İstanbul
Bilgi University
LAUREATE INTERNATIONAL UNIVERSITIES

INTRUSION DETECTION EVALUATION DATASET (CIC-IDS2017)

Analysis and Discovery of CIC-IDS2017 Dataset

Emine Esin Yılmaz 120200059

Pınar Sude Gürdamar 120203073

Resul Erdem Arduç 119200056

Instructor

Mennan Güder

Table of Contents

a) Introduction

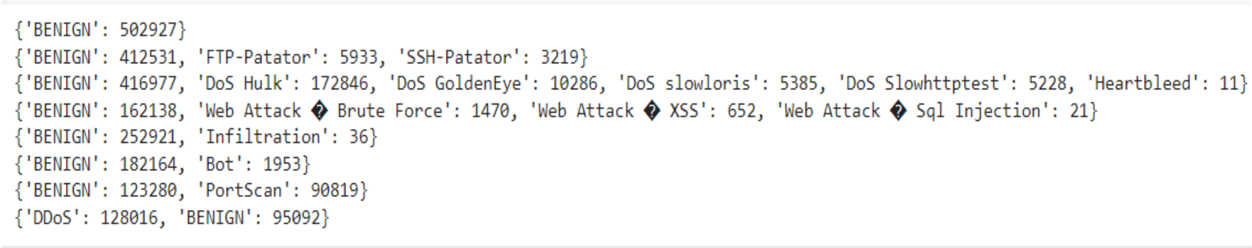
The CIC-IDS2017 dataset is one of the important datasets for cybersecurity analysis. The CIC-IDS2017 dataset contains benign and six common attack network flows, which mirror the true real-world data (PCAPs). Additionally, it contains labeled flows based on timestamps, source and destination IP addresses, source and destination ports, protocols, and attack from the network traffic analysis performed using CICFlowMeter.

The obtained data set was stored in 8 different CSV files. These are drawn separately as working hours on Monday, working hours on Tuesday, working hours on Wednesday, morning working hours and afternoon working hours on Thursday, and both morning and afternoon working hours on Friday. Friday afternoon working hours appear in two ways: portscan and DDOS.

This report explains the applied data preprocessing, exploratory data analysis, and data mining techniques to detect-classify-cluster potential security threats in detail.

b) Data Preprocessing

Since the obtained dataset was stored in 8 different CSV files, the preprocessing steps were done separately on each of them. In the first step, a function was written to remove empty values and duplicates from the datasets. After the removal of NaN values and duplicates, different labels that each dataset has, and their counts were examined.



```
{'BENIGN': 502927}
{'BENIGN': 412531, 'FTP-Patator': 5933, 'SSH-Patator': 3219}
{'BENIGN': 416977, 'DoS Hulk': 172846, 'DoS GoldenEye': 10286, 'DoS slowloris': 5385, 'DoS Slowhttptest': 5228, 'Heartbleed': 11}
{'BENIGN': 162138, 'Web Attack � Brute Force': 1470, 'Web Attack � XSS': 652, 'Web Attack � Sql Injection': 21}
{'BENIGN': 252921, 'Infiltration': 36}
{'BENIGN': 182164, 'Bot': 1953}
{'BENIGN': 123280, 'PortScan': 90819}
{'DDoS': 128016, 'BENIGN': 95092}
```

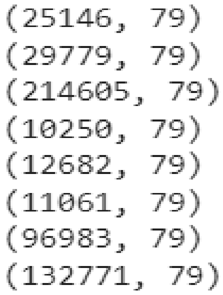
Figure 1: The Labels in Each Eight CSV Files and their Counts

In the data consisting of working hours on Monday, all the activity data observed have been labeled as benign. In the second dataset of working hours on Tuesday, 412531 of the activities were labeled as benign, 5933 have been observed as FTO-Patator, and 3219 are labeled as SSH-Patator. The six of the remaining pcap data files which the CIC-IDS2017 dataset consists, including the activities collected on Wednesday working hours, Thursday working hours morning web attacks and Thursday afternoon, Friday working hours morning, afternoon portscan and DDos pcap; have different types of attack tags, categorizing network traffic data into different attack types and normal status.

When we examine the data in our datasets, we can see that the majority of the data set consists of network traffic data labeled Benign. The fact that the number of data with the BENIGN label in our data is much higher than other attack types creates an imbalance in the dataset. Data imbalance refers to situations where one class has many more of the same samples than other classes. An unbalanced dataset makes it difficult for the model to learn minority classes and may result in a model biased against the majority class.

To prevent the problem of imbalance in our input data and achieve a better scoring model to detect different kinds of attacks and network traffic, the downsampling method is used.

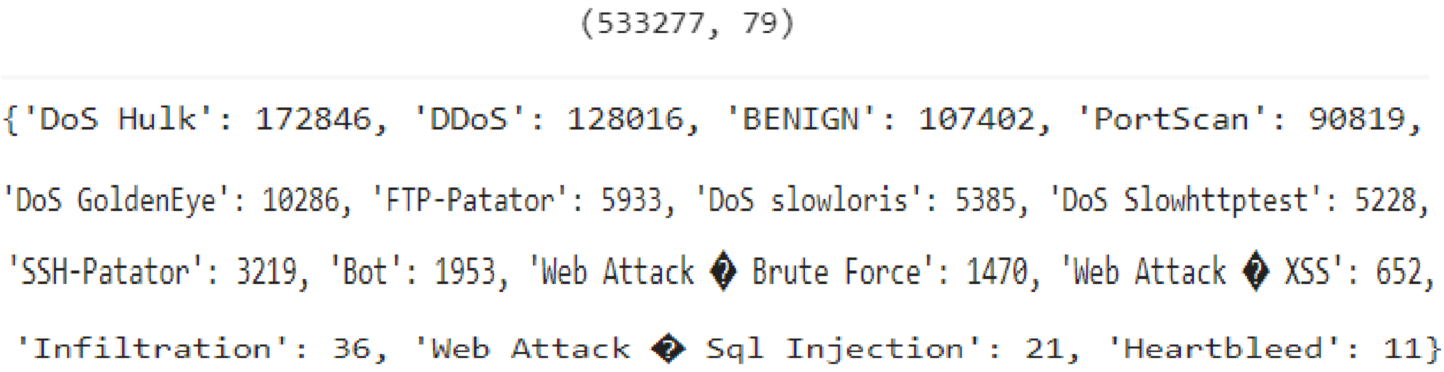
Downsampling is the process of balancing the dataset by reducing the number of data in the majority class (Majidi et al., 2023). During this process, the minority and the majority label classes were detected as shown in Figure 1 above. After the detection, A random subset (5%) is taken from the data belonging to the BENIGN class. All data from classes other than BENIGN are added to this subset as it is. Thus, the number of BENIGN data is reduced while all data of other classes are preserved, achieving a balanced dataset to use as input. The results are shown in Figure 2 below, containing the row and column sizes of each eight different datasets.



```
(25146, 79)
(29779, 79)
(214605, 79)
(10250, 79)
(12682, 79)
(11061, 79)
(96983, 79)
(132771, 79)
```

Figure 2: Record and feature (column) counts of each 8 dataset after downsampling

After balancing the datasets, eight separately processed data frames containing 8 csv data of the CIC-IDS2017 dataset were combined. The number of rows, columns, and labels of the dataset are shown in detail below as Figure 3.



```
(533277, 79)

{'DoS Hulk': 172846, 'DDoS': 128016, 'BENIGN': 107402, 'PortScan': 90819,
'DoS GoldenEye': 10286, 'FTP-Patator': 5933, 'DoS slowloris': 5385, 'DoS Slowhttptest': 5228,
'SSH-Patator': 3219, 'Bot': 1953, 'Web Attack � Brute Force': 1470, 'Web Attack � XSS': 652,
'Infiltration': 36, 'Web Attack � Sql Injection': 21, 'Heartbleed': 11}
```

Figure 3: The row, column, and label counts of the combined and sampled dataset.

