Politecnico di Torino

Politecnico di Torino
Department of Control and Computer Engineering

Group Project – Machine learning for networking

DDoS attacks detection and characterization (Nr. 5)

Group Nr. 09

Aresca Massimo – Mat. 328743
Carcagnì Andrea – Mat. 331618
de Maria Giovanni – Mat. 331031
Rizzo Leonardo – Mat. 328764

Accademic Year 2023-2024

Contents

Section 0 Introduction

0.1 DDoS attacks

In the cyber threat landscape, Distributed Denial of Service (DDoS) attacks are distinguished by their ability to cripple network services and systems, denying access to legitimate users. These attacks exploit vulnerabilities inherent in the protocols underlying the World Wide Web, flooding targets with a disproportionate amount of traffic from multiple compromised devices, often thousands.

0.2 Project Objectives

The primary objective of this project is to develop a machine learning model for network traffic classification with the intent to distinguish between benign traffic and DDoS (Distributed Denial of Service) attacks.

To achieve this goal, an approach based on machine learning techniques will be adopted. The model will be trained using a dataset provided by the instructors, which includes a variety of packet flow samples. This dataset has been specifically curated to encompass both legitimate traffic and data related to various DDoS attack scenarios, thereby ensuring an accurate and balanced representation of the possible situations the model will encounter in a real context.

0.3 Project Structure

The project is divided into a series of comprehensive sections, each addressing a key component of the machine learning process for classifying network traffic. For organizational purposes, the code for each section is contained within a single notebook.

1. **Data Exploration and Pre-processing:** Employ statistical and visualization techniques to understand the data. Normalize features and remove highly correlated ones to enhance model performance and address the curse of dimensionality.
2. **Supervised Learning – Classification:** Involves training and evaluating various machine learning models on the dataset to identify the most effective approach for detecting and classifying flows, including performance analysis and hyperparameter tuning.
3. **Unsupervised Learning – Clustering:** Explores clustering algorithms to identify patterns in network data, emphasizing the discovery of natural groupings without labeled data.
4. **Clusters Explainability and Analysis:** Analyzes the identified clusters, examining their features and distributions to draw conclusions about network traffic characteristics and cybersecurity threats.

Section 1 Data exploration and pre-processing

Explore the Dataset The initial and crucial step we undertook was a visual analysis of the database. Using a simple online viewer to open the .csv file allowed us to navigate more efficiently. In particular, this approach allowed us to quickly sort the database by clicking on the column headers of the features we wished to sort.

From this initial exploration, we immediately assigned a definition to the term "flow," understanding it as an exchange of data between two devices during which packets are transferred bidirectionally. A closer examination of the flow ID suggests that it is composed of a concatenation of Source IP, Source Port, Destination IP, Destination Port, and Protocol as outlined in the design specifications provided by our professors.

Examining the features, it was noted that many were expanded during the data collection process as indicated by labels with added prefixes such as total, average, standard deviation, maximum, and minimum. This organization suggests the presence of some correlation among these features.

In addition, we observed that some fields consistently had a value of 0, thus bringing no meaningful information to our analysis.

After this preliminary overview, we proceeded to load the database into our environment for a more detailed examination.

1.1 Clean the Dataset

Flow ID issue Once the dataset is loaded, we want to make sure that the flow ID is unique as we expect, so we count the occurrences of the flow ID.

Contrary to what we expected, the flow ID is not unique; performing further analysis (presented in Notebook 1) showed a discrepancy in the structure of some flow IDs. We interpreted this error as a data quality problem and decided to delete the flow ID and recreate one with the same pattern: Source IP - Destination IP - Source Port - Destination Port - Protocol.

Even this was not enough to make the flow ID a true unique ID, so we added an additional feature: the timestamp. Now the flow ID is unique and can be set as an index of the dataframe.

No variance features While exploring the database, we noticed that some features had a value of 0. Therefore, we decided to eliminate all features that do not add information and thus have zero variance.

1.2 Visualizations and Statistical Analysis

In this section, the goal is to visualize using various methodologies the data to highlight interesting patterns that may lead to the creation of new features and to get a general picture of the data processed: flows, ports used, IP affected, and more. The section is divided in turn into general traffic investigation and investigation with reference to ground truth.

1.2.1 Generic traffic level

In this part, we analyze the generic network traffic without being aware of the label.

IP distribution To accomplish this, we extract the most frequent source and destination IP addresses from our data. These data are shown in the figures 1.1 and 1.2

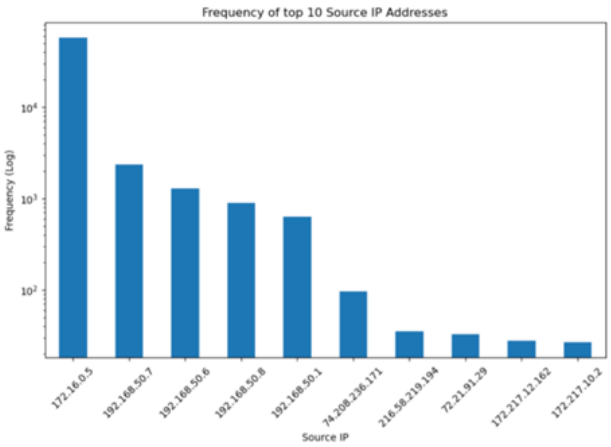


Figure 1.1: Frequency of top 10 Source IP Addresses

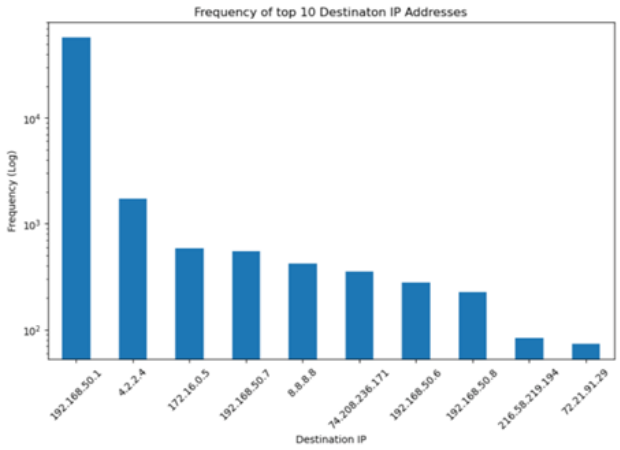


Figure 1.2: Frequency of top 10 Destination IP Addresses

Despite the y-axis being in a logarithmic scale, there are two particular addresses (one source and one destination) that appear with a significantly higher frequency compared to all others.

Additionally, it is noteworthy that many IP addresses appear both as destinations and sources. Therefore, we can consider merging the two graphs. To achieve this, we combine all the IP addresses into a single set and count the frequency of each address in both the destination IP and source IP fields.

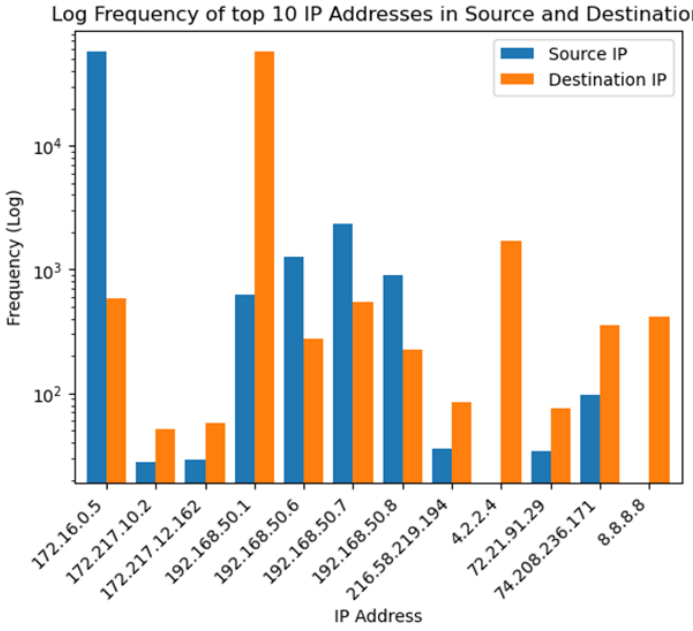


Figure 1.3: Log Frequency of top 10 IP Addresses in Source and Destination

The figure 1.3 is quite interesting as it highlights the presence of certain specific IP addresses such as 8.8.8.8. A high frequency of requests originating from specific IP addresses that appear both as sources and destinations but with significantly different frequencies may suggest a botnet being used for a DDoS attack. However, some of these IP addresses might also be legitimate servers or network nodes with high traffic. For example, the address 8.8.8.8 appears in the flows only as a destination and never as a source. Despite the significant difference in frequency that might suggest the IP is being attacked, we know that 8.8.8.8 is Google's address. Therefore, it is normal for it to receive many requests but not send any.

Another interesting aspect highlighted by this graph concerns the two IP addresses with the highest frequency: 172.16.0.5 and 192.168.50.1. We observe a perfect symmetry: the source frequency of 172.16.0.5 is equal to the destination frequency of 192.168.50.1 and vice versa. This symmetry suggests that these two IP addresses might primarily communicate with each other.

Incoming and outgoing flows distribution If we want to understand the potential variations in incoming and outgoing flows of an IP address, we can observe the figure 1.4.

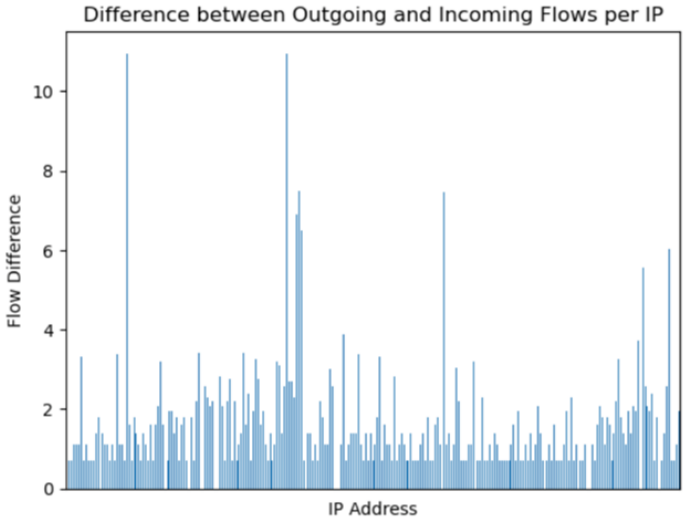


Figure 1.4: Difference between Outgoing and Incoming Flows per IP

Note: Our analysis will primarily focus on the 10 most significant IP addresses as it is challenging to visualize a characteristic graph with a large amount of data.

Flow duration distribution of IPs Let's shift our focus to the duration of the flow and see what insights can be derived.

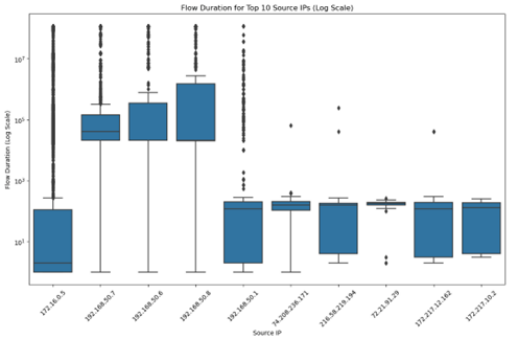


Figure 1.5: Flow Duration for Top 10 Source Ports

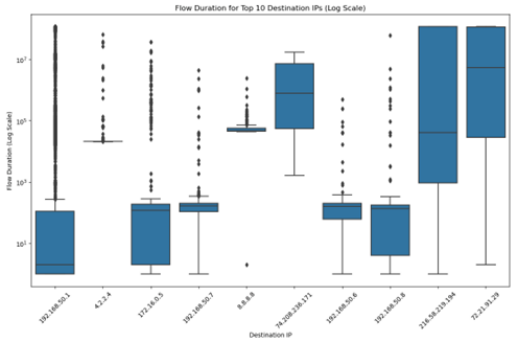


Figure 1.6: Flow Duration for Top 10 Destination Ports

From figures 1.5 and 1.6, we observe a striking similarity when comparing 172.16.0.5 as a source to 192.168.50.1 as a destination. The same similarity is noted between other pairs of addresses, for example, 74.208.236.171 and 192.168.50.7.

However, if we examine the box plot of the flow duration for an IP address by looking at flows where it appears as a source and those where it appears as a destination, we do not see significant similarities. For instance, examining 192.168.70.7 first as a source and then as a destination shows a significantly different box plot.

Port distribution Another crucial aspect we wanted to analyze, given its importance in these types of attacks, is the distribution of ports. The frequency of top 10 Source and Destination port are shown in the figures 1.7 and 1.8.

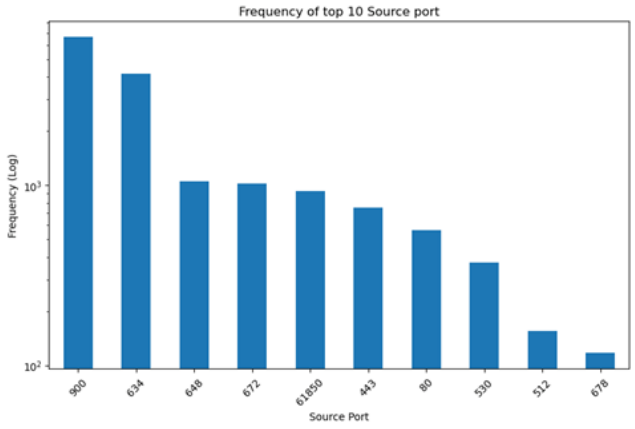


Figure 1.7: Frequency of top 10 Source port

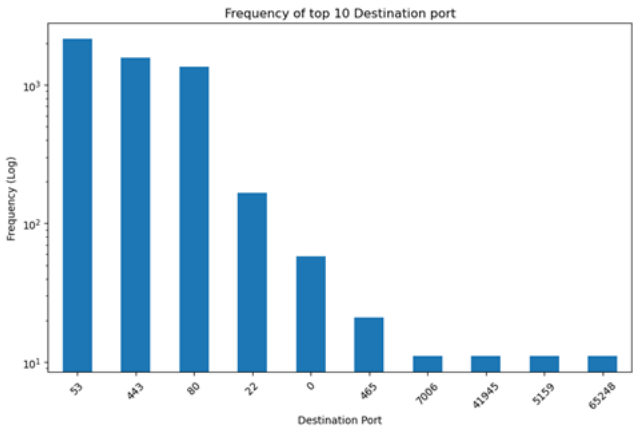


Figure 1.8: Frequency of top 10 Destination port

We considered it appropriate to conduct further investigations on the ports as they represent an interesting aspect. Therefore, similar to our analysis of IP addresses, we will analyze the flow duration for the ports.

Flow duration distribution of ports From the figure 1.9 and 1.10 is evident that source ports have less dispersion in flow duration compared to destination ports, indicating that the flow duration varies less for a particular source port.

