot	8	42702	0	0	0	1	0	0
Brute Force -Web	1	0	35	0	0	0	0	0
Brute Force -XSS	0	0	0	18	0	0	0	0
FTP-BruteForce	0	0	0	0	21	0	0	0
Infilteration	7513	1	0	0	0	1643	0	0
SQL Injection	5	0	0	0	0	1	9	0
SSH-Bruteforce	0	0	0	0	4	0	0	28153

Table 4.1: Confusion Matrix for the Decision Tree Model

The model took aprox. 5 seconds to train, and gave an accuracy of 0.98, precision of 0.97, recall of 0.98, and F1-score of 0.97. There seems to be a major number of misclassifications of infiltration attacks, which are mostly misclassified as Benign, consequently, some Benign packets are being misclassified as infiltration attacks. So the model seems to be confused about the two.

4.2 - Voting Classifier

Voting is a machine learning technique that combines several base models to improve results. In recent years, several algorithms were applied to Intrustion Detection Systems which were successful, however, each algorithm has its advantages and disadvantages in specific types of attacks. So to overcome this, we can combine the results of different models, which would then improve generalizability and robustness [7]. For this, we used Random Forest, Decision Trees, and AdaBoost as our base estimators, and the model will choose the output according to a majority vote. The results are shown in the table below.



	Benign	Bot	Brute Force -Web	Brute Force -XSS	FTP-BruteForce	Infilteration	SQL Injection	SSH-Bruteforce
Benign	361634	9	0	0	0	804	0	2
Bot	9	42702	0	0	0	0	0	0
Brute Force -Web	1	0	35	0	0	0	0	0
Brute Force -XSS	0	0	0	18	0	0	0	0
FTP-BruteForce	0	0	0	0	21	0	0	0
Infilteration	7682	0	0	0	0	1475	0	0
SQL Injection	7	0	0	0	0	0	8	0
SSH-Bruteforce	0	0	0	0	4	0	0	28153

Table 4.2: Confusion Matrix for the Voting Classifier

The model gave an accuracy of 0.98, precision of 0.98, recall of 0.98, F1-score of 0.98. One thing that has improved from only using a decision tree classifier, is the precision score for the infiltration attack, which increased from 0.54 to 0.65.

4.3 - Isolation Forest

The algorithm that made the most sense and is the most applicable to our particular use case is the Isolation Forest algorithm. The Isolation Forest algorithm is an anomaly detection algorithm used in machine learning. It's very effective for identifying anomalies or outliers in large datasets. Any data point or observation that deviates significantly from the other observations is called an anomaly/outlier. Anomaly detection is very important and is applicable in areas like fraudulent bank transactions, network intrusion detection, etc [4].

The Isolation Forest algorithm primarily uses binary trees to detect anomalies, which results in a linear time complexity and low memory usage that is well-suited for processing large datasets. Since we are using the CSE-CIC-IDS2018 dataset, this is particularly advantageous for our use case since the dataset is very large.

During our study, we employed a small subset of network data from particular days, as explained in the beginning of the preprocessing section, to train our model as quickly as possible (as well as the fact that we are limited by available hardware). To classify certain points as anomalies, we calculate "anomaly scores" for our test dataset, then find an optimal "threshold", where if the anomaly score exceeds this threshold, it's classified as an "anomaly".

Threshold	Accuracy	Recall	Precision	F1
0.0922	**0.79**	0.79	0.67	0.72
-0.1358	0.11	0.11	**0.75**	0.05
0.0922	0.79	**0.79**	0.67	0.72
0.0922	0.79	0.79	0.67	**0.72**

Table 4.3: Result table for the Isolation Forest Classifier

During each of our trials, we tried to optimize the threshold value for a certain statistic (e.g. accuracy, recall, precision, f1). In the table above, the specific statistic we were optimizing for is bolded in each trial. Most of the trials point towards one threshold value (0.0922). Thus, we settled on the threshold value of 0.0922.

5 - Conclusion:

Model	Accuracy	Precision	Recall	F1-score	Execution Time (approx.)
Decision Tree	**0.98**	**0.97**	**0.98**	**0.97**	**5 sec**
Voting Classifier	0.98	0.98	0.98	0.98	10-15 min
Isolation Forest	0.79	0.67	0.79	0.72	4 sec

Table 5.1: Table showing the final results of this study

Through the utilization of the CSE-CIC-IDS2018 dataset, we were able to build an anomaly detection algorithm by utilizing Decision Trees. To accomplish this, we selected a subset of the initial dataset, preprocessed the dataset, performed feature selection, did Exploratory Data Analysis (EDA), and then trained and tested different kinds of models, which are Decision Tress, Voting Classifier (which is made up of Random Forest, AdaBoost, and Decision Trees), and finally Isolation Forest. We tested the Isolation Forest algorithm with a range of threshold values in an attempt to maximize our summary statistics. However, as a result of this study, it seems that Decision Trees is the most qualified to do anomaly detection, with an accuracy of 0.98, precision of 0.97, recall of 0.98, F1-score of 0.97, and finally a fast execution time (approx. 5 seconds). In the future, hyperparameter tuning can be applied to further improve the models, and we could also create and test a DNN (Deep Neural Network) and compare its results with the current models in this study.

References

[1] https://www.unb.ca/cic/datasets/ids-2018.html

[2]https://arxiv.org/pdf/2307.11544

[3]https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

[4]https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/

[5](PDF) MACHINE LEARNING TECHNIQUES FOR INTRUSION DETECTION AND BANDWIDTH IN NETWORK SYSTEM USIN CIC-IDS-2018 (researchgate.net)

[6]https://scikit-learn.org/stable/modules/tree.html

[7] https://ieeexplore.ieee.org/abstract/document/9172014

[8]https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c#:~:text=The%20Gini%20Index%20or%20Impurity,lower%20the%20likelihood%20of%20misclassification.&text=P(i)%20represents%20the%20ratio,of%20observations%20in%20node.