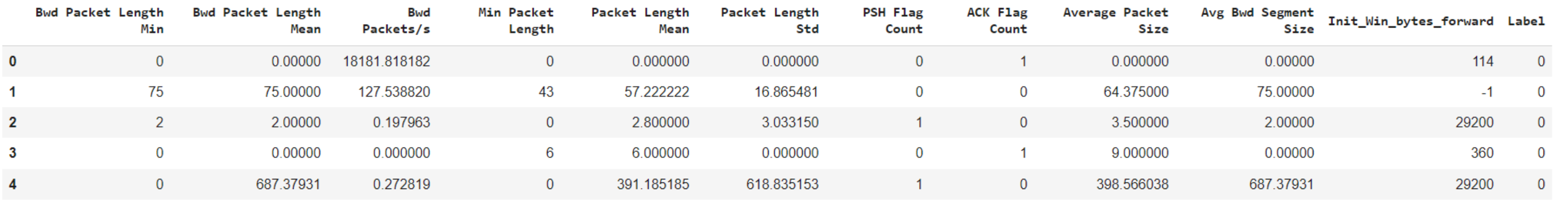
As shown in the figure above, the combined dataset has a total of 533277 recorded traffic data with 79 features. After the sampling, the count of the benign labeled data was reduced to 107402. The most common label in the combined dataset is DoS Hulk with 172846 instances, and the less common label is the Heartbleed with only 11 samples existent in the whole dataset.

In the following steps, data transformation techniques were applied to bring raw data into a format that can be processed by machine learning models. To transform the data, various techniques were used such as encoding the data, finding the correlation of the features with the label feature, selecting of the features according to the correlation, removing the outliers, and scaling the numerical data.

The data transformation begins with encoding the categorical values in the label column. Encoding is one of the crucial steps of data preprocessing since converting categorical variables into numerical values (encoding) enables machine learning models to work in harmony with categorical data. This transformation ensures that the data is correctly processed and analyzed by the model. It also improves model performance, reduces data size, and improves generalization ability. In addition, encoding provides numerical values to understand the relationships between categorical variables better (Potdar et al, 2017).

In the scope of the project, label encoding was performed on the combined and sampled final dataset.

After the encoded dataset was obtained, the correlation of the features of the data to the label features was examined. This process aims to reduce data size and improve model performance by selecting features that are important and relevant for model training. With this correlation step, the correlations between features in a data frame were analyzed, and important features were selected according to their relationship with a specific target variable, which is the “Label” column. First, the Pearson correlation coefficients between all numerical features in the data frame were calculated and a correlation matrix was created. Then, features with a correlation value greater than 0.20 (absolute value) (Senthilnathan and Samithamby, 2019) with the target variable were determined and a new data frame was created using these selected features. According to the results, only 12 out of 79 features were correlated more than 20 percent with the target variable, “Label”.



	Bwd Packet Length Min	Bwd Packet Length Mean	Bwd Packets/s	Min Packet Length	Packet Length Mean	Packet Length Std	PSH Flag Count	ACK Flag Count	Average Packet Size	Avg Bwd Segment Size	Init_Win_bytes_forward	Label
0	0	0.00000	18181.818182	0	0.000000	0.000000	0	1	0.000000	0.00000	114	0
1	75	75.00000	127.538820	43	57.222222	16.865481	0	0	64.375000	75.00000	-1	0
2	2	2.00000	0.197963	0	2.800000	3.033150	1	0	3.500000	2.00000	29200	0
3	0	0.00000	0.000000	6	6.000000	0.000000	0	1	9.000000	0.00000	360	0
4	0	687.37931	0.272819	0	391.185185	618.835153	1	0	398.566038	687.37931	29200	0

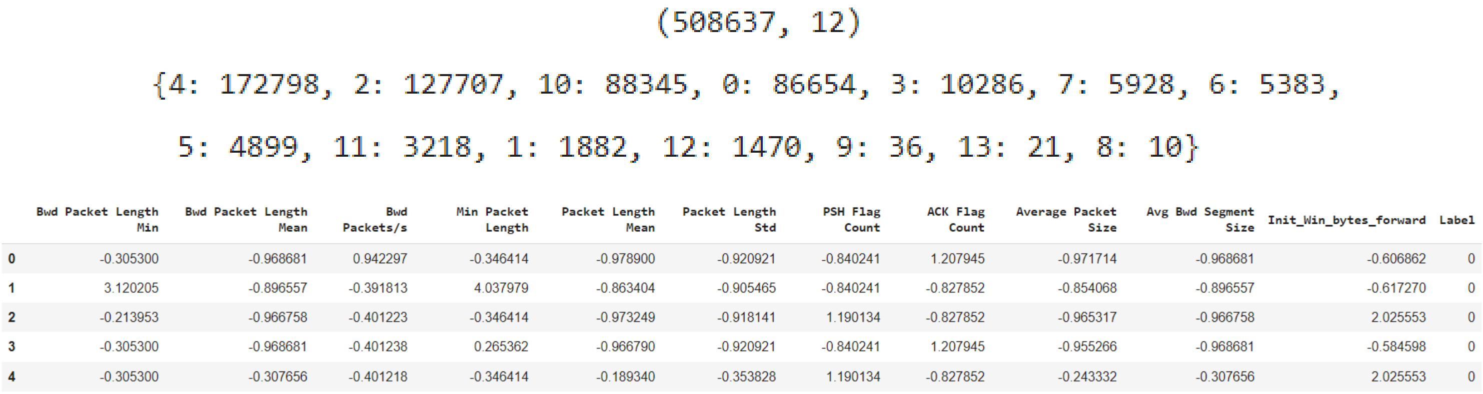
Figure 4: The 12 features that are correlated with the target feature more than 20 percent

In the last steps, removing the outliers and data scaling were performed.

Outliers are values that are rare in datasets, and significantly different from other observations. These values can mislead the data analysis and modeling process and make the results misleading. Therefore, the removal of outliers is an important step in the data processing process. Removing outliers increases the accuracy of analysis results and improves model performance. Furthermore, removing outliers allows for better interpretation of model results and improves the quality of the dataset. For these reasons, the removal of outliers plays an important role in data analysis and modeling processes and is essential for obtaining accurate results.

In the last step, StandardScaler was applied to the data to scale the numeric values. With this scaling step, it was aimed to address scale differences between different features and improve the performance of the model. Moreover, scaling is important because it also speeds up optimization processes, helps to make model comparisons better, and ensures that optimization algorithms such as gradient descent run more stably.