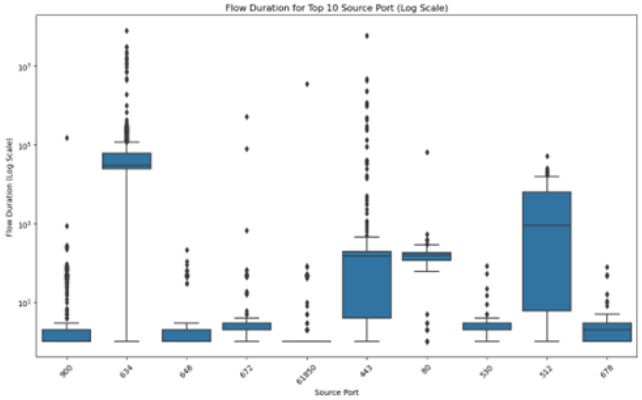


Figure 1.9: Flow Duration for Top 10 Source Port

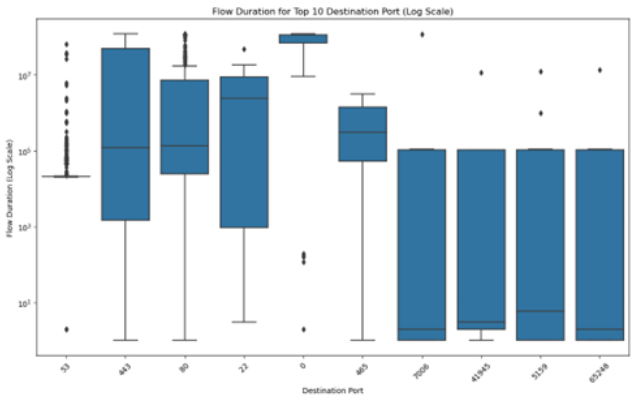


Figure 1.10: Flow Duration for Top 10 Destination Port

Timestamp distribution Given our domain knowledge and understanding that DDoS attacks are characterized by an enormous amount of data from different sources concentrated in a short time interval, we find it appropriate to thoroughly explore the temporal characteristics using timestamps, as shown in 1.11.

We consider this graph very interesting. In fact, having peaks where many flows occur in a short time can be an alarming signal. We will explore this characteristic further.

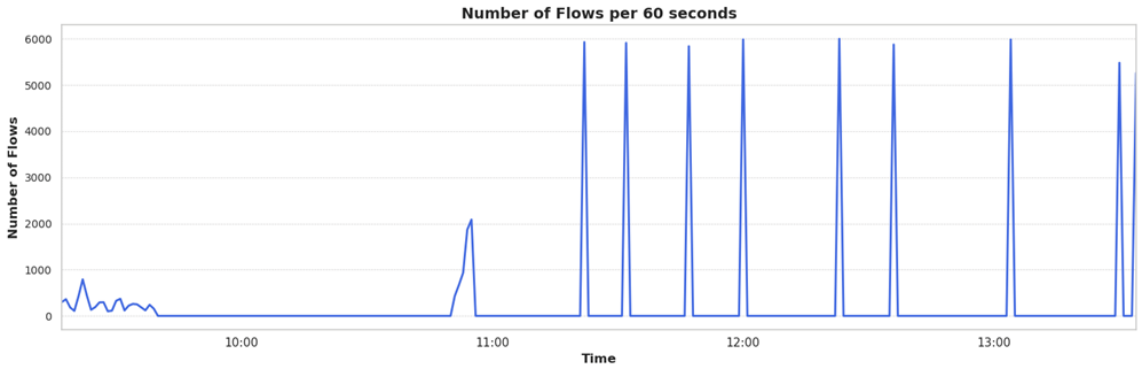


Figure 1.11: Number of Flows per 60 seconds

1.2.2 Ground truth level

We have obtained an overall view of our database and know what to expect. However, we have a crucial piece of information that allows us to conduct more in-depth analyses: the label ground truth. The objective of this second part of the section is to highlight additional characteristics that we can observe by using the label as a feature.

Label distribution Let's analyze the distribution of our labels to understand if the flows are evenly distributed. The result are shown on figure 1.12.

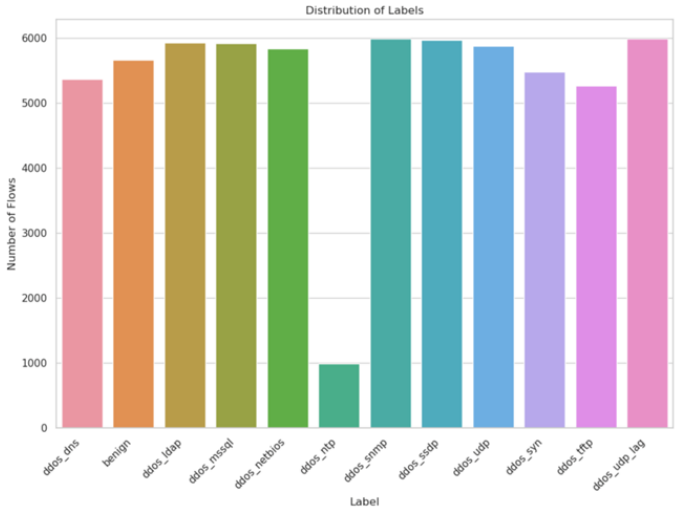


Figure 1.12: Distribution of Label

The dataset is almost homogeneous except for the attack class ddos_ntp which is underrepresented is no small detail and should be taken into account.

Port distribution As can be seen on figure 1.13, we found an interesting result: except for the ddos_ntp class which we remember is underrepresented and could lead to misleading results, benign flows tend to use smaller ports typically associated with basic protocols while attack flows use larger and more unusual ports.

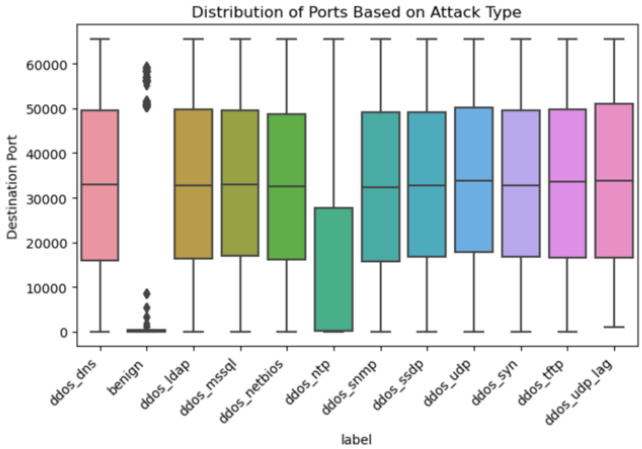


Figure 1.13: Distribution of Ports Based on Attack Type (Destination port)

Regarding the ports used by the sources, on figure 1.14 we do not observe a linear trend as we do for the destinations. However, there are some common characteristics among certain attack types such as ddos_ldap, ddos_mssl, ddos_snmp, and ddos_ssdp.

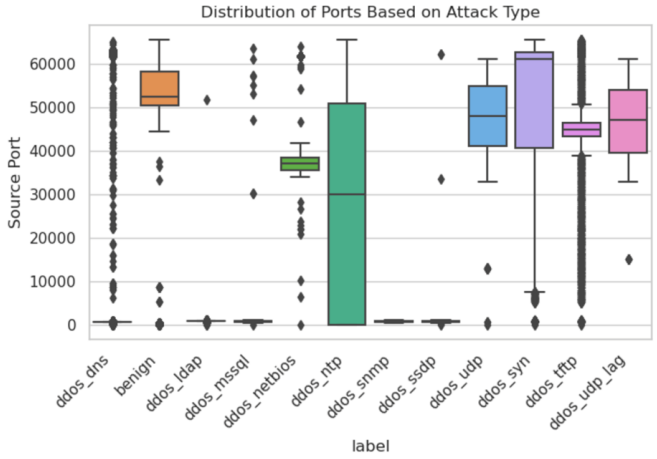


Figure 1.14: Distribution of Ports Based on Attack Type (Source port)

Flow duration distribution It is interesting to note the relationship between flow duration and packets per second; in fact, the two graphs appear to be mirror images of each other.

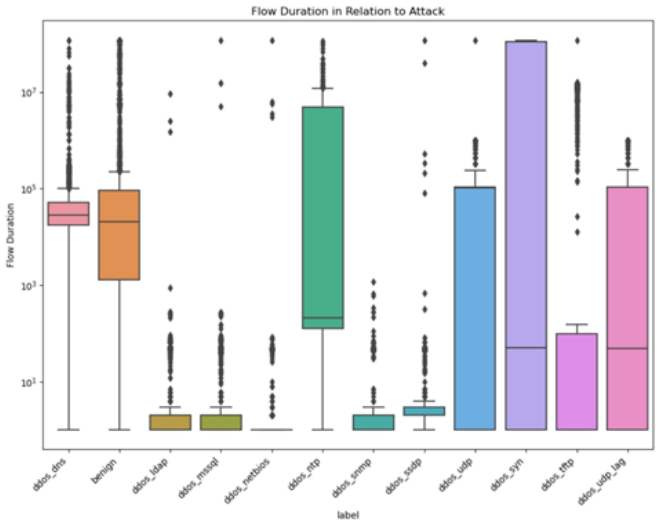


Figure 1.15: Flow Duration in Relation to Attack

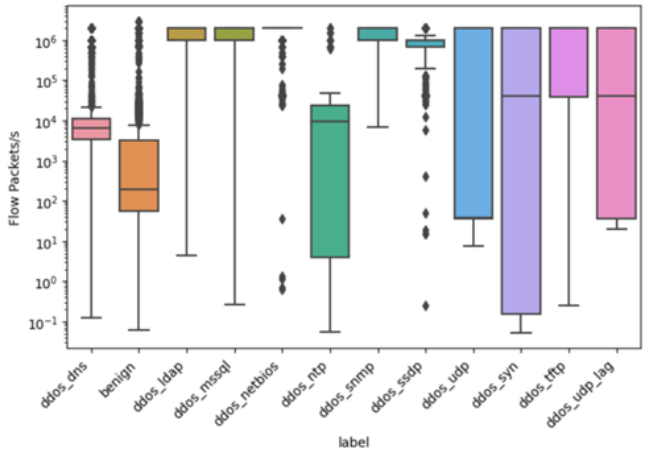


Figure 1.16: Flow Packets/s in Relation to Attack

The figures 1.15 and 1.16 provide an interesting perspective; not all labels assume the same value, although many share similar characteristics such as ddos_ldap, ddos_mssl, ddos_snmp, and ddos_ssdp. However, it should be noted that the graph is on a logarithmic scale and for certain labels like ddos_syn and ddos_ntp, there is a highly dispersed distribution.

Distribution of Packets Forwarded per Flow To overcome the limitations of the previous graphs, we will attempt to visualize the distribution of forwarded packets.

In the figure 1.17, we observe a very linear trend for all flows except for ddos_dns.

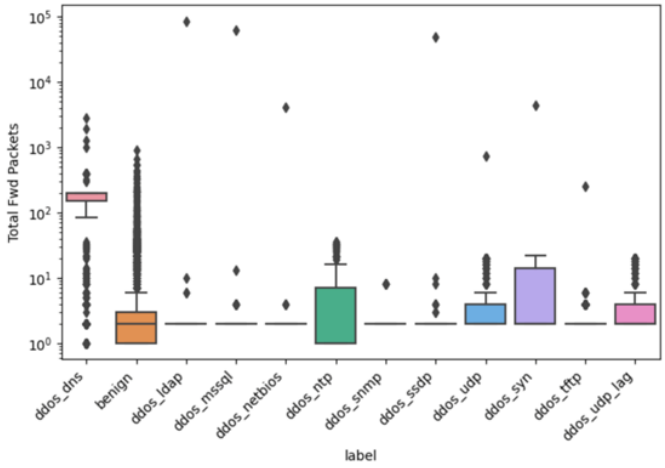


Figure 1.17: Total Fwd Packets in Relation to Attack

Timestamp distribution On figure 1.18 is shown a very important result.

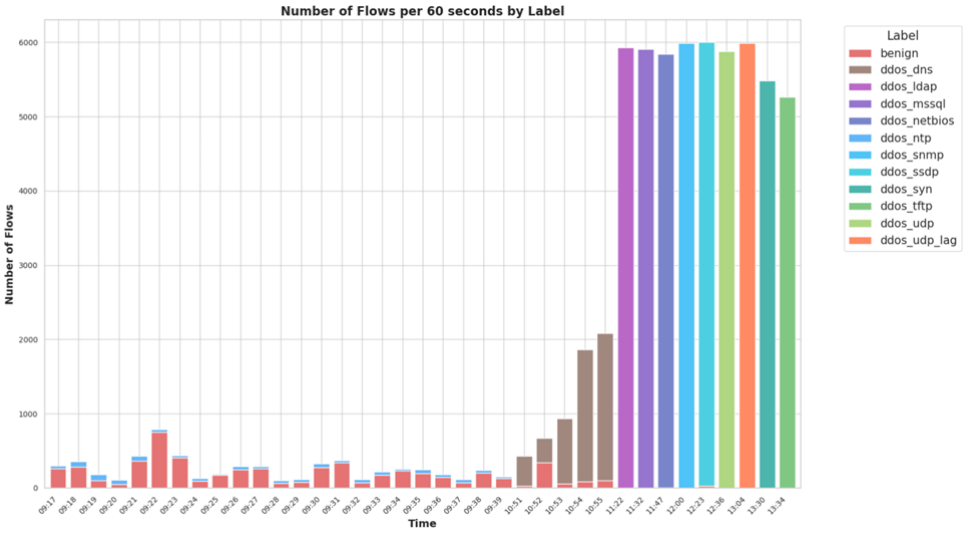


Figure 1.18: Number of Flows per 60 seconds by Label

At first glance, it is evident that with the exception of a few classes, the distribution of the others is concentrated in specific time slots.

Note: in the temporal distribution graph, we remember that for many time intervals there were no flows. In this latest graph with labels, these empty spaces have been removed because they hindered the readability of the graph itself. Indeed, various jumps between different dates can be noted.

At the beginning of our analysis, we were confident that the timestamp would provide us with crucial information and it might have been so if not for the temporal distribution of our data.

What we hoped to achieve with the timestamp was to calculate the intra-intervals between one flow and another to understand how quickly the flows occurred.

Unfortunately for us, this is not possible. Looking at the graph, it is evident that for certain periods of time there is a different type of attack. Using this information, the model would risk overfitting, learning that a certain type of attack occurs at a specific time slot which is not what we want to achieve. To demonstrate this, we illustrate the obtained results (the code of which is visible at the end of section 2) including the timestamp information.

These are the lines used to add the features for the experiment:

```
ddos_data['Timestamp'] = pd.to_datetime(ddos_data['Timestamp'])
ddos_data['hours'] = ddos_data['Timestamp'].dt.hour
```

```
ddos_data['minutes'] = ddos_data['Timestamp'].dt.minute
ddos_data['seconds'] = ddos_data['Timestamp'].dt.second
ddos_data['milliseconds'] = ddos_data['Timestamp'].dt.microsecond // 1000
```

On figures 1.19,1.20 we present the results obtained by training the database with a random forest after adding these temporal features.