As a result of these preprocessing steps, the final dataset was obtained and served as an input for the exploratory data analysis processes.



```
(508637, 12)
{4: 172798, 2: 127707, 10: 88345, 0: 86654, 3: 10286, 7: 5928, 6: 5383,
 5: 4899, 11: 3218, 1: 1882, 12: 1470, 9: 36, 13: 21, 8: 10}
```

	Bwd Packet Length Min	Bwd Packet Length Mean	Bwd Packets/s	Min Packet Length	Packet Length Mean	Packet Length Std	PSH Flag Count	ACK Flag Count	Average Packet Size	Avg Bwd Segment Size	Init_Win_bytes_forward	Label
0	-0.305300	-0.968681	0.942297	-0.346414	-0.978900	-0.920921	-0.840241	1.207945	-0.971714	-0.968681	-0.606862	0
1	3.120205	-0.896557	-0.391813	4.037979	-0.863404	-0.905465	-0.840241	-0.827852	-0.854068	-0.896557	-0.617270	0
2	-0.213953	-0.966758	-0.401223	-0.346414	-0.973249	-0.918141	1.190134	-0.827852	-0.965317	-0.966758	2.025553	0
3	-0.305300	-0.968681	-0.401238	0.265362	-0.966790	-0.920921	-0.840241	1.207945	-0.955266	-0.968681	-0.584598	0
4	-0.305300	-0.307656	-0.401218	-0.346414	-0.189340	-0.353828	1.190134	-0.827852	-0.243332	-0.307656	2.025553	0

Figure 5: Row and Column Count of the Dataset, the Encoded Version of the Label Column and its’ Counts, Overall View of the Processed Dataset

The above figure 5 contains the final processed dataset’s properties. The final dataset contains 508637 of network traffic data and 12 features. After the shape of the dataset, the encodings of the “Label” column which consists of the labels of the attacks and the network traffic are printed, showing the numerical values that stand for each category. In the 4th row, the overall view of the dataset was included.

Preprocessing is one of the most crucial steps before integrating the dataset into the models for training and testing steps (Aditya and Navneet, 2022). The real-life data often consists of an overall imbalance, null values, and object data types which are not usable for machine learning models and data mining techniques. With the data preprocessing steps, the improvement of the performance, accuracy, and reliability of the model is achieved. Preprocessing the data improves data quality and reduces noise by correcting incomplete, inaccurate, or inconsistent information existent in the dataset. The steps used in the scope of this project starting from the elimination of the data imbalance, normalization of data, the extraction of irrelevant and repetitive records, and converting the data into the format required by the models are all included in the preprocessing processes (Bhaya, 2017). Thus, by processing the data, it is ensured that it is clean, balanced, and in a format suitable for the model, which allows the models to make more accurate predictions.

c) Exploratory Data Analysis (EDA)

The exploratory data analysis (EDA) for this dataset gives us a good idea of what's going on with all the different network traffic features and the target variable, which tells us what kind of network traffic each packet is. The features we're looking at are Bwd Packet Length Min, Bwd Packet Length Mean, Bwd Packets/s, Min Packet Length, Packet Length Mean, Packet Length Std, PSH Flag Count, ACK Flag Count, Average Packet Size, Avg Bwd Segment Size, and Init_Win_bytes_forward. The target variable, Label, was changed from being an object to an integer so it could be used with the rest of the data in the analysis.

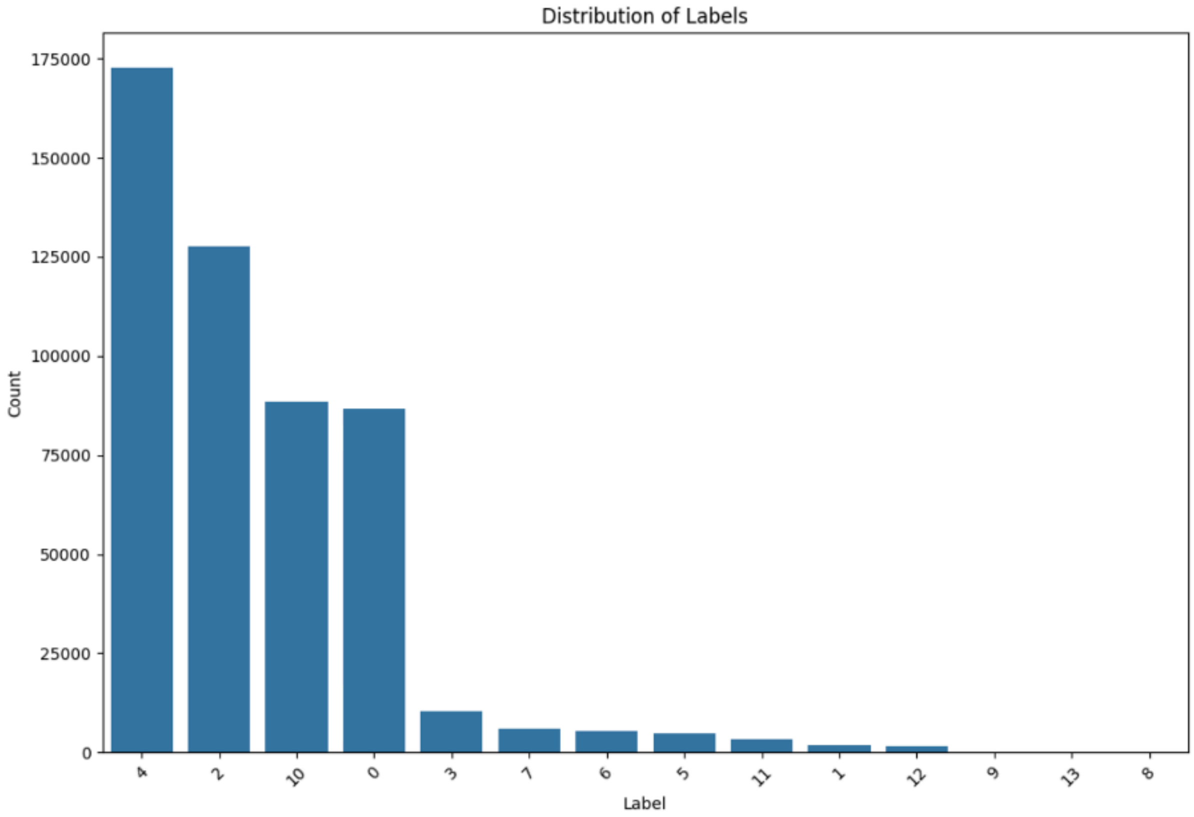


Figure 6 - Count of Labels

Figure 6 showcases the count of each label, highlighting a clear imbalance among the classes. The dataset's labels and their corresponding counts are as follows: 'DoS Hulk' (172,846 instances), 'DDoS' (128,016 instances), 'BENIGN' (107,402 instances), 'PortScan' (90,819 instances), 'DoS GoldenEye' (10,286 instances), 'FTP-Patator' (5,933 instances), 'DoS slowloris' (5,385 instances), 'DoS Slowhttptest' (5,228 instances), 'SSH-Patator' (3,219 instances), 'Bot' (1,953 instances), 'Web Attack – Brute Force' (1,470 instances), 'Web Attack – XSS' (652 instances), 'Infiltration' (36 instances), 'Web Attack – Sql Injection' (21 instances), and 'Heartbleed' (11 instances).

The label distribution plot shows that there's a big difference in how many times each type of label appears in the dataset. There are some labels that are really common, and others that hardly show up at all. The label with the most instances, 'DoS Hulk' (label 4), has a whopping 172,846 of them! That's way more than any other label. The next most common label is 'DDoS' (label 2) with 128,016, and then there's 'BENIGN' (label 0) with 107,402. 'PortScan' (label 10) also has a ton of instances, with 90,819.

In contrast, some labels have significantly fewer instances. 'DoS GoldenEye' (label 3) has 10,286 instances, 'FTP-Patator' (label 7) has 5,933 instances, 'DoS slowloris' (label 6) has 5,385 instances, 'DoS Slowhttptest' (label 5) has 5,228 instances, and 'SSH-Patator' (label 11) has 3,219 instances. Labels such as 'Bot' (label 1), 'Web Attack – Brute Force' (label 12), 'Web Attack – XSS' (label 14), 'Infiltration' (label 9), 'Web Attack – Sql Injection' (label 13), and 'Heartbleed' (label 8) have even fewer instances, with counts of 1,953, 1,470, 652, 36, 21, and 11, respectively.