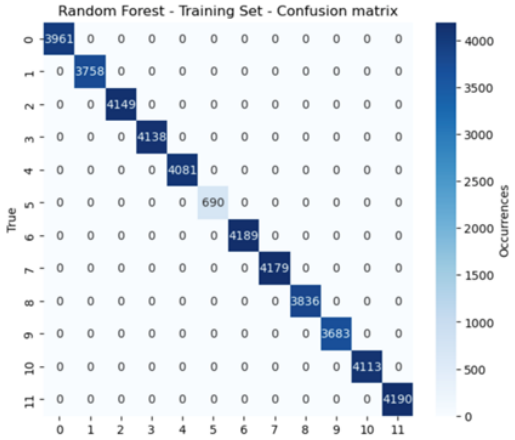


Figure 1.19: Random Forest – Training Set – Confusion matrix

Table 1.1: Random Forest – Training Set – Classification report

	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	3961
1	1.00	1.00	1.00	3758
2	1.00	1.00	1.00	4149
3	1.00	1.00	1.00	4138
4	1.00	1.00	1.00	4081
5	1.00	1.00	1.00	690
6	1.00	1.00	1.00	4189
7	1.00	1.00	1.00	4179
8	1.00	1.00	1.00	3836
9	1.00	1.00	1.00	3683
10	1.00	1.00	1.00	4113
11	1.00	1.00	1.00	4190
Accuracy		1.00 (44967)		
Macro avg	1.00	1.00	1.00	44967
Weighted avg	1.00	1.00	1.00	44967

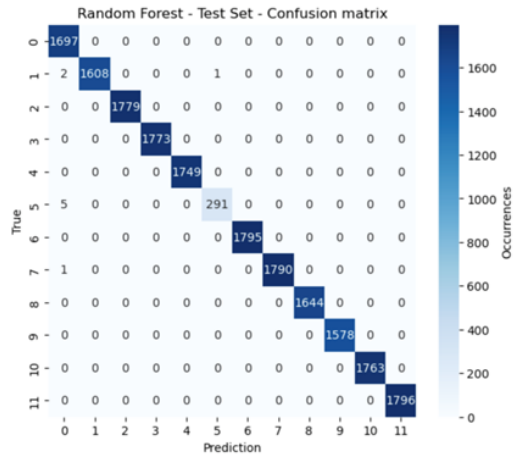


Figure 1.20: Random Forest – Test Set – Confusion matrix

Table 1.2: Random Forest – Test Set – Classification report

	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	1697
1	1.00	1.00	1.00	1611
2	1.00	1.00	1.00	1779
3	1.00	1.00	1.00	1773
4	1.00	1.00	1.00	1749
5	1.00	0.98	0.99	296
6	1.00	1.00	1.00	1795
7	1.00	1.00	1.00	1791
8	1.00	1.00	1.00	1644
9	1.00	1.00	1.00	1578
10	1.00	1.00	1.00	1763
11	1.00	1.00	1.00	1796
Accuracy		1.00 (19272)		
Macro avg	1.00	1.00	1.00	19272
Weighted avg	1.00	1.00	1.00	19272

As we anticipated, we encounter a situation of overfitting. As mentioned earlier, the reason for this lies in the fact that the attacks occur only within a specific time interval.

1.3 Feature Engineering

Following the exploration and visualization of the database conducted in the previous sections, we decided to expand some features within our database.

We have decided to remove the features of source IP and destination IP. This decision was based on the need to avoid overfitting. In fact, if a new IP were to start sending malicious traffic, it would likely be classified as benign because it is an IP that has never been seen before.

We have recognized the importance of ports and since they are numerical features without an inherent order of magnitude, we need to decide on the appropriate technique to incorporate these features considering that everything will need to be scaled. After careful analysis, we have decided to follow the frequency-based feature strategy, which involves using the frequency or distribution of the characteristics to transform categorical or numerical variables into a format that can be effectively used in machine learning models.

For all the reasons previously examined, we did not include the temporal features. Additionally, we deemed it important to highlight the protocol used. Therefore, utilizing one-hot encoding, we introduced features named Protocol 0, Protocol 6, Protocol 16, which are assigned a value of 0 if absent and a value of 1 if present.

1.4 Data Scaling or Standardization

Scaling or standardizing data is a fundamental preprocessing step that benefits various machine learning models by ensuring that each feature contributes equally to the analysis regardless of their original scale. This not only helps in the application of PCA but also enhances the overall performance and accuracy of many machine learning algorithms by providing a uniform scale for all features thus facilitating a more balanced and fair learning process.

1.5 Correlation Analysis and Dimensionality Reduction

In this section, we have decided to utilize correlation analysis, Principal Component Analysis (PCA), and loading scores to optimize the feature set of our dataset. These tools allow us to reduce the dimensionality of the data, eliminate redundancies, and retain the most relevant information, thereby enhancing the efficiency and interpretability of our model.

Correlation analysis was the first step in our process. This technique enabled us to identify the linear relationships between the different features of the dataset. We calculated the correlation matrix (see figure 1.21), which measures the strength and direction of the relationship between each pair of variables. This helped us identify highly correlated features. Variables with high correlation (greater than 0.9 or less than -0.9) are considered redundant, as they provide similar information. By eliminating these redundant features, we reduce the complexity of the model without losing significant information, thereby improving its ability to generalize to new data.

Subsequently, we applied Principal Component Analysis (PCA). PCA is a dimensionality reduction technique that transforms the original features into a new set of uncorrelated variables called principal components. These components are ordered so that the first captures the maximum possible variance in the data, followed by subsequent components that capture the remaining variance in decreasing order. Before applying PCA, as mentioned in the previous paragraph, we standardized the features to ensure they all have zero mean and unit variance, as PCA is sensitive to the scale of the variables.

Once the Principal Component Analysis (PCA) was performed, we examined the loading scores of the first three principal components, which explain 50% of the cumulative variance. The loading scores represent the contribution of each original variable to the principal components, allowing us to assess the impact of individual features. In other words, they indicate how much each original variable influences a particular principal component.

By cross-referencing the data obtained from the correlation analysis with those derived from the PCA loading scores, we assessed all the most redundant features for elimination.

Correlation analysis

Figure 1.21: Correlation matrix

We selected and removed the most correlated features using a correlation threshold of 0.85: for each pair of features with a correlation higher than this value, we eliminated one of the two features, making sure not to remove the same feature multiple times in case it was highly correlated with multiple other features. This process resulted in the selection of a unique set of highly correlated features, while retaining those with more independent information in our dataset (See figure 1.22).

```
34 features to be removed
['Active Max', 'Active Mean', 'Average Packet Size', 'Avg Bwd Segment Size', 'Avg Fwd Segment Size', 'Bwd IAT Std',
'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Duration', 'Flow IAT Max', 'Flow IAT Min', 'Flow IAT Std',
'Fwd Header Length', 'Fwd IAT Max', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Total', 'Fwd Packet Length Mean',
'Fwd Packet Length Min', 'Fwd Packets/s', 'Idle Max', 'Idle Mean', 'Min Packet Length', 'Packet Length Mean',
'Packet Length Std', 'Packet Length Variance', 'RST Flag Count', 'Subflow Bwd Bytes', 'Subflow Bwd Packets',
'Subflow Fwd Bytes', 'Subflow Fwd Packets', 'Total Backward Packets', 'Total Length of Bwd Packets',
'Total Length of Fwd Packets']
```

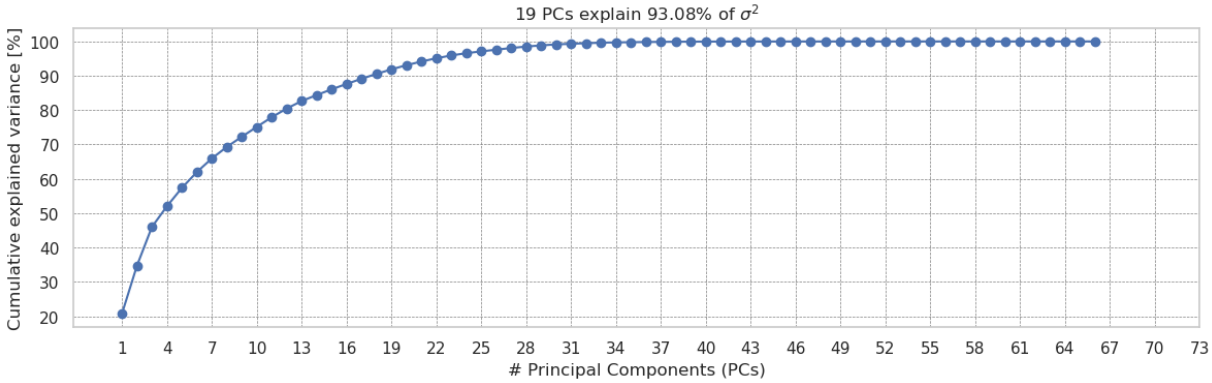


Figure 1.22: Cumulative Explained Variance vs. Number of Principal Components

Despite the significant reduction in dimensions, having still 19 components can make the visualization and interpretation of patterns complex, especially when using loading scores to identify the most influential features for each principal component.

Principal component analysis By manually cross-referencing the results of the correlation analysis and the results of the loading scores, we identified the following features to be removed (See figure 1.23).

```
36 features to be removed
['Active Max', 'Active Min', 'Average Packet Size', 'Avg Bwd Segment Size', 'Avg Fwd Segment Size',
'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Duration',
'Flow IAT Max', 'Flow IAT Min', 'Flow IAT Std', 'Flow Packets/s', 'Fwd Header Length', 'Fwd IAT Max',
'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Total', 'Fwd Packet Length Mean', 'Fwd Packet Length Min',
'Idle Max', 'Idle Mean', 'Min Packet Length', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance',
'Protocol 17', 'Protocol 6', 'Subflow Bwd Bytes', 'Subflow Bwd Packets', 'Subflow Fwd Bytes',
'Subflow Fwd Packets', 'Total Backward Packets', 'Total Length of Bwd Packets', 'Total Length of Fwd Packets']
```

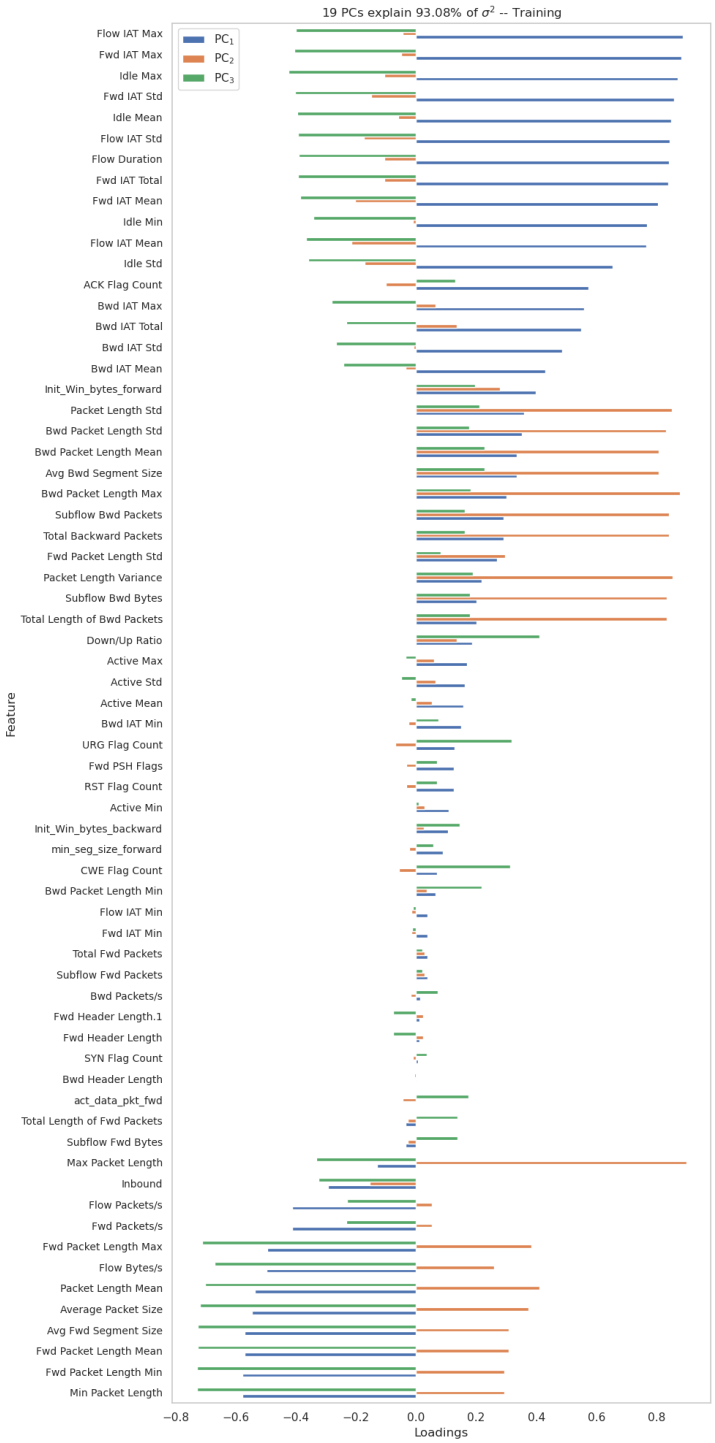


Figure 1.23: Loadings of Principal Components (PC1, PC2, PC3) for Each Feature

Section 2 Supervised Learning

Introduction The goal of this section is to build and evaluate machine learning models that can accurately identify different attacks based on the provided ground truth data which includes the labels of the attacks.

We start by splitting the dataset into training and test subsets in a stratified manner to ensure that the distribution of attack labels is consistent across both subsets. This approach maintains the representativeness of the data and ensures robust model evaluation.

Next, we select three different machine learning methods and train models using their default parameter configurations. We evaluate the performance of each model on both the training and test sets. For each model, we output the confusion matrix and a classification report which includes metrics such as precision, recall, and F1-score. These results help us identify any signs of overfitting or underfitting.

Following this, we tune the hyperparameters of the selected models using cross-validation techniques to improve model performance. This involves finding the optimal configuration of hyperparameters and comparing the performance of the models before and after tuning. The model with the best performance is identified based on the evaluation metrics.

Finally, we investigate the instances of false positives and false negatives produced by the models. This analysis focuses on understanding the characteristics of the misclassified samples and identifying patterns or features that contribute to these errors. We report our findings highlighting the most notable cases of misclassification and providing insights into potential areas for model improvement.

Approach to select the best hyperparameters Initially, validation curves are created by varying the hyperparameters over a wide range to observe the performance of the model. We then identify regions where performance is best, narrowing the intervals and improving computational efficiency and model understanding. Finally, we perform a grid search on the selected parameter intervals using cross-validation techniques to evaluate model performance on each parameter combination. This combined approach allows us to initially explore a wide range of values to identify promising regions and then perform a detailed search within those regions, optimizing both computational time and model accuracy.

What metrics do we use in the validation score? When creating a validation curve to evaluate the performance of a model, the choice of metric to use between accuracy and F1 score depends mainly on the distribution of classes in the dataset. Accuracy is a simple measure that represents the proportion of correct predictions out of the total number of predictions but it has limitations in the presence of unbalanced classes. The F1 score, on the other hand, is the harmonic mean of accuracy and recall and is useful when you have a dataset with unbalanced classes.

So in our specific case where one class is underrepresented with a size of 15% relative to the others, accuracy is not the best metric so using the F1 score is more appropriate.

2.1 Logistic Regression

Introduction Logistic regression is a technique used for binary classification problems that can be extended to multiclass classification problems. This statistical model is based on the concept of estimating the probability that a given observation belongs to one of the possible classes using a logistic function.

The advantages of logistic regression include the possibility of using regularization (such as L1 and L2) to prevent overfitting. In addition, logistic regression is relatively simple to implement and does not require significant computational resources, making it suitable even for large datasets.

However, logistic regression also has some disadvantages. A significant limitation is its assumption of linearity between the independent variables and the logarithm of probabilities which can reduce performance in the presence of nonlinear relationships.

Hyperparameters Below are the hyperparameters we evaluate for our model:

- **solver**: Algorithm to use in the optimization problem.
  - newton-cg
  - lbfgs
  - liblinear
  - sag
  - saga
- **multi_class**: Specifies how to handle multi-class classification.
  - ovr: One-vs-Rest.
  - multinomial: Multinomial loss fit across the entire probability distribution.
- **penalty**: Specifies the norm used in the penalization (regularization).
  - l1: Uses L1 regularization (Lasso) which can set some coefficients to zero effectively performing feature selection.
  - l2: Uses L2 regularization (Ridge) which tends to distribute the error among all the parameters reducing the impact of each one.
  - elasticnet: Combines L1 and L2 penalties allowing for a mix of feature selection and regularization.
  - none: No regularization applied.
- **C**: Inverse of regularization strength, controls the trade-off between achieving a low error on the training data and minimizing the norm of the coefficients. Smaller values specify stronger regularization.