This significant imbalance in the numbers of certain things in a dataset can be a problem for machine learning models. These models might get used to the majority classes and not do as well with the minority classes. This can make the whole model not work as well as it should. For example, the class 'Heartbleed' (label 8), with only 11 instances, may not be adequately represented during training, causing the model to have difficulty accurately predicting this class. Similarly, classes like 'Infiltration' (label 9) and 'Web Attack – Sql Injection' (label 13), with 36 and 21 instances, respectively, are underrepresented.

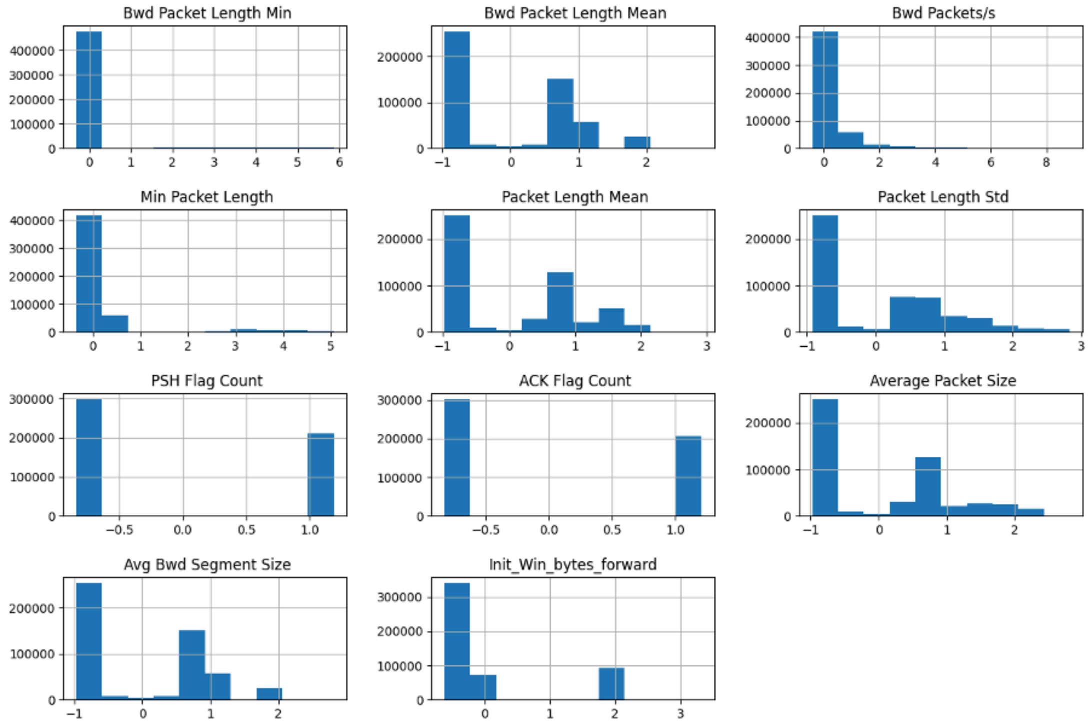


Figure 7 - Histogram of the Feature Values

Now, consider the Bwd Packet Length Min. Figure 7 shows a skewed distribution, with the majority of the values more or less crowded around zero, indicating that in most instances, the minimum of the backward packet length is quite small. There are a few outliers with quite larger values, as witnessed by the long tail to the right.

Bwd Packet Length Mean also reveals a right-skewed distribution in which most values appear bunched up at zero. That would tell us that the mean backward packet lengths are small for most of them, except in a few instances with larger packet length. There are several peaks inside the histogram, too, which would indicate that the packet lengths are variable among different instances.

Similarly, for Bwd Packets/s, a huge concentration of values is more inclined towards zero, with a long tail skewed to the higher side. Such a right-skewed distribution means that the rate of backward packets per second is generally low, with some bursts of higher packet rates.

For Min Packet Length, we see that the histogram shows that most minimum packet lengths are close to zero. From the graph, we notice that this feature contains very few longer packet lengths while the majority is differences of close to zero. This is consistent with network traffic, where most packets are minimum.

Packet Length Mean also shows much concentrator near zero, but the histogram spread more widely across different values, showing the greater variance in the mean packet length among the instances.

The Packet Length Std histogram shown in Figure 7 shows the standard deviation of packet lengths. Standard deviation is a measure of the spread of packet sizes within an instance. Most values are clustered close to zero, meaning that in many instances, the packet sizes are not spread out by much. However, there is a long tail stretching further out to the right, meaning that some instances have a greater spread in pack sizes.

The PSH Flag Count histogram looks like it's basically just showing us whether the Push (or PSH) flag is there or not in each packet. And it turns out that in most cases, the flag's either there or it's not. So we see these two big clusters around zero and one.

The ACK Flag Count histogram also looks like it has a binary distribution, just like the PSH Flag Count. It's got lots of numbers around -0.5 and 1, which basically shows if there's an ACK (Acknowledgment) flag in the network packets or not.

The Average Packet Size histogram shows a right-skewed distribution, with most values concentrated towards the lower end. This indicates that the average packet sizes are generally small, with a few instances of larger average packet sizes. The distribution suggests that smaller packets are more common in the dataset.

The Avg Bwd Segment Size histogram displays a distribution similar to Bwd Packet Length Mean, with most values concentrated near zero and a long tail extending towards higher values. This suggests that the average backward segment sizes are generally small, with some instances of larger segment sizes.

Finally, the histogram for Init_Win_bytes_forward shows that most values are close to zero, with just a few numbers that are bigger. This right-skewed distribution means that in most cases, the initial window sizes are small, and just sometimes they're a bit bigger.

In summary, Figure 7 illustrates the distribution of various network traffic features within the dataset. Most features exhibit right-skewed distributions, indicating the presence of a large number of small values with a few larger outliers. Binary features, such as PSH Flag Count and ACK Flag Count, show distinct clusters, representing the presence or absence of specific flags.

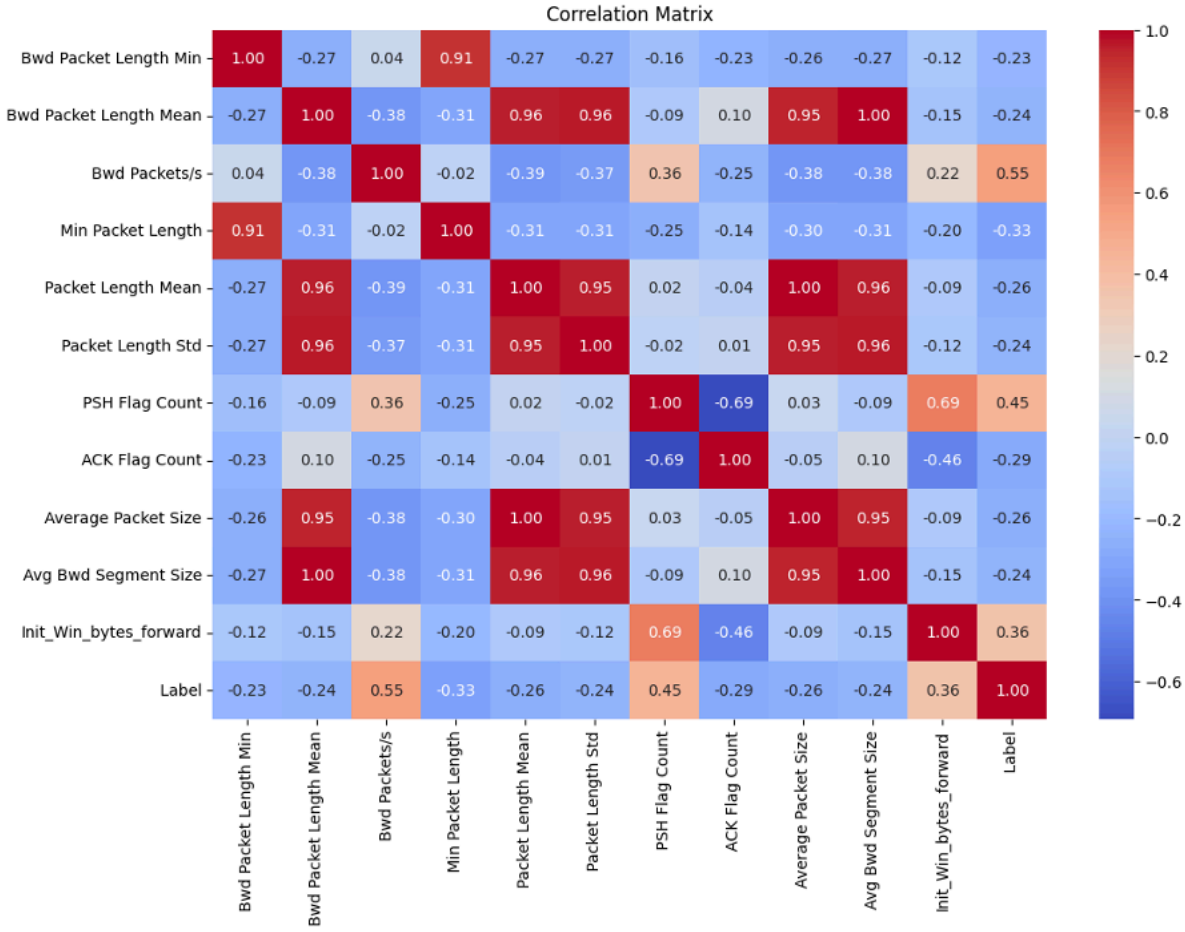


Figure 8 - Correlation Matrix for Features

The correlation matrix highlights several noteworthy relationships between the features in the dataset. Particularly strong positive correlations include a value of 0.96 between Bwd Packet Length Mean and Avg Bwd Segment Size, indicating a strong, direct connection between these measurements. Additionally, Packet Length Mean and Packet

Length Std exhibit a correlation of 0.95, indicating a similar degree of association. Finally, a correlation of 0.95 exists between Packet Length Mean and Average Packet Size, suggesting a measure of redundancy between these features.

Moderate correlations, such as Backward Packets/s and Label (0.55), suggest that the rate of backward packets per second is somewhat related to the target variable. A correlation of 0.45 between PSH Flag Count and Label indicates a moderate relationship, possibly indicating certain network behaviors associated with different labels. Furthermore, Init_Win_bytes_forward and Label exhibit a correlation of 0.36, implying some association between the two variables.

Slight but noteworthy correlations involve the parameters, ACK Flag Count and Init_Win_bytes_forward (0. 46), that shows a moderate relationship between acknowledgment flags and initial size of the window. A negative correlation occurs, for example, between the Min Packet Length and the Label (-0.33), implying that the smaller minimum packet length is somewhat correlated with specified labels.

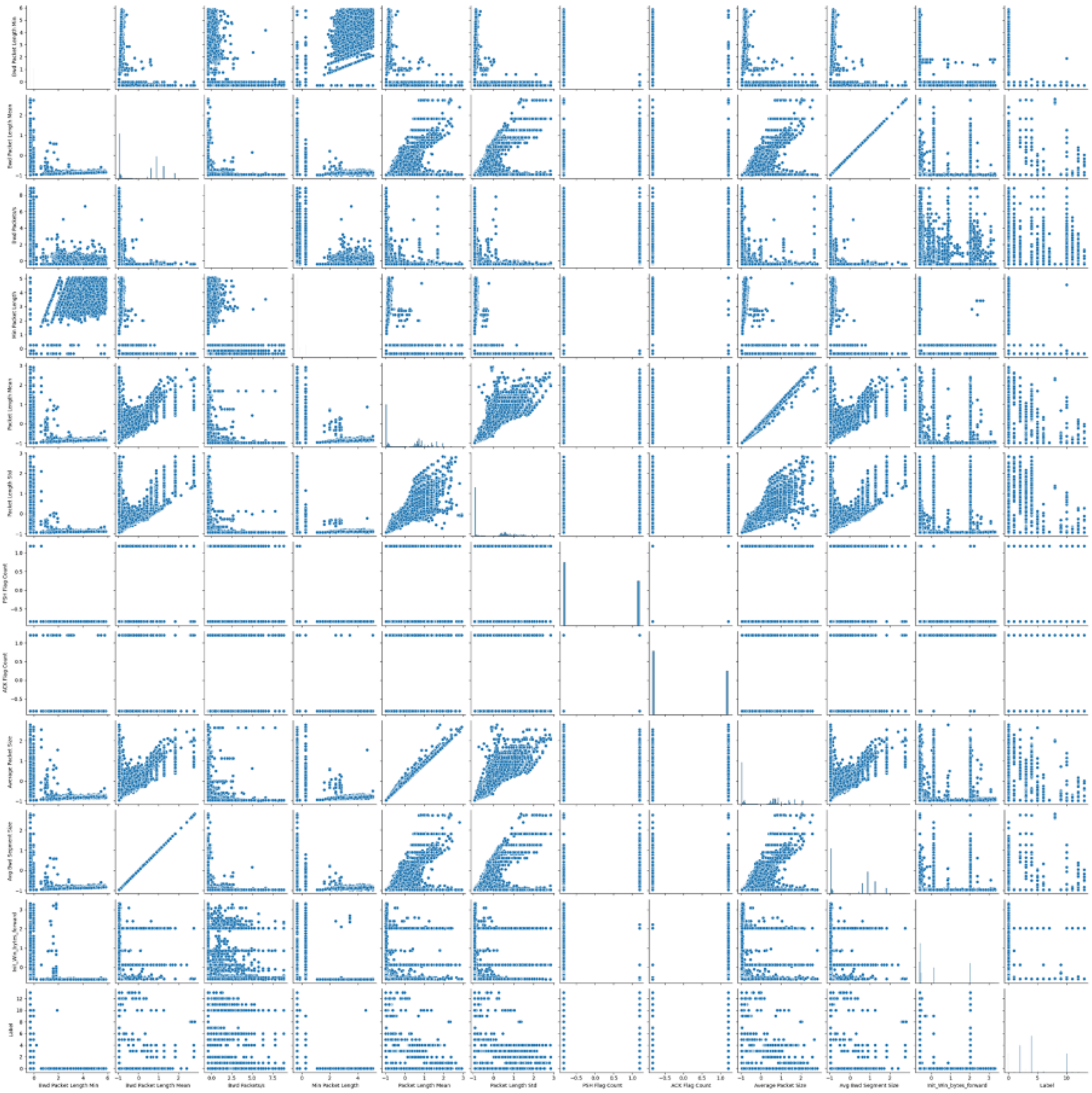


Figure 9 - Scatterplot for Relationships Between Features

Figure 9 provides a comprehensive overview of the relationships between various features in the dataset, highlighting several pairs with near-perfect linear relationships and others with weak or no correlation. Here, we delve deeper into the pairs exhibiting strong linear correlations and those with the weakest relationships.