Training with default parameters

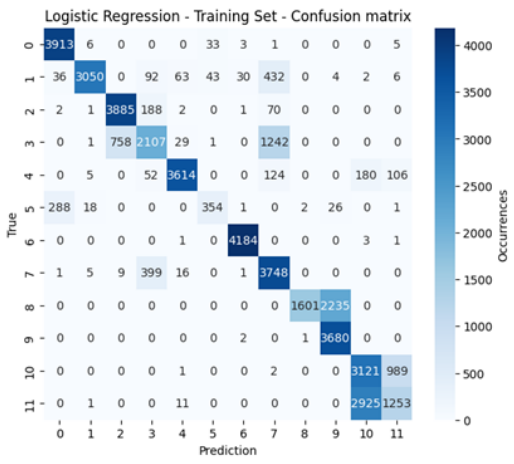


Figure 2.1: Logistic Regression – Training Set – Confusion Matrix

Table 2.1: Logistic Regression – Training Set – Classification report

	Precision	Recall	F1-Score	Support
0	0.92	0.99	0.95	3961
1	0.99	0.81	0.89	3758
2	0.84	0.94	0.88	4149
3	0.74	0.51	0.60	4138
4	0.97	0.89	0.92	4081
5	0.82	0.51	0.63	690
6	0.99	1.00	0.99	4189
7	0.67	0.90	0.77	4179
8	1.00	0.42	0.59	3836
9	0.62	1.00	0.76	3683
10	0.50	0.76	0.60	4113
11	0.53	0.30	0.38	4190
Accuracy		0.77 (44967)		
Macro avg	0.80	0.75	0.75	44967
Weighted avg	0.80	0.77	0.76	44967

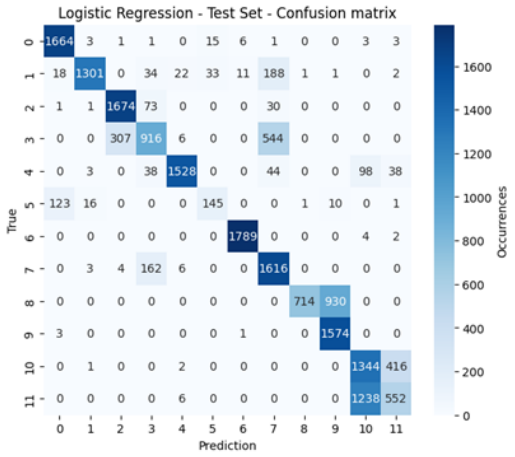


Figure 2.2: Logistic Regression – Test Set – Confusion Matrix

Table 2.2: Logistic Regression – Test Set – Classification report

Class	Precision	Recall	F1-Score	Support
0	0.92	0.98	0.95	1697
1	0.98	0.81	0.89	1611
2	0.84	0.94	0.89	1779
3	0.75	0.52	0.61	1773
4	0.97	0.87	0.92	1749
5	0.75	0.49	0.59	296
6	0.99	1.00	0.99	1795
7	0.67	0.90	0.77	1794
8	1.00	0.43	0.61	1644
9	0.63	1.00	0.77	1578
10	0.50	0.76	0.60	1763
11	0.54	0.31	0.39	1796
Accuracy		0.77 (19272)		
Macro avg	0.79	0.75	0.75	19272
Weighted avg	0.80	0.77	0.76	19272

The results obtained in the classification reports and in the figures 2.1 and 2.2 show parallel features in both the training set and the test set indicating that the model seems to generalize well. Using the macro_avg metric we aim to identify hyperparameters that can further improve the performance of the model.

Validation Curve Multiclass

Multiclass

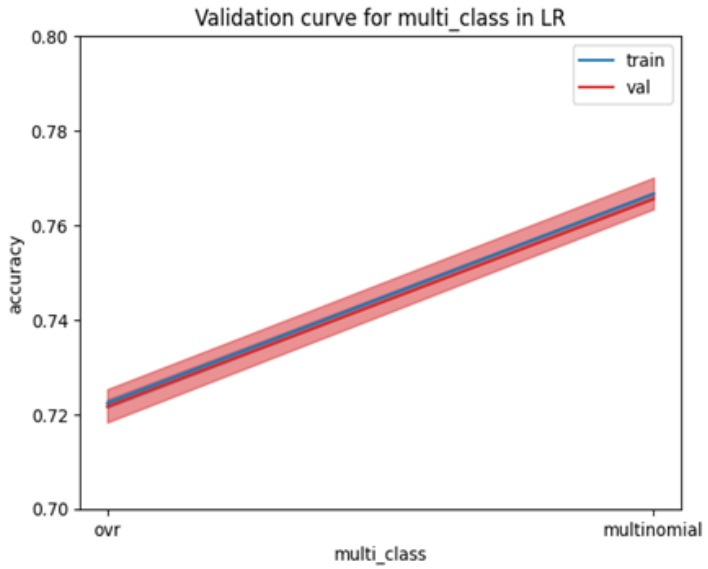


Figure 2.3: Validation curve for multi_class in LR

The figure 2.3 shows that performance as measured by the F1 score is better for the "multinomial" option than "ovr" for both training and validation datasets. It is also noted that the training and validation curves are parallel and very close indicating that the model generalizes well showing no signs of significant overfitting or underfitting. Finally, the confidence intervals are narrow indicating relatively low variability in performance between the different folds of the cross-validation.

Solver

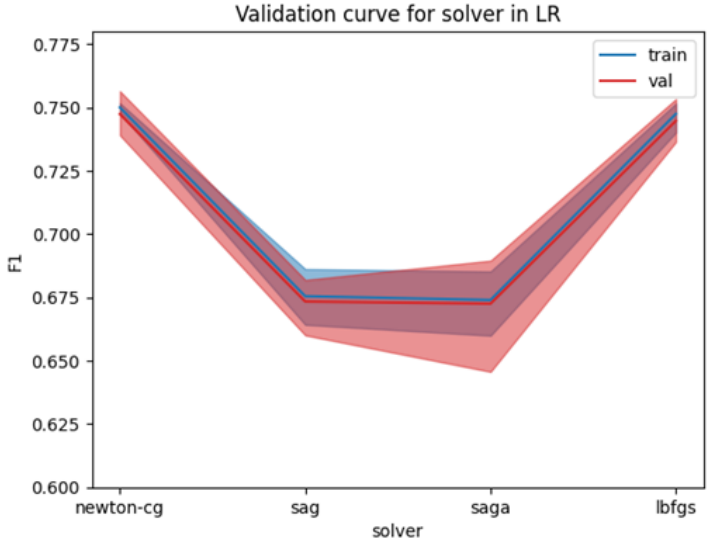


Figure 2.4: Validation curve for solver in LR

Analyzing the figure 2.4, we see that the "newton-cg" solver and the "lbfgs" solver have the highest F1 score values for both the training and validation sets. These two solvers maintain F1 scores close to 0.75 suggesting that they offer superior performance compared to the other solvers considered. The validation curves for "newton-cg" and "lbfgs" are parallel and close to the training curves indicating that the model generalizes well without showing obvious signs of overfitting or underfitting.

The "sag" and "saga" solvers, on the other hand, show lower performance with F1 scores dropping to about 0.65-0.68. This drop in performance is evident in both the training and validation data. In addition, the curves for "sag" and "saga" are lower and show greater variability as indicated by the wider confidence intervals. This suggests that these solvers may not be suitable for the dataset.

In conclusion, we choose "newton-cg" as the solver because it has a slightly higher F1 than "lbfgs"; later we will analyze both of them in the random search grid.

Penalty

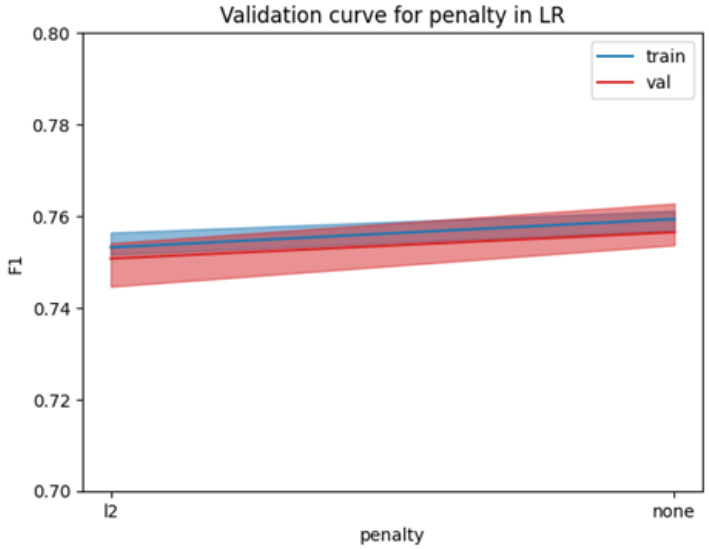


Figure 2.5: Validation curve for penalty in LR

The figure 2.5 shows that performance as measured by the F1 score is slightly better for the "none" option than for "l2" for both the training and validation datasets indicating that the absence of regularization may lead to better performance in the context of this specific dataset.

To confirm the results obtained, we will proceed by using a randomized search, a random grid search examining a range of values for the parameter C in comparison with the penalty.

Randomized search grid

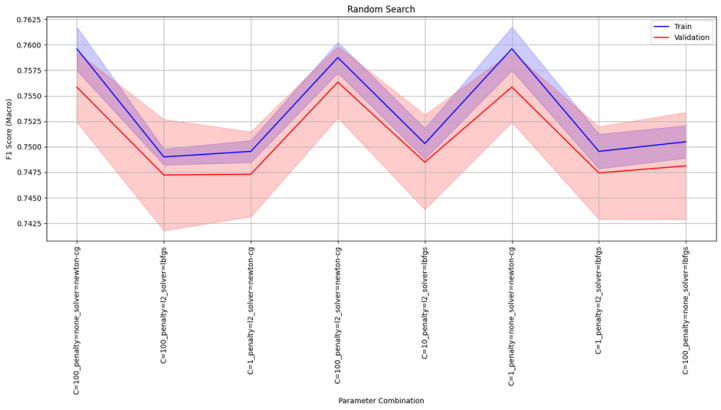


Figure 2.6: Random Search – Logistic Regression

```
Best F1 Score (Macro): 0.7563350567282467
Parameter Combination: C=100_penalty=l2_solver=newton-cg
```

These parameters were identified as the best in the context of our classification problem achieving an F1 macro score on the validation set of 76 percent. This represents an improvement over the initial score of 75% suggesting that the model has gained in accuracy and recall for the different classes.

The validation graph (figure 2.6) which shows the performance in terms of accuracy for each iteration of the search shows that the training and validation scores have remained stable indicating that the generalization of the model has not been compromised.

Analyzing Misclassifications: False Positives and Negatives in Feature Context

```
[[1662    6    0    1    0   17    6    1    0    1    3    0]
 [  13 1335    1   34   19   34   11  160    1    1    1    0]
 [   0    3 1668   72    0    0   36    0    0    0    0    0]
 [   0    1  307  908    6    0  551    0    0    0    0    0]
 [   0    3    0   38 1624    0   48    0    0    0   25   11]
 [ 116    5    0    0    0  163    0    0    1   10    0    1]
 [   0    0    0    0    0    0 1787    0    0    0    6    2]
 [   0    3    3  137    9    0 1638    1    0    0    0    0]
 [   0    0    0    0    0    0    0    0  716  927    0    0]
 [   0    0    0    0    0    0    0    0    2 1572    0    0]
 [   0    0    0    0    0    0    0    0    0    0 1118  643]
 [   0    0    0    6    0    0    0    0    0    0 1017  773]]
```

Listing 2.1: Confusion matrix

Classes 0, 2, 6, and 9 show a high number of true positives indicating that the model is particularly effective in classifying these categories. However, class 1 shows numerous false positives and false negatives with many instances misclassified as class 7. Class 3 shows significant confusion with class 7 with 551 instances misclassified. Class 10 has 643 false positives from class 11 and 1017 false negatives misclassified as class 11. In addition, class 7 shows confusion with several other classes particularly with classes 3 and 1. Classes 8 and 11 show significant confusion with each other with 927 instances of class 8 classified as class 9 and 643 instances of class 10 classified as class 11.

These observations suggest that the model is effective for some classes but has separability problems for others.

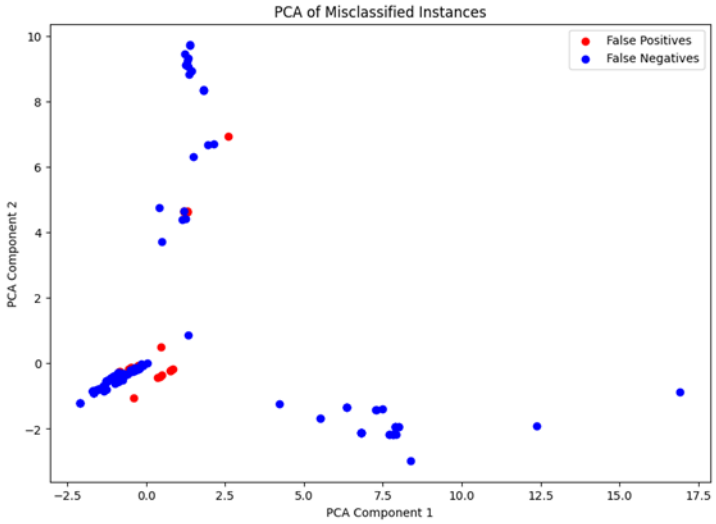


Figure 2.7: PCA of Misclassified Instances

As can be seen on figure 2.7, false negatives are distributed in different regions of the PCA space indicating that instances misclassified as negative are scattered among different principal components. False positives are fewer in number and show some concentration suggesting that there are specific regions of the feature space where the model tends to misclassify instances as positive. The presence of misclassifications in specific areas of the PCA space suggests that there are some combinations of features that the model fails to discriminate effectively.

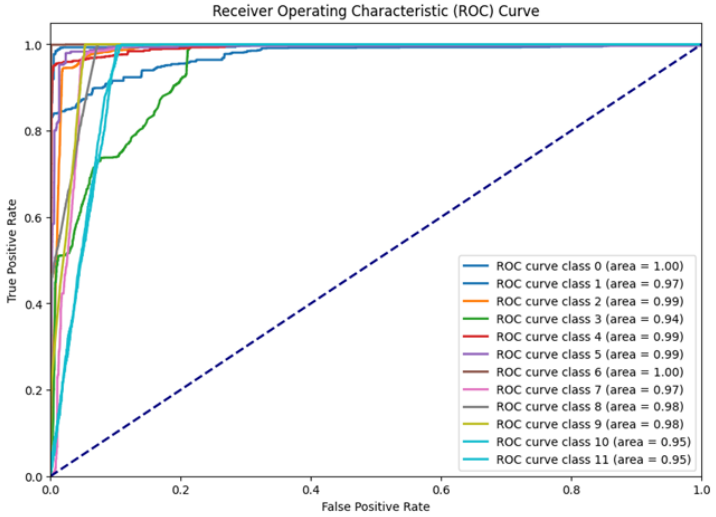


Figure 2.8: Receiver Operating Characteristic (ROC) Curve

The plot of ROC curves for each class (figure 2.8) shows that the logistic regression model has excellent discrimination abilities for most classes. Classes 0 and 6 show an area under the curve (AUC) of 1.00 indicating perfect performance and an ability of the model to distinguish these classes without error. Classes 2, 4, 5, 8, and 9 show AUCs between 0.98 and 0.99 suggesting that the model discriminates these classes very well with high rates of true positives and low rates of false positives. However, classes 1, 3, 7, 10, and 11 with AUCs between 0.94 and 0.97 show greater challenges indicating that the model has more difficulty distinguishing these classes than those with higher AUCs.