One of the most noticeable things about the data is how closely related Backward Packet Length Mean and Average Backward Segment Size are. When you look at a scatterplot of the two, they form a nearly perfect straight line, showing a really strong positive correlation (r = 0.96). This strong relationship makes sense because both features are about the size of packets going backward. So, when the average length of backward packets goes up, the average size of the parts that make them up also goes up, kind of mirroring each other.

In the same way, the mean Packet Length and the standard deviation of Packet Lengths are strongly related, forming a straight line with a positive slope (r = 0.95). This means that when the average packet size gets bigger, the variety of packet sizes inside those instances also increases. As the average gets higher, the spread gets wider.

The relationship between Packet Length Mean and Average Packet Size is another example of a near-perfect linear correlation. The scatterplot demonstrates a strong positive correlation, as both features essentially measure similar aspects of packet size. The average packet size increases in tandem with the mean packet length, resulting in a linear relationship. Min Packet Length and Bwd Packet Length Min, are closely related. They have a really strong positive linear relationship, with a correlation coefficient of 0.91.

In contrast, several feature pairs display weak or no correlation, characterized by scattered plots with no discernible pattern. For example, the scatterplot between PSH Flag Count and Bwd Packets/s is widely dispersed, indicating no significant relationship. The number of PSH flags in the packets does not seem to affect the rate of backward packets per second, suggesting that these features capture independent aspects of network traffic.

The relationship between ACK Flag Count and Min Packet Length also shows a scattered distribution with no apparent correlation. This lack of a clear pattern suggests that the number of acknowledgement flags in the packets is unrelated to the minimum packet length, indicating that these features do not influence each other.

Similarly, Init_Win_bytes_forward and Bwd Packets/s show a dispersed pattern with no evident correlation. The initial window size of bytes moving forward does not appear to impact the rate of backward packets, highlighting the independence of these two features. The scatterplot between Avg Bwd Segment Size and Bwd Packets/s shows a slightly positive trend but is not as strong as the previous linear relationships. While there is a weak correlation, it is not as pronounced, indicating some level of relationship but with significant variability.

Overall, the pairplot reveals critical insights into the dataset's feature relationships. The near-perfect linear relationships, such as those between Bwd Packet Length Mean and Avg Bwd Segment Size, Packet Length Mean and Packet Length Std, and Packet Length Mean and Average Packet Size, indicate strong correlations and similar behavior patterns. On the other hand, the scattered plots with no discernible patterns, such as those between PSH Flag Count and Bwd Packets/s, ACK Flag Count and Min Packet Length, and Init_Win_bytes_forward and Bwd Packets/s, highlight weak or no correlation, indicating independent behavior between these features. Understanding these relationships is crucial for effective feature selection and engineering in the modeling process, ensuring robust and accurate machine-learning models.

d) Data Mining Technique Selection and Application

In order to analyze the dataset and determine the activities to be examined in the future by using the model we have trained, it was decided to proceed with the classification technique, which is one of the data mining techniques. The main reason why the classification technique was chosen over other techniques is that the CIC-IDS2017 dataset is already labeled. The labels in the dataset indicate whether a data item is a benign activity or a specific type of attack. Classification algorithms are best suited for creating models that learn from this labeled data and predict the label of new, unseen samples.

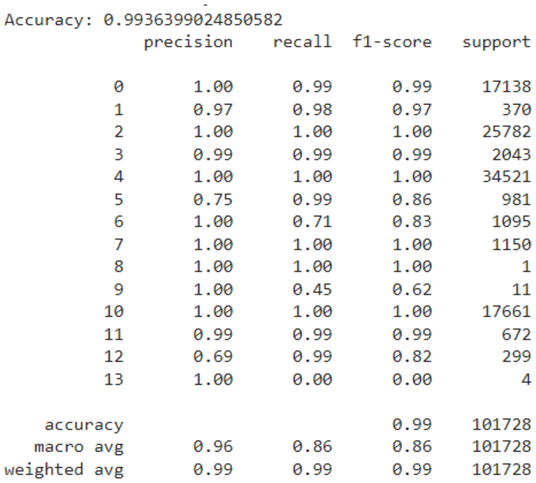
From the point of view of cybersecurity, this choice is of great importance because, thanks to the classification models we have trained, we can precisely identify certain types of attacks and detect them in real-time. This, in turn, increases our ability to respond quickly and effectively to attacks. Expected outcomes include accurate intrusion detection, reduction of false positives, and improvement of the overall security posture.

Overall, the use of the classification technique for the analysis of the CIC-IDS2017 dataset, which is within the scope of the project, has the potential to strengthen cybersecurity defenses, increasing our capacity to quickly detect attacks, and take appropriate measures.

After the data set is divided into X and Y(Y being the target “Label” values), 20 percent of the data is reserved for testing. The part reserved for Train was used for training 4 different models, and these models were tested on the test set and the score metrics of the models were obtained. The four different models used in this project are RandomForestClassifier, Logistic Regression, GradientBoostingClassifier, and KNeighborsClassifier models.

Within the scope of this project, different score metrics were used to test the score of the models and to decide on the best model. These score metrics are precision, recall, f1-score, support, and accuracy. Each metric measures the performance of the model from different angles. Precision shows the proportion of true positives in the model, and recall shows how well it detects true positives. F1-score is a balanced combination of precision and recall and is especially important in scoring unbalanced data sets. Accuracy indicates the overall accuracy of the model, while support refers to how much each class is represented in the data set. These metrics identify the strengths and weaknesses of the model, allowing us to evaluate its performance comprehensively.

In the first trial, the RandomForestClassifier model was used. Random Forest model consists of multiple decision trees. Each tree is trained with a random subset of the dataset, and the final decision is made based on the majority decision of all the trees. The accuracy, precision, recall, and f1-score metrics for every different label class are calculated and given below in Figure 10.



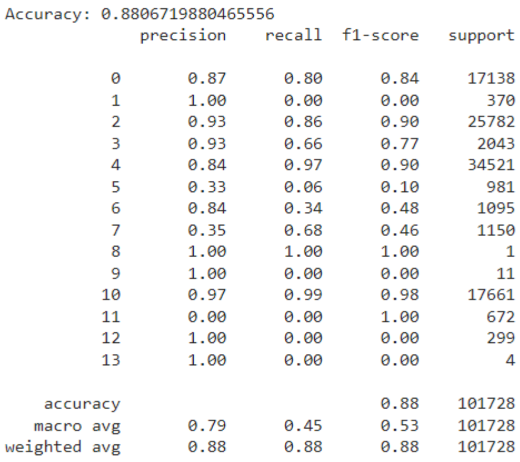
```
Accuracy: 0.9936399024850582
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     17138
           1       0.97      0.98      0.97       370
           2       1.00      1.00      1.00     25782
           3       0.99      0.99      0.99      2043
           4       1.00      1.00      1.00     34521
           5       0.75      0.99      0.86       981
           6       1.00      0.71      0.83      1095
           7       1.00      1.00      1.00      1150
           8       1.00      1.00      1.00         1
           9       1.00      0.45      0.62        11
          10       1.00      1.00      1.00     17661
          11       0.99      0.99      0.99       672
          12       0.69      0.99      0.82       299
          13       1.00      0.00      0.00         4

    accuracy                           0.99    101728
   macro avg       0.96      0.86      0.86    101728
weighted avg       0.99      0.99      0.99    101728
```