In conclusion, the ROC curves and associated AUCs indicate that the logistic regression model has excellent discrimination capabilities for most classes with some exceptions where performance is good but can be improved. Feature analysis and adoption of more advanced models are promising strategies to address the identified challenges.

2.2 Random Forest

Introduction The Random Forest is a powerful machine learning algorithm used for classification and regression. It combines multiple decision trees to enhance robustness and accuracy, adapting well to various types of data without requiring extensive tuning. Thanks to bagging and the random selection of feature subsets, it significantly reduces the risk of overfitting ensuring better generalization on new data. It can effectively handle missing data maintaining good performance. Additionally, it provides estimates of variable importance helping to identify the most influential features.

However, the Random Forest is less interpretable compared to simpler models like logistic regression, which can be an issue when model interpretability is crucial. Furthermore, it is computationally expensive and requires long training times especially with large datasets like ours. Despite these disadvantages, the Random Forest remains an excellent choice for those seeking a robust and accurate model capable of handling complex data and providing useful insights.

Hyperparameters Below are the hyperparameters we evaluate for our model:

- **n_estimators**: The number of trees in the forest.
  - Positive integer. A higher number of trees generally improves the model's performance (up to a certain point) because it combines more estimates but it also increases computation time and memory usage.
- **max_depth**: The maximum depth of each tree.
  - Positive integer. A higher value allows the trees to grow deeper capturing more details of the dataset but can lead to overfitting. A lower value can prevent overfitting but risks underfitting.
- **bootstrap**: Indicates whether to use bootstrap sampling when building trees.
  - True: Trees are built on bootstrapped samples of the dataset (with replacement).
  - False: Trees are built on samples without replacement.
- **max_features**: The maximum number of features to consider for the best split.
  - auto: Uses all features.
  - sqrt: Uses the square root of the total number of features.
- **min_samples_leaf**: The minimum number of samples required to be at a leaf node.
  - Positive integer. Specifying a minimum number of samples per leaf can reduce overfitting as nodes cannot be too specific to the training data.
- **min_samples_split**: The minimum number of samples required to split an internal node.
  - Positive integer. Setting a higher value for this parameter can prevent the data from fragmenting into too many small segments thereby reducing overfitting.

Training with default parameters Training the model without using any particular parameters we obtained the following performance.

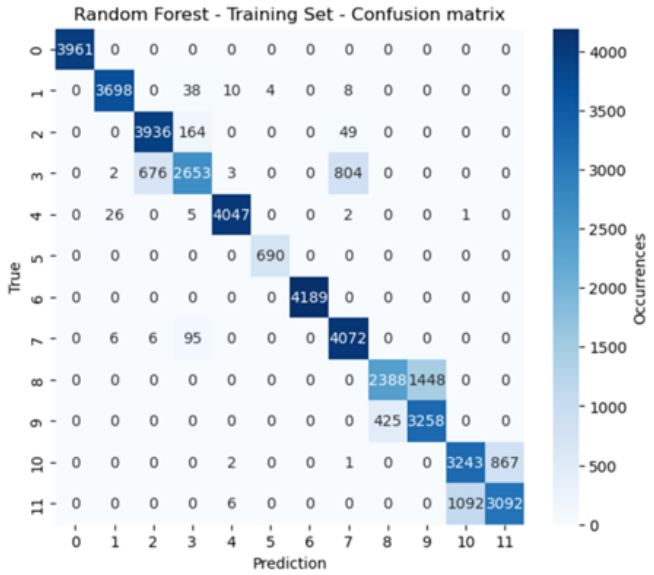


Figure 2.9: Random Forest – Training Set – Confusion matrix

Table 2.3: Random Forest – Training Set – Classification report

	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	3961
1	0.99	0.98	0.99	3758
2	0.85	0.95	0.90	4149
3	0.90	0.64	0.75	4138
4	0.99	0.99	0.99	4081
5	0.99	1.00	1.00	690
6	1.00	1.00	1.00	4189
7	0.82	0.97	0.89	4179
8	0.85	0.62	0.72	3836
9	0.69	0.88	0.78	3683
10	0.75	0.79	0.77	4113
11	0.78	0.74	0.76	4190
Accuracy		0.87 (44967)		
Macro avg	0.89	0.88	0.88	44967
Weighted avg	0.88	0.87	0.87	44967

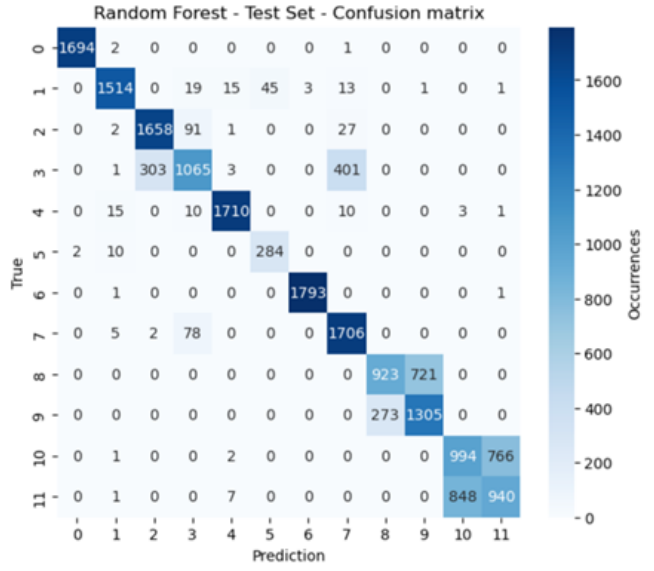


Figure 2.10: Random Forest – Test Set – Confusion matrix

Table 2.4: Random Forest – Test Set – Classification report

	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	1697
1	0.98	0.94	0.96	1611
2	0.84	0.93	0.89	1779
3	0.84	0.60	0.70	1773
4	0.98	0.98	0.98	1749
5	0.86	0.96	0.91	296
6	1.00	1.00	1.00	1795
7	0.79	0.95	0.86	1791
8	0.77	0.56	0.65	1644
9	0.64	0.83	0.72	1578
10	0.54	0.56	0.55	1763
11	0.55	0.52	0.54	1796
Accuracy		0.81 (19272)		
Macro avg	0.82	0.82	0.81	19272
Weighted avg	0.81	0.81	0.81	19272

The results of the Random Forest model show an accuracy of 87% on the training set and 81% on the test set suggesting a fair ability to generalize. Some classes, however, show a significant drop in performance in the test set compared to the training set highlighting possible overfitting problems. Classes 10 and 11 have particularly low F1-scores in the test set (0.55 and 0.54) indicating difficulties in their correct classification on new data. On figures 2.9 and 2.10 are shown the confusion matrix of the training set and test set.

Validation Curves

Max depth

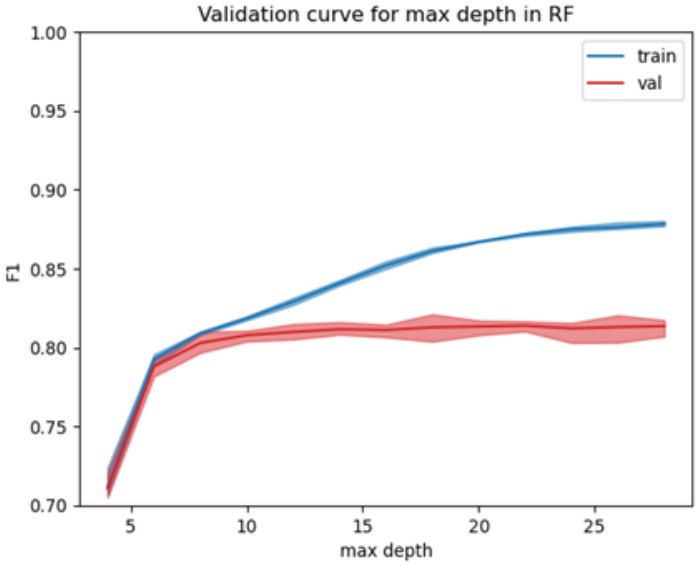


Figure 2.11: Validation Curve for max depth in RF

The validation curve for the maximum depth (see figure 2.11) in the Random Forest model shows that training accuracy (blue line) continuously increases as the maximum depth increases. However, validation accuracy (red line) initially increases and peaks around a maximum depth of 10 to 15. Beyond this point, the validation accuracy plateaus and even slightly decreases suggesting that the model begins to overfit the training data.

Using this analysis, we can set the max_depth to 10 for our subsequent experiments and proceed to explore other parameters.

Number of estimators

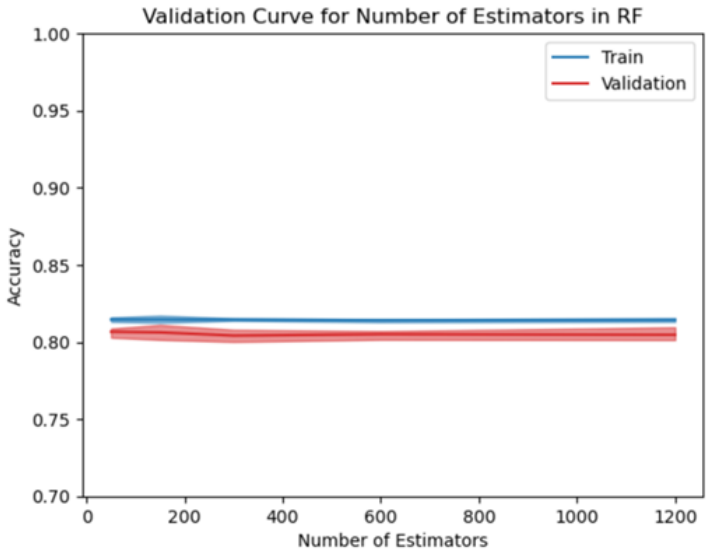


Figure 2.12: Validation Curve for Number of Estimator in RF

We observe from figure 2.12 that as the number of estimators increases, the training accuracy remains relatively stable around 0.82. This indicates that adding more trees beyond a certain point does not significantly enhance the model's ability to fit the training data.

The validation accuracy also remains relatively stable around 0.81 across the different values of n_estimators. This suggests that increasing the number of trees does not substantially impact the model's generalization performance.

Based on the validation curve, there is no significant change in performance with the increase in n_estimators. Therefore, a value in the range of 100-200 estimators is likely sufficient as adding more trees does not improve accuracy but increases computational cost.

Using this analysis, we can confidently set the n_estimators to 150 for our subsequent experiments balancing performance and computational efficiency.

Min Samples

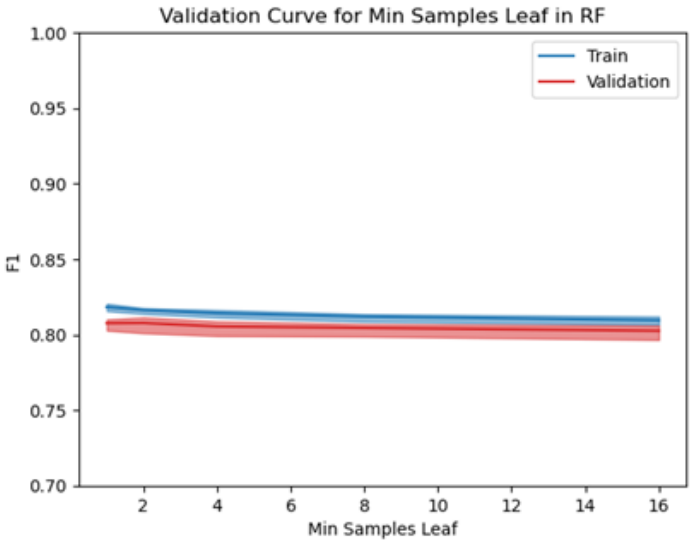


Figure 2.13: Validation Curve for Min Samples Leaf in RF

Looking at the figure 2.13, we see that both the training curve (blue) and the validation curve (red) tend to decrease slightly as the value of min_samples_leaf increases. Both curves stabilize around an F1 score of about 1-2.

The optimal value of min_samples_leaf to choose is 1 since F1 scores for training and validation are slightly better than higher values and the model maintains a good balance between complexity and generalization ability.

We will now proceed to explore other parameters using grid search to further fine-tune the model.

Grid Search

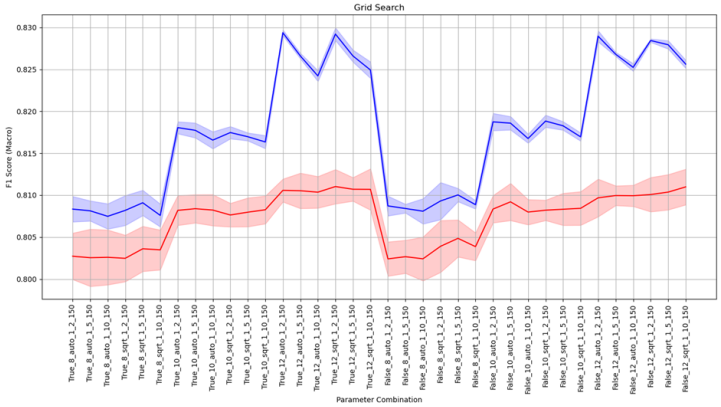


Figure 2.14: Grid Search

```
Best F1 Score (Macro): 0.8081864094324255
Parameter Combination: True_10_auto__1_2_150
```

There is no improvement in model performance before and after tuning as evidenced by the fact that the Macro Avg F1-Score remains essentially unchanged around 0.81. However, the level of overfitting has decreased as indicated by the reduction in training set accuracy from 87% to 82% signaling that the model is less overtrained on the training set. The Grid Search is shown on figure 2.14

Analyzing Misclassifications: False Positives and Negatives in Feature Context

```
Confusion Matrix:
[[1694    2    0    0    0    0    0    1    0    0    0    0]
 [   0 1442    0   24   20   47    3   73    0    1    0    1]
 [   0    2 1672   76    1    0    0   28    0    0    0    0]
 [   0    1  306 1011    7    0  448    0    0    0    0    0]
 [   0   13    0    8 1704    0   10    0    0   11    3    0]
 [   3    1    0    0    0    1  291    0    0    0    0    3]
 [   0    0    0    0    0    0    0 1794    0    0    0    0]
 [   0    2    2   75    3    0 1709    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0  721  923    0    0]
 [   0    0    0    0    0    0    0    0    7 1570    0    0]
 [   0    1    0    0    0    0    0    0    0    0  950  808]
 [   0    0    0    0    0    0    0    0    7    0  746 1043]]
```

The confusion matrix shows that the model has good overall accuracy with most instances classified correctly. However, there are significant misclassifications between some pairs of classes such as between classes 3 and 8 and between classes 6 and 8.

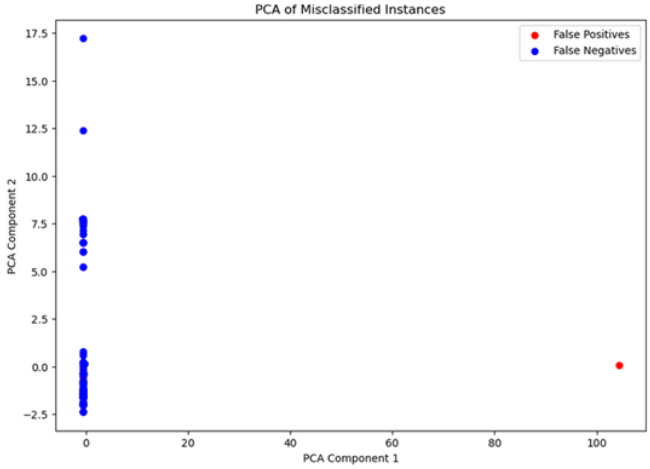


Figure 2.15: PCA of Misclassified instances

What the matrix illustrates is further evidenced by PCA where false positives and false negatives are distinguished indicating potential areas of improvement for the model (see figure 2.15). The reduction in overfitting observed in the decrease in training set accuracy suggests that the model is now less prone to learning noise specific to the training dataset.