Figure 10: The accuracy, precision, recall, and f1-score metrics for every different label class, RandomForestClassifier Model

The second model used for training is the Logistic Regression model. This model is a statistical model used specifically for binary classification problems. It calculates the probability of an event and classifies it according to this probability. The results of each label category of the second model is included in Figure 11 below:



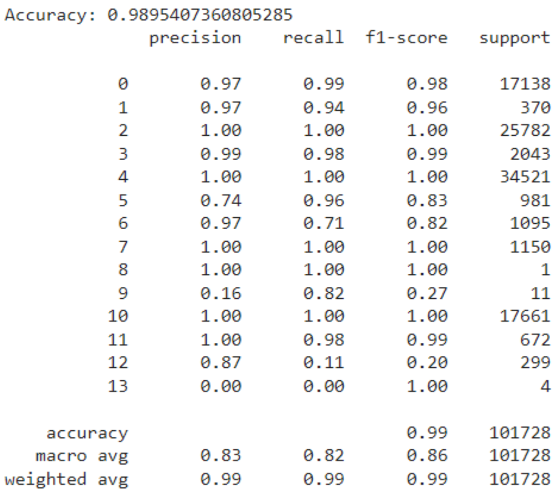
```
Accuracy: 0.8806719880465556
              precision    recall  f1-score   support

           0       0.87      0.80      0.84     17138
           1       1.00      0.00      0.00       370
           2       0.93      0.86      0.90     25782
           3       0.93      0.66      0.77      2043
           4       0.84      0.97      0.90     34521
           5       0.33      0.06      0.10       981
           6       0.84      0.34      0.48      1095
           7       0.35      0.68      0.46      1150
           8       1.00      1.00      1.00         1
           9       1.00      0.00      0.00        11
          10       0.97      0.99      0.98     17661
          11       0.00      0.00      1.00       672
          12       1.00      0.00      0.00       299
          13       1.00      0.00      0.00         4

    accuracy                           0.88    101728
   macro avg       0.79      0.45      0.53    101728
weighted avg       0.88      0.88      0.88    101728
```

Figure 11: Accuracy, precision, recall, and f1-score metrics for every different label class, LogisticRegression Model

For the third model, Gradient Boosting Classifier model was tried on the dataset. This model works by reducing errors by creating successive decision trees, and each tree tries to correct the mistakes of the previous trees. The results obtained with this model are shown with the Figure 12 below:



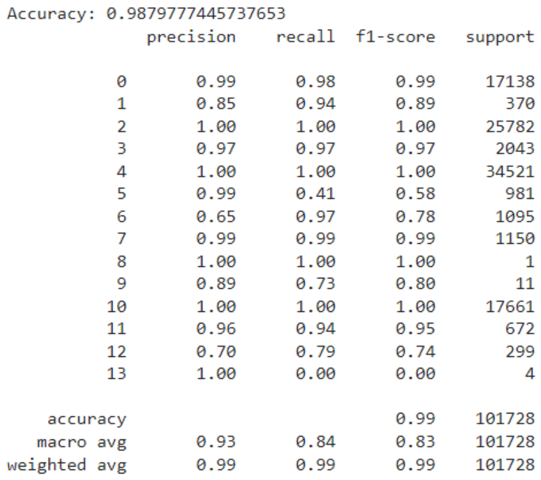
```
Accuracy: 0.9895407360805285
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     17138
           1       0.97      0.94      0.96       370
           2       1.00      1.00      1.00     25782
           3       0.99      0.98      0.99      2043
           4       1.00      1.00      1.00     34521
           5       0.74      0.96      0.83       981
           6       0.97      0.71      0.82      1095
           7       1.00      1.00      1.00      1150
           8       1.00      1.00      1.00         1
           9       0.16      0.82      0.27        11
          10       1.00      1.00      1.00     17661
          11       1.00      0.98      0.99       672
          12       0.87      0.11      0.20       299
          13       0.00      0.00      1.00         4

    accuracy                           0.99    101728
   macro avg       0.83      0.82      0.86    101728
weighted avg       0.99      0.99      0.99    101728
```

Figure 12: Accuracy, precision, recall, and f1-score metrics for every different label class, GradientDescentClassifier Model

The last model used is the KNeighborsClassifier model, and this model uses the k nearest neighbors when classifying new data points. The results are included below in Figure 13.



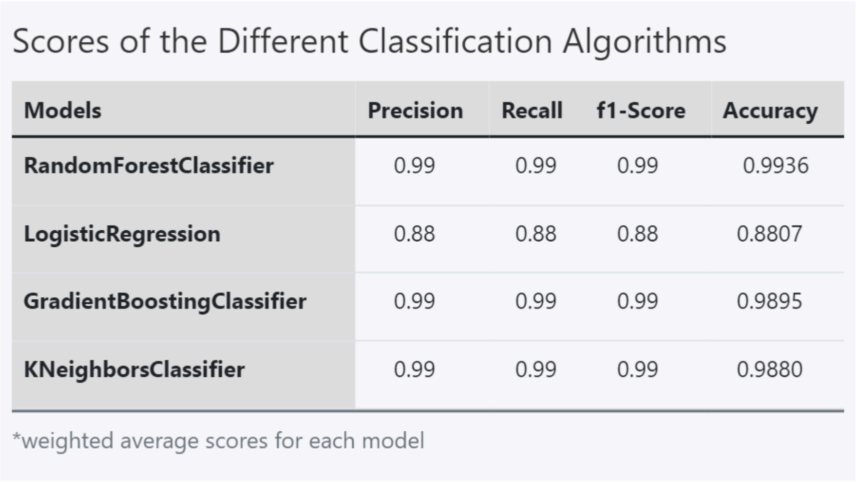
```
Accuracy: 0.9879777445737653
              precision    recall  f1-score   support

           0       0.99      0.98      0.99     17138
           1       0.85      0.94      0.89       370
           2       1.00      1.00      1.00     25782
           3       0.97      0.97      0.97      2043
           4       1.00      1.00      1.00     34521
           5       0.99      0.41      0.58       981
           6       0.65      0.97      0.78      1095
           7       0.99      0.99      0.99      1150
           8       1.00      1.00      1.00         1
           9       0.89      0.73      0.80        11
          10       1.00      1.00      1.00     17661
          11       0.96      0.94      0.95       672
          12       0.70      0.79      0.74       299
          13       1.00      0.00      0.00         4

    accuracy                           0.99    101728
   macro avg       0.93      0.84      0.83    101728
weighted avg       0.99      0.99      0.99    101728
```

Figure 13: Accuracy, precision, recall, and f1-score metrics for every different label class, KNeighborsClassifier Model

After examining the scores for each label class separately, it is also relevant to check all the scores with the weighted averages. All the weighted scores of all four different models were transferred into a table, making a visible comprehension possible by looking at the scores in Figure 14.



Scores of the Different Classification Algorithms

Models	Precision	Recall	f1-Score	Accuracy
RandomForestClassifier	0.99	0.99	0.99	0.9936
LogisticRegression	0.88	0.88	0.88	0.8807
GradientBoostingClassifier	0.99	0.99	0.99	0.9895
KNeighborsClassifier	0.99	0.99	0.99	0.9880

*weighted average scores for each model

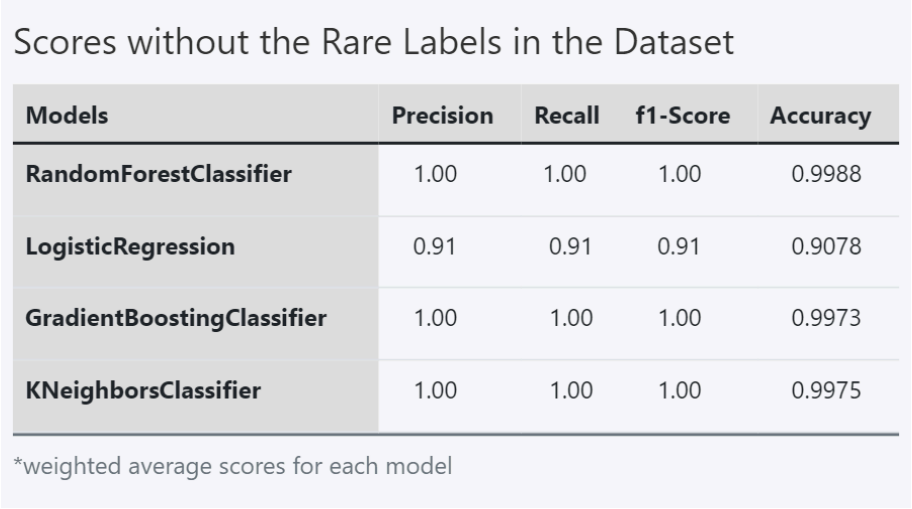
Figure 14: Weighted Averages of Score Metrics of 4 Models

According to the figure, it is visible that the best algorithm is the RandomForestClassifier with precision, recall, f1-score, and accuracy having the highest scores, indicating the model’s performance.

In addition to the model training in which all the labels are used above, a last last step has been tried to be used in systems and situations where it is more important to detect the large numbered labels when they have more importance for detection in terms of maintaining security. In cases where there is not enough data for the model to learn rare classes correctly, removing these classes can contribute to healthier results. Below are the score metrics calculated by a weighted average of the 4 models tested using the remaining 5 label classes.

After training and testing the models, the same 4 different models were trained and their scores were examined again by removing the rare class labels to ensure that the data could be predicted with better performance in cases where some class labels were much more numerous than others and it was more important to detect the large number of classes. In this step, labels with more than 10000 samples were added, and the training and testing were continued with only 5 of the labels.

Below is Figure 15 which shows the score metrics calculated by the weighted average of the 4 models tested using the remaining 5 label classes.



Scores without the Rare Labels in the Dataset

Models	Precision	Recall	f1-Score	Accuracy
RandomForestClassifier	1.00	1.00	1.00	0.9988
LogisticRegression	0.91	0.91	0.91	0.9078
GradientBoostingClassifier	1.00	1.00	1.00	0.9973
KNeighborsClassifier	1.00	1.00	1.00	0.9975

*weighted average scores for each model

Figure 15: Weighted Averages of Score Metrics with the Removal of Rare Labels in the Dataset

It has been noted that many cyberthreat kinds have been identified as a consequence of the analyses done utilizing data mining techniques. Specifically, malware, DDoS assaults, SQL injections, and other forms of attacks have been identified with the trained models. With the analysis conducted on the dataset, the findings provide the conclusion that these dangers have the potential to seriously jeopardize an organization's data security, network infrastructure, and information systems. As a result, data mining techniques are essential for identifying and neutralizing cyber threats early on.

It's critical to assess the efficiency and accuracy of the studies to understand the degree to which data mining techniques are useful in the realm of cyber security as well as identify any areas that require further development. In the scope of the project, after training several models and comparing their precision, recall, f1-score, and accuracy score metrics, it is apparent that the best model for the detection is RandomForestRegression model. Having an accuracy of 0.99, the model is successful in detecting 99 percent of attacks correctly. With this study, the creation of better defensive plans against potential threats will be increased.

e) Conclusion

Within the scope of this assignment, it is aimed to detect and classify cyber security threats by applying various data mining techniques on the CIC-IDS2017 dataset. First, it is aimed to understand the data set. The preparation of the data set for application to the models was provided by preprocessing steps such as reducing the NaN values, eliminating the multiplexing data, downsampling for the equal distribution of label classes, digitizing the label column for the use of the model, and making correlation and feature selection.

The Exploratory Data Analysis (EDA) covers everything about the dataset, which provides critical insights for effective development of a machine learning model. It explores key patterns and relationships in the data, including the distribution of network traffic features and the target variable.The discovery of an unusual imbalance in label distribution is important because it shows that some types of network traffic are more likely to appear on a network than others. This significant imbalance poses hurdles for model performance since machine learning algorithms can make models biased towards majority classes, hence resulting into poorer performance on minority classes.

The skewed nature of diverse features, with many values clustered close to 0 and some large outliers, suggests that most network packets are small, with occasional larger packets. This sample is consistent across numerous capabilities, which include Bwd Packet Length Min and Bwd Packet Length Mean. Binary functions like PSH Flag Count and ACK Flag Count exhibit wonderful clusters, representing the presence or absence of precise flags in the network traffic.

The correlation analysis famous strong high quality correlations among sure characteristic pairs, indicating redundancy and capacity opportunities for characteristic reduction to simplify the model. Conversely, weak or no correlations among other capabilities endorse that they seize unbiased aspects of network traffic that can contribute to an extra comprehensive expertise of the facts.

The EDA gives us important information about the datasets structure and connections, helping us prepare the data and choose the right features for our machine learning models. To make sure our models work well, we need to fix any problems with the labels, understand how the features are spread out, and use the features that are related to each other. These insights ensure that the models are well-equipped to handle the diverse and complex nature of network traffic data, ultimately leading to better performance and more reliable predictions.

Then, threats were detected using classification models such as RandomForest, Logistic Regression, Gradient Boosting, and K-Nearest Neighbors (KNN). After the model training and testing steps, the performance metrics such as accuracy, precision, recall rate, and F1 score of the models were calculated and examined. With the examination of these score metrics, it was observed that the RandomForest model achieved the highest accuracy rate of 99.36% With this project, it is apparent that the use of these trained prediction models has enabled the effective detection of various cyber threats, especially DoS attacks, malware, and SQL injections. The results of this study shows that data mining techniques play a critical role in the field of cyber security and that these techniques increase the early detection and response-prevention capabilities of the organizations against cyber threats. This study has revealed that with the usage of correct and effective data mining techniques, cyber security strategies can be strengthened and more prevention for future threats can be achieved.

f) References

Aditya Pratap Singh , Navneet Kaur . Introduction To Data Preprocessing: A Review. *TechRxiv.* September 12, 2022.

Bhaya, Wesam. (2017). Review of Data Preprocessing Techniques in Data Mining. *Journal of Engineering and Applied Sciences.* 12. 4102-4107. 10.3923/jeasci.2017.4102.4107.

Majidi, Noorollah & Tajmir Riahi, Hossein & Zandi, Mahdi & Hajirasouliha, Iman. (2023). Development of practical downsampling methods for nonlinear time history analysis of complex structures. *Soil Dynamics and Earthquake Engineering*. 175. 108247. 10.1016/j.soildyn.2023.108247.

Potdar, Kedar & Pardawala, Taher & Pai, Chinmay. (2017). A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. *International Journal of Computer Applications.* 175. 7-9. 10.5120/ijca2017915495.

Senthilnathan, Samithamby. (2019). Usefulness of Correlation Analysis. *SSRN Electronic Journal*. 10.2139/ssrn.3416918.

https://www.kaggle.com/code/sauravbisht0129/attacks-prediction

https://www.kaggle.com/code/seveensamir/aisecurityissues-project

https://www.kaggle.com/datasets/cicdataset/cicids2017/data