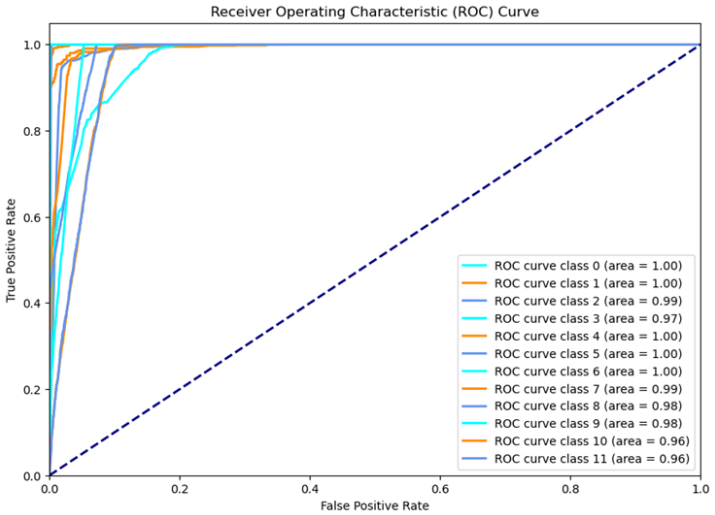


Figure 2.16: Receiver Operating Characteristic (ROC) Curve

The ROC curves (see figure 2.16) show that the model has excellent ability to discriminate between classes with most classes achieving an area under the curve (AUC) very close to 1.0. Classes 0, 1, 4, 5, and 6 have a perfect AUC of 1.0

indicating error-free classification. Classes 2, 7, 8, and 9 have an AUC between 0.98 and 0.99 demonstrating very high but not perfect discrimination ability. Classes 3, 10, and 11 have AUCs of 0.97 and 0.96 suggesting a slight imperfection in discrimination although they still remain high performers. These results indicate that while the model is generally very effective there is room for improvement for classes with slightly lower AUCs especially to ensure perfect separability in all cases.

2.3 Support Vector Machines

Introduction The Support Vector Machine (SVM) is a machine learning model capable of handling high-dimensional data and demonstrates robustness in preventing overfitting. SVMs work by finding the hyperplane that maximizes the margin between different classes contributing to improved model generalization. Additionally, their ability to find an optimal hyperplane in high-dimensional spaces makes them ideal for complex problems.

Thanks to kernels, SVMs can handle non-linear relationships in the data by transforming them into a high-dimensional space where a linear hyperplane can be used to separate the classes. The goal is to find the hyperplane that separates the classes with the widest possible margin thereby reducing the risk of misclassification.

Despite their versatility, effectiveness in high-dimensional spaces, and robustness, training an SVM is computationally expensive and slow. Furthermore, selecting the hyperparameters is time-consuming. Additionally, SVMs can sometimes be sensitive to noisy data.

Hyperparameters

- **Kernel**: Determines the kernel function used to transform the data. The explored options include:
  - Linear
  - Polynomial
  - Radial Basis Function
  - Sigmoid
- **C (Regularization Parameter)**: Controls the trade-off between maximizing the margin of the hyperplane and minimizing the classification error on the training data. A high value of C aims to classify all training points correctly while a low value allows for a wider margin.
- **Gamma (γ)**: Specific to RBF, polynomial, and sigmoid kernels. It influences the shape of the decision boundary.

Training with default parameters

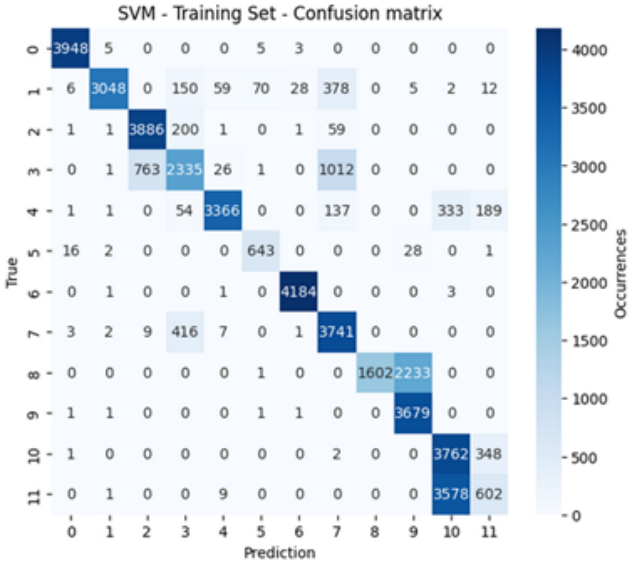


Figure 2.17: SVM – Training Set – Confusion matrix

Table 2.5: SVM – Training Set – Classification report

	Precision	Recall	F1-Score	Support
0	0.99	1.00	0.99	3961
1	1.00	0.81	0.89	3758
2	0.83	0.94	0.88	4149
3	0.74	0.56	0.64	4138
4	0.97	0.82	0.89	4081
5	0.89	0.93	0.91	690
6	0.99	1.00	1.00	4189
7	0.70	0.90	0.79	4179
8	1.00	0.42	0.59	3836
9	0.62	1.00	0.76	3683
10	0.49	0.91	0.64	4113
11	0.52	0.14	0.23	4190
Accuracy		0.77 (44967)		
Macro avg	0.81	0.79	0.77	44967
Weighted avg	0.80	0.77	0.76	44967

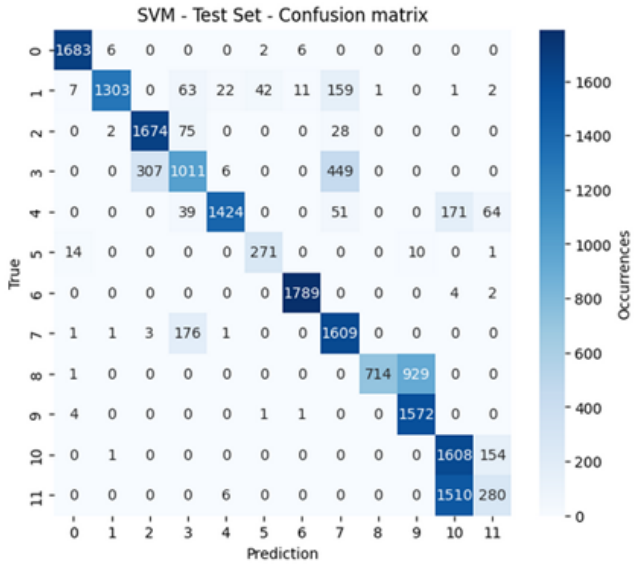


Figure 2.18: SVM – Test Set – Confusion matrix

Table 2.6: SVM – Test Set – Classification report

Class	Precision	Recall	F1-Score	Support
0	0.98	0.99	0.99	1697
1	0.99	0.81	0.89	1611
2	0.84	0.94	0.89	1779
3	0.74	0.57	0.64	1773
4	0.98	0.81	0.89	1749
5	0.86	0.92	0.89	296
6	0.99	1.00	0.99	1795
7	0.70	0.90	0.79	1791
8	1.00	0.43	0.61	1644
9	0.63	1.00	0.77	1578
10	0.49	0.91	0.64	1763
11	0.56	0.16	0.24	1796
Accuracy		0.78 (19272)		
Macro avg	0.81	0.79	0.77	19272
Weighted avg	0.81	0.78	0.76	19272

Analysis of the SVM model on the test set shows significant misclassifications (See figures 2.17 and 2.18 for the correlation matrixes). Class 3 is often confused with classes 2 and 7, class 7 with class 3, and class 8 with class 9. Class 11 presents the greatest difficulties with many examples misclassified as class 10. Accuracy ranges from 0.49 (class 10) to 1.00 (class 8), recall from 0.16 (class 11) to 1.00 (class 6), and the F1-score is particularly low for class 11 (0.24).

In the training set, the misclassification patterns are similar confirming the difficulties already observed in the test set. Suboptimal performance is mainly due to dataset imbalance and feature overlap between some classes. Optimizing hyperparameters and improving feature quality could help in reducing these misclassifications.

Validation Curves

Kernel

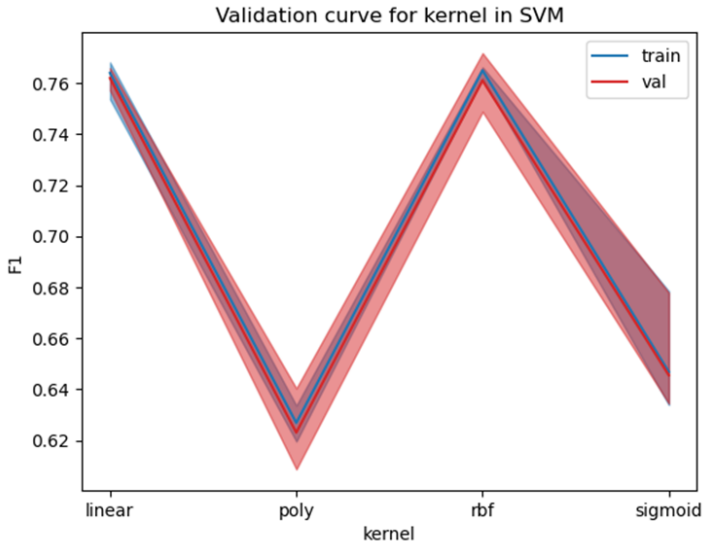


Figure 2.19: Validation curve for kernel in SVM

The linear kernel (see figure 2.19) shows similar accuracy as the RBF kernel. By performing various experiments, we found that the linear kernel requires significantly more training time making it difficult to tune. Therefore, the choice of the RBF kernel is motivated by practical considerations related to computational efficiency and model performance offering greater flexibility in modeling complex relationships in the data and better ability to generalize to unseen data.

C

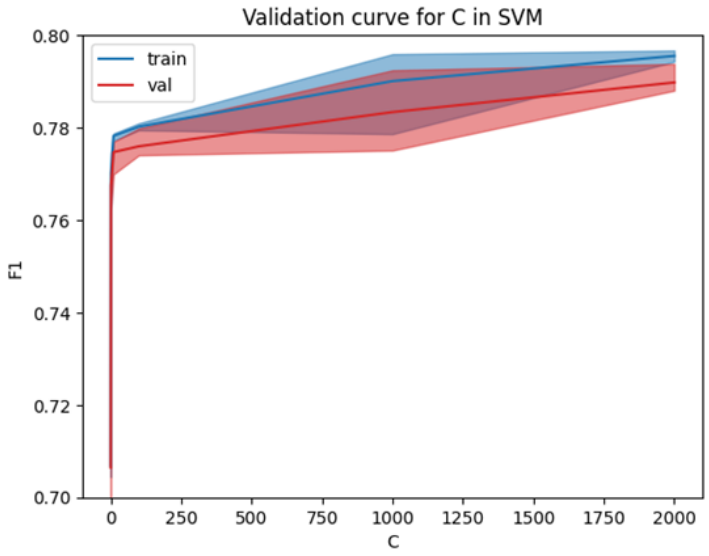


Figure 2.20: Validation curve for C in SVM

On figure 2.20, both curves show a significant initial improvement in F1 score as C increases which then tends to decrease the slope of this increase for large C. With very low values of C, performance on the validation set is poor indicating that the model is not sufficiently complex to capture the features of the data. As C increases, the model becomes more complex improving performance on both sets. However, at very high values of C, there is a slight tendency toward overfitting where the score on the training set is slightly higher than that on the validation set.

Nevertheless, at C=1000 the model seems to balance the trade-off between bias and variance well with an F1 score close to the maximum for both the training and validation sets indicating good generalization. This value of C avoids both underfitting associated with very low C values and the risk of overfitting associated with very high C values not printed here because they require large computational resources.

Gamma

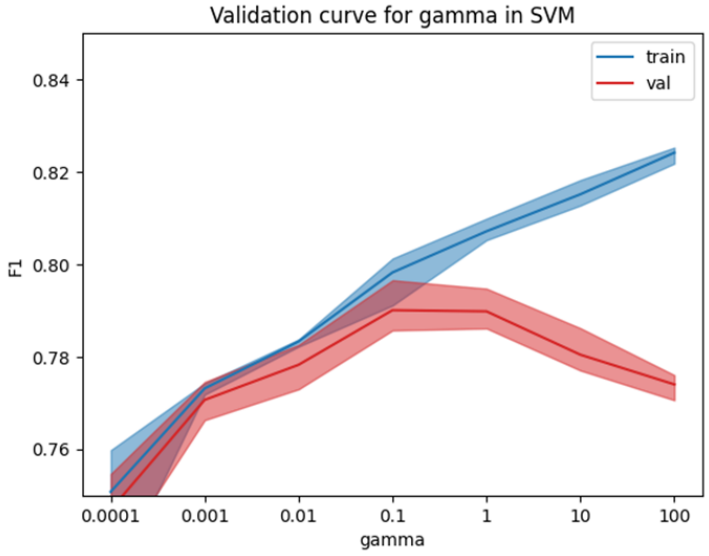


Figure 2.21: Validation curve for gamma in SVM

On figure 2.21 is represented the Validation curve for gamma in SVM. Gamma is a crucial parameter because it determines the influence of each training point. A small value of gamma implies that the influence of a single point extends far, while a large value means that the influence is limited to a small area. Looking at the graph, we notice that with very low values of gamma, both F1 scores for training and validation are low. This suggests that the model is too simple to capture the complexities of the data (underfitting). As the range increases, the F1 score on the training set continues to increase,

indicating that the model is becoming more complex and fitting the training data better. However, the behavior of the F1 score on the validation set is different. Initially, it increases with increasing range, reaching a peak around gamma = 0.1. This is the point at which the model manages to generalize best on the validation data, striking a good balance between model complexity and the ability to fit the training data. After this point, the F1 score on the validation set begins to decrease, while that on the training set continues to rise. This indicates that the model is beginning to overfit to the training data, losing the ability to generalize to the unseen data (overfitting).

Performance evaluation of tuning

```
Average F1 Score (Training): 0.7994878524579546
Average F1 Score (Test): 0.7958686795979264
Parameter Combination: kernel=rbf, C=1000, gamma=0.1 (Used in this run)
```

Although grid search or random search techniques are valid for hyperparameter optimization, their computational requirements were found to be excessive for this specific model. We have already achieved a significant increase in performance with F1 scores going from 77% to 80% on the training set and from 77% to 79.6% on the test set.

To further optimize the model without incurring significant computational overhead, we could pursue a more focused approach exploring a narrow range of gamma parameters between 0.1 and 1.0 where a maximum might reside evaluating them over a range of C values.

Analyzing Misclassifications: False Positives and Negatives in Feature Context

```
Confusion Matrix:
[[1693    1    0    0    0    1    2    0    0    0    0]
 [   6 1418    0   19   16   41    7  103    0    0    1]
 [   2    1 1673   75    0    0    0   28    0    0    0]
 [   0    1  307  992    6    0    0  467    0    0    0]
 [   0    2    0   30 1624    0    0   58    0   31    4]
 [   5    9    0    0    1  280    0    0    1    0    0]
 [   0    0    0    0    0    0 1789    0    0    4    2]
 [   1    5    2   62    6    0    0 1715    0    0    0]
 [   5    0    0    0    0    0    0    0  711  928    0]
 [   3    1    0    0    0    0    0    0    0 1574    0]
 [   0    1    0    0    3    0    0    0    0    0 1336  423]
 [   0    0    0    0    7    0    0    0    0    0 1231  558]]
```

The confusion matrix shows that most classes are well classified, with high values on the main diagonal, but there are some obvious misclassifications. Class 1 is frequently confused with class 7, indicating 103 false positives for class 1 classified as class 7. Class 3 has 307 false negatives, mainly confused with class 2, while class 7 shows problems with 62 false negatives confused with class 3. Class 8 is strongly confused with class 9, with 928 false positives. Class 11 shows 558 false negatives mainly confused with class 12. These misclassifications indicate that there are specific pairs of classes that the model cannot distinguish correctly.

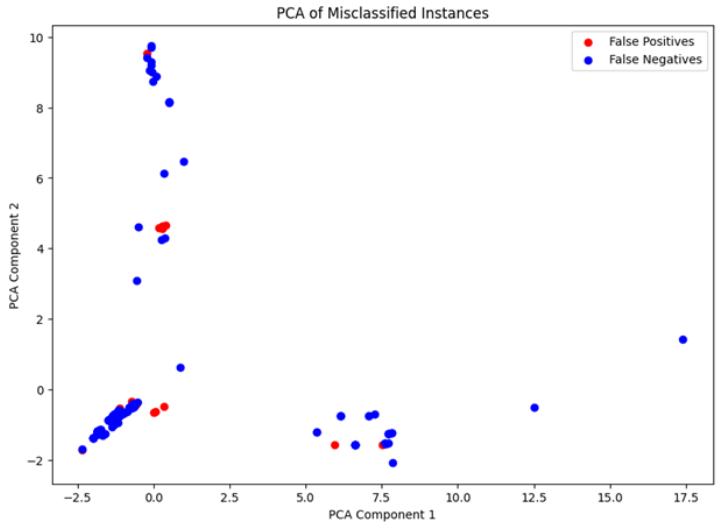


Figure 2.22: PCA of Misclassified Instances

Looking at the graph on figure 2.22, we notice that there are several areas with a concentration of false negatives (blue point) and some false positives (red point). This indicates that there are groups of data that the model frequently confounds. In particular, we can see:

- A significant concentration of false negatives around PCA Component 1 = [-2.5,0] and PCA Component 2 = [2,0]. This suggests that in this region of the feature space, the model has difficulty correctly classifying instances.
- Some false positives are present in the same areas, indicating that these areas may have similar features across classes, causing confusion in the model.
- There are a few scattered areas where false negatives are present in isolation, indicating that some instances are particularly difficult to classify correctly.

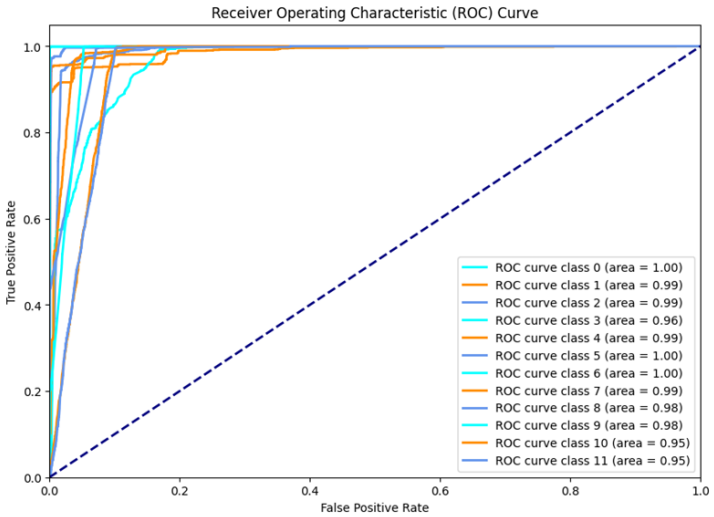


Figure 2.23: Receiver Operating Characteristic (ROC) Curve

Looking at the graph on figure 2.23, we see that classes 0, 5 and 6 have an area under the curve of 1.00, indicating perfect separation between classes. This result is consistent with the confusion matrix, which shows very few false negatives and false positives for these classes. For example, class 0 has only 1 false positive in class 1 and 1 in class 6, confirming the excellent performance of the model for this class.

Classes 1, 2, 4, 7 and 8 have an AUC of 0.99, indicating excellent performance. However, the confusion matrix reveals some misclassifications, especially for class 1, which has 103 false positives confused with class 7. Although these errors are relatively few compared to the total, they indicate areas for improvement.

Class 3 has an AUC of 0.96, slightly lower than the others. This is consistent with the confounding matrix, where class 3 shows 307 false negatives, mainly confounded with class 2. This suggests that the model has some difficulty in correctly distinguishing these two classes.

Classes 9 and 10 have AUCs of 0.98 and 0.95, respectively. The confounding matrix shows difficulty for class 9, with 928 false positives in class 8, and for class 10, with 423 false negatives confounded with class 11. These results indicate that although the performance is good, there are specific areas where the model could be improved.

Class 11 has an AUC of 0.95, which is the lowest among the classes.

This is in line with the confounding matrix, where class 11 shows 558 false negatives, mainly confounded with class 12. This suggests that the model has more difficulty in correctly classifying these instances than other classes.

Section 3 Unsupervised learning – Clustering

Introduction In this section, the goal is to cluster streams that produce similar, related, or coordinated patterns using unsupervised clustering techniques regardless of labels. To achieve this goal, we chose to use three clustering algorithms: K-Means, Gaussian Mixture Models (GMM), and DBSCAN.

The choice of these models is motivated by their different nature and their abilities to detect structures in the data. K-Means is known for its simplicity and efficiency in dividing data into spherical clusters based on Euclidean distance. This method is useful when the clusters are well separated and regular in shape. However, it may not perform well in the presence of clusters of irregular shape or variable density.

To address the limitations of K-Means, we have included the Gaussian Mixture Model, which uses a probabilistic approach to identify elliptical clusters based on Gaussian distributions. GMM is flexible in modeling clusters with different shapes and can provide a membership probability for each point, making it useful in complex scenarios.

Finally, we chose DBSCAN because of its ability to identify clusters of arbitrary shape and to handle noise. DBSCAN does not require specifying the number of clusters a priori but identifies dense areas of points and separates them from less dense regions, which it considers as noise. This approach is particularly advantageous when the data contain noise or have clusters of varying density.

For each algorithm, determining the optimal number of clusters is critical and can be done using methods such as the elbow method or silhouette analysis. These techniques help us understand the internal structure of the data and choose the number of clusters that best represents the data. Next, it is essential to find the best hyperparameters for each model to optimize their performance.

3.1 K-Means

Introduction The K-means method is a particularly effective clustering algorithm for clustering a dataset. The main objective of K-means is to minimize the sum of the distances between data points and the centroid of their respective cluster. The centroid is the midpoint of all points that belong to that cluster.

One of the critical aspects of using K-means is choosing the correct number of clusters. This number not only directly affects the quality of clustering but can also reveal significant details about the data itself. Inappropriate choice of the number of clusters can lead to misleading results, such as over- or under-segmentation of the data. Techniques such as the elbow method or silhouette index can help determine an appropriate number of clusters, balancing between optimizing the internal variance of clusters and minimizing model complexity.

Determine the Parameters of K-Means To select the optimal number of clusters for K-means clustering, it is useful to consider a combination of different evaluation metrics as each provides a different perspective on clustering quality.

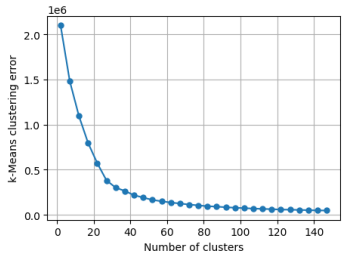


Figure 3.1: Silhouette

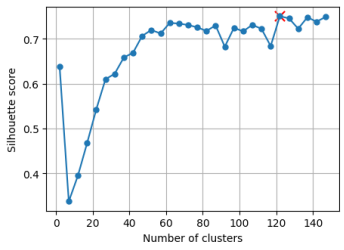


Figure 3.2: K-Means clustering error

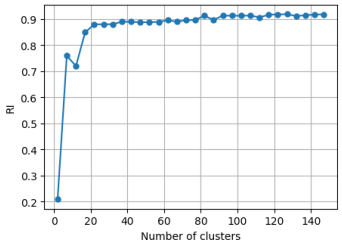


Figure 3.3: RI

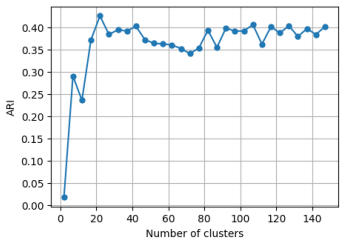


Figure 3.4: ARI

The elbow method looks for the point where the decrease in clustering error slows down (see figure 3.2), indicating that adding more clusters does not significantly improve the model. From the clustering error graph, it is observed that the elbow is around 20-40 clusters, where the rate of decrease in error slows down.

The silhouette score (see figure 3.1) measures how similar an object is to its own cluster relative to other clusters. A higher score indicates better-defined clusters. Looking at the silhouette score graph, it can be seen that it peaks around 120 clusters. However, choosing the maximum number of clusters based solely on this criterion could lead to selecting too many clusters, which may not be meaningful from a practical point of view.

The Rand Index (RI) and Adjusted Rand Index (ARI), which are shown on figures 3.3 and 3.4, both evaluate clustering similarity, but the ARI is normalized to account for chance, making it more reliable, especially in noisy datasets. Unlike RI, which ranges from 0 to 1, ARI ranges from -1 to 1, where negative values indicate agreement less than chance, enhancing its sensitivity to noise and outliers.

Combining these observations, a balance can be found between silhouette score and a practical number of clusters. Both ARI and RI suggest that around 20-40 clusters is optimal, with ARI peaking around 35 and RI stabilizing around 20. The elbow method supports these results, indicating a range of 20-40 clusters. This range offers a good compromise between clustering quality and practical significance. In light of the observed peak and stabilization points across various metrics, we have decided to utilize 35 clusters for our analysis.

Applying K-Means

```
k-Means with 35 clusters
Size of each cluster:  [  8216  14869   1244    239     38      2      5   8441   2125
                           53   6155      1   1978    250    196      2    163    283
                          109      3   7357     27   3396   2016    538    472      1
                          512   4366    158     30    129    823      5]
k_means clustering error: 282283.04
Silhouette: 0.65
Calinski-Harabasz: 13581.98
Davies-Bouldin: 0.73
RI: 0.89
ARI: 0.42
```

As can be seen, we have chosen to use two additional indices to aid in our evaluation:

- **The Calinski-Harabasz index**, also known as the between-cluster variance ratio, is defined as the ratio of the sum of between-cluster variance to the sum of within-cluster variance. A higher value indicates better cluster separation. An elevated value of 13581.98 suggests that the clusters are well-separated and compact, meaning that the points within

each cluster are close to each other and distant from points in other clusters. This index indicates that the clustering model has successfully created distinct clusters, which is a positive sign of the quality of the clustering.

- **The Davies-Bouldin index** measures the average similarity between each cluster and its most similar cluster. A lower value indicates better separation of the clusters. A value of 0.73 is considered good, suggesting that the clusters are well-separated from each other. A value close to zero would be ideal, indicating that each cluster is distinct with no significant overlaps.

ECDF of Number of Clusters

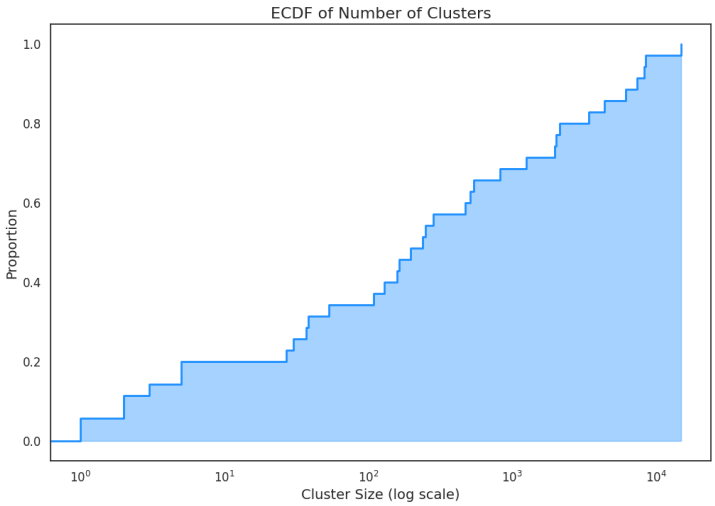


Figure 3.5: ECDF of Number of Clusters

On figure 3.5, the ECDF of Number of Clusters is shown. The curve starts to rise from relatively small cluster sizes (near 10^0 or 1) and continues to rise gradually. This indicates that there are many small cluster sizes that can identify noise.

The curve becomes steeper between 10^1 (10) and 10^3 (1000), suggesting that a large proportion of the clusters fall within this size range. This could indicate that most of the data are clustered in clusters of moderate size.

Toward the right edge of the graph, the curve flattens out near level 1, indicating that almost all of the data were considered in the ECDF calculation. This suggests that there are few very large clusters (near 10^4), while between 10^3 and 10^4 about 35 percent fall.

Thus, the ECDF shows that much of the data is concentrated in a moderate number of medium-sized clusters, with fewer very large or very small clusters. This may indicate that the K-means method has identified some dominant patterns in the data but also that there are many small clusters or outliers.

Cluster Visualization

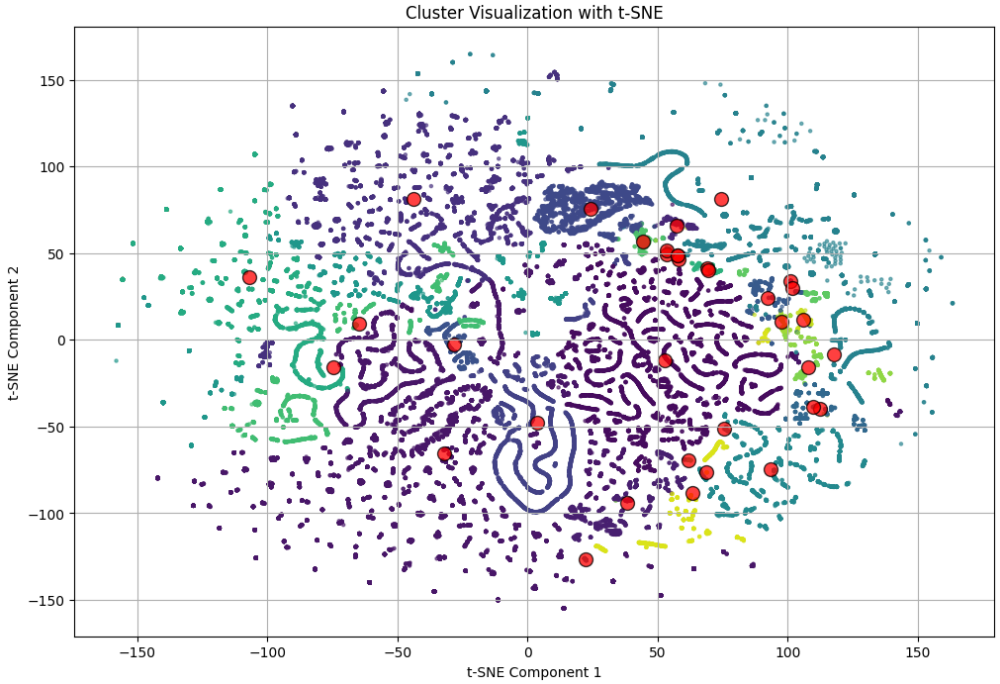


Figure 3.6: Cluster Visualization with t-SNE

Several significant aspects emerge from the image analysis of the t-SNE visualization (see figure 3.6). Some clusters appear well separated and compact, indicating effective classification of K-means for certain portions of the data. Centroids, represented by red dots, are accurately placed in the center of their respective clusters, suggesting correct identification of centers of mass for these groups. In addition, some dots are noted to appear isolated or located between multiple clusters, suggesting the presence of outliers or data that do not clearly adhere to a single cluster.

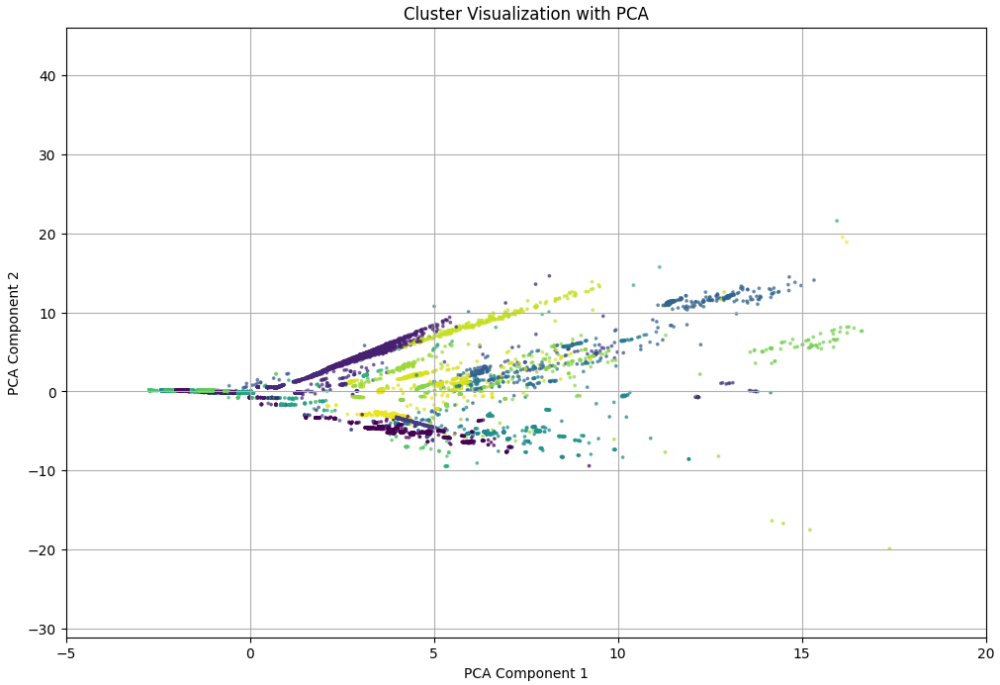


Figure 3.7: Cluster Visualization with PCA

Analysis by PCA of the clustered data using the K-means algorithm (see figure 3.7) reveals how the clusters are predominantly distributed along the first principal component, which is confirmed as the direction of greatest variance and informational relevance. This oriented distribution suggests a clear delineation of the most influential features of the data.

The different shapes and directions observed in the clusters indicate significant structural variations. However, the proximity and partial overlap of some clusters near the origin of the components may signal that not all data sets are distinct.

3.2 Gaussian Mixture Model

Introduction The Gaussian Mixture Model (GMM) is a probabilistic clustering model that assumes the generation of data by combining several Gaussian distributions, each of which corresponds to a cluster in the data set. Unlike the simpler K-means algorithm, which rigidly assigns each point to a single cluster, GMM evaluates the probability that each point belongs to several clusters. This approach, known as soft clustering, provides a richer and more detailed picture of the structure of the data, particularly when clusters intersect or overlap.

Unlike K-means, which presumes that all clusters are spherical in shape and of similar size, GMM is capable of fitting clusters of different elliptical shapes and sizes, thus allowing more complex relationships among the data to be captured.

The GMM, being a parametric model, fits well in scenarios where explicit statistical modeling of the data is essential, and it offers more flexibility in dealing with clusters of different shapes and sizes. On the other hand, its effectiveness may be limited by the choice of initial parameters and sensitivity to outliers in the data, aspects in which DBSCAN shows greater robustness due to its ability to effectively handle outliers and adapt to density variations in the data.

Determine the Parameters of GMM In the previous discussion, we examined the utility of the Silhouette Score, Rand Index (RI), and Adjusted Rand Index (ARI) for selecting the number of clusters. To these metrics, we now add log-likelihood, a measure assessing the probability that a Gaussian Mixture Model (GMM) generates the observed data.

It is important to note that, unlike the Elbow method commonly used in K-means analysis, this approach is not suitable for GMMs due to their probabilistic nature and intrinsic complexity. GMMs do not rely merely on distance measures like K-means but model the probabilities and covariances of the data. This shifts the focus from simply reducing the variance within clusters to a more critical sensitivity to initial parameters and the type of covariance used. Thus, applying a criterion based solely on variance change, as in the Elbow method, could yield less stable and reliable results.

In the context of Gaussian Mixture Models (GMM), log-likelihood serves as a fundamental measure to evaluate how accurately a model represents the observed data, indicating the probability that the model has generated the observed data. A higher value suggests that the model better fits its Gaussian components to the clusters in the data, providing a direct indication of model fit.

Moreover, log-likelihood also serves as the basis for information criteria such as the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC), which integrate log-likelihood with penalties accounting for the number of parameters in the model. These criteria are designed to help balance the complexity of the model against its ability to fit the data well without overfitting, which is why it is deemed appropriate to calculate them.

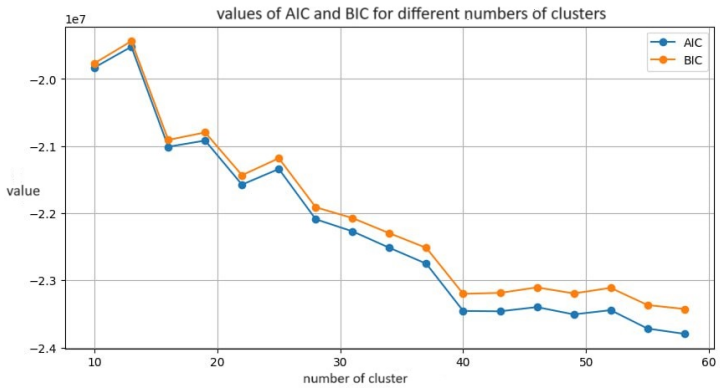


Figure 3.8: Values of AIC and BIC for different numbers of clusters

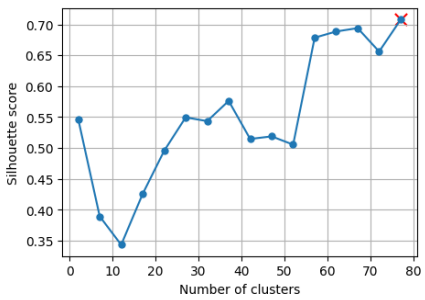


Figure 3.9: Silhouette score

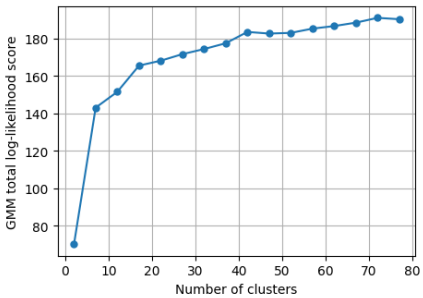


Figure 3.10: GMM total log-likehood score

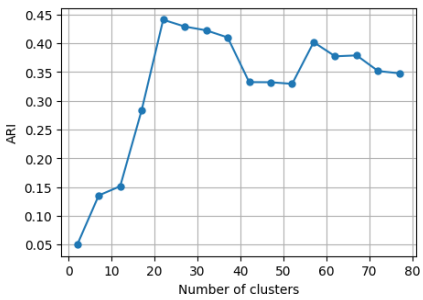


Figure 3.11: ARI

We examine the information from the upper images with the goal of identifying the optimal number of clusters based on measures such as Adjusted Rand Index (ARI), log-likelihood, Silhouette score, and AIC and BIC criteria (see figures 3.8, 3.9, 3.10 and 3.11).

The Silhouette score, which assesses how accurately the data were assigned to clusters, was initially low and showed a noticeable increase to 35 clusters. Continuing beyond this number, the value continued to increase, peaking around 80 clusters and then stabilizing. As mentioned above for K-Means, it is normal that the more clusters you have, the more the Silhouette goes up, but we need to find the right trade-off.

As for log-likelihood, a rapid increase up to 20 clusters was noted, after which its increase becomes marginal. This suggests that adding additional clusters beyond 20 does not make substantial improvements to the verisimilitude of the model, indicating that a number of clusters between 20 and 30 could adequately represent the data.

The ARI value showed rapid growth until it reached about 20 clusters, then fluctuated between 0.3 and 0.4 without showing any further significant trends. This index, being high when the clustering closely matches the actual clustering of the data, suggests that a high value is desirable for valid analysis, so a value between 20 and 30 seems good.

Finally, the AIC and BIC criteria, both indices that penalize increasing model parameters, showed a marked decrease up to 30-40 clusters, a point beyond which their reduction becomes less pronounced. This phenomenon suggests a good balance between model complexity and data fit.

This cross-sectional analysis suggests that a range of 20-40 clusters may represent the optimal clustering compromise. This range is supported by stabilizing log-likelihood, reducing AIC and BIC penalties, and improving Silhouette score.

Using a number of clusters of 30 thus seems to be the most balanced choice, maximizing the goodness of fit of the data without making the model overly complex, thereby also facilitating the interpretation of the results.

Applying GMM

```
Number of clusters:  30
Size of each cluster:  [  3425   237   17118   6196   489   2121   1121   6113
                            27     246    631   8313     38      1     686    2043
                            53       1     183   11358    240     21     333       2
                          1831     268       5      37     1100]
Silhouette: 0.55
RI: 0.88
ARI: 0.43
```

ECDF of Number of Clusters

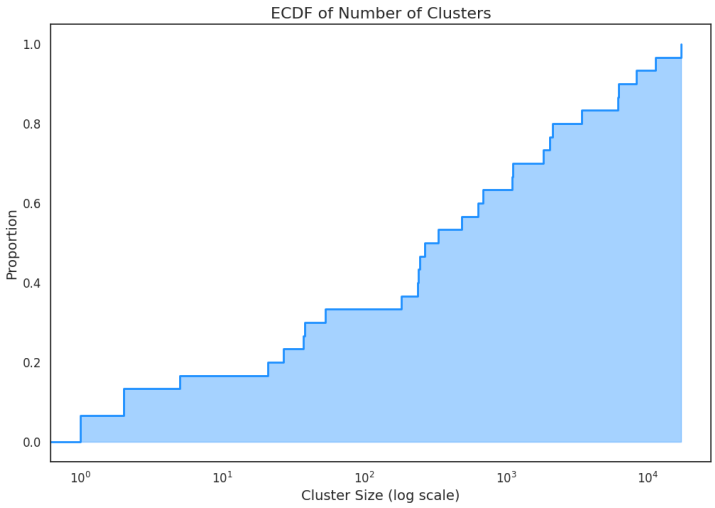


Figure 3.12: ECDF of Number of Clusters

The figure 3.12 reveals a significant variation in cluster size. A large number of smaller clusters suggests that the GMM identified several subgroups in the data, while the presence of a few larger clusters indicates that the model detected dominant features that attracted more points. This distribution highlights both the diversity of the data and the existence of predominant trends.

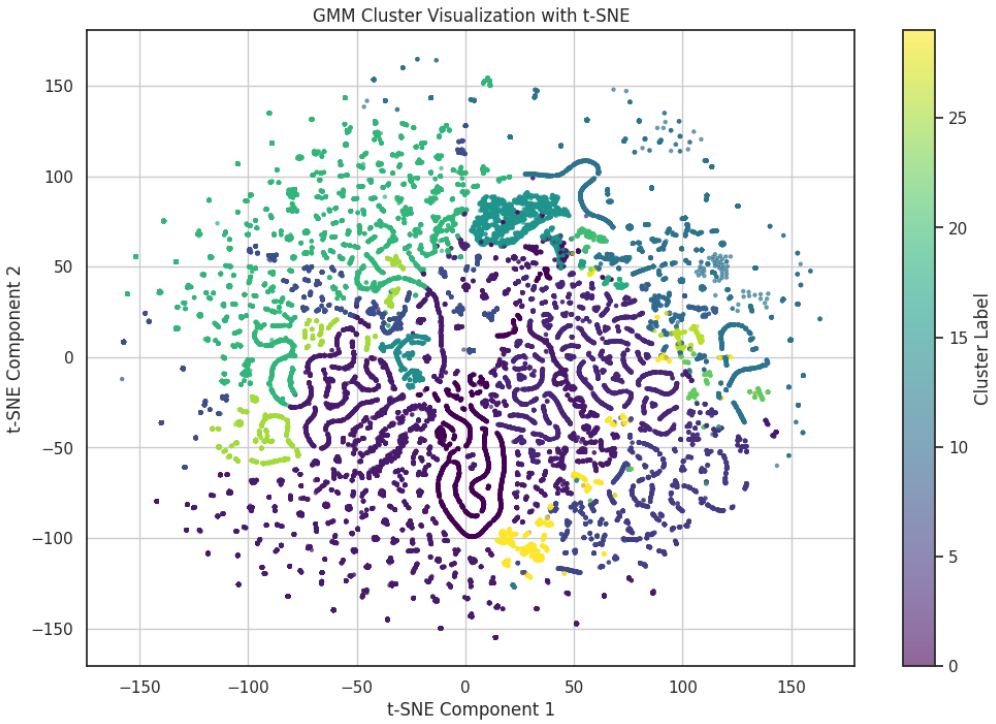


Figure 3.13: GMM Cluster visualization with t-SNE

The analysis of the t-SNE visualisation of the clusters formed by the Gaussian Mixture Model (GMM), shown on figure 3.13 reveals a number of interesting dynamics in the behaviour of the clustering model. Some clusters appear well-defined and distinctly separated, highlighting the GMM's ability to identify groups with well-differentiated characteristics, as demonstrated by the clusters in light green, blue and yellow. These clusters not only show good separation, but also a compactness that suggests high internal homogeneity. In contrast, other regions, particularly those coloured purple, show more overlap and a wider distribution, indicating a difficulty for the model to clearly distinguish between certain data groups.

While some small and concentrated clusters suggest the presence of well-defined data niches, other larger and less delineated clusters might indicate data groups that share cross-cutting characteristics.

3.3 DBSCAN

Introduction Among the various clustering algorithms available, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is particularly useful in situations where the data have a complex spatial structure or contain noise (anomalous data or outliers). Unlike algorithms such as K-means, which require specifying the number of clusters a priori, DBSCAN independently determines the number of clusters based on the density of the data. This makes it particularly suitable for exploratory data analysis.

DBSCAN identifies clusters as areas of high point density separated by areas of low density. One of the main advantages of DBSCAN is its ability to identify clusters of arbitrary shape, unlike methods such as K-means that assume a spherical shape of clusters.

However, DBSCAN also has some disadvantages. The performance of DBSCAN is highly dependent on the choice of ϵ and MinPts parameters, which can be difficult to determine a priori. In addition, the algorithm may be inefficient on large or high-dimensional datasets since it requires the evaluation of distances between all points. Another limitation is that DBSCAN may have difficulty detecting clusters in datasets with varying densities since a single value of ϵ may not be adequate for all regions of the dataset.

DBSCAN uses two fundamental parameters:

- Epsilon (ϵ): The maximum distance within which to search for nearby points to consider a point as part of a cluster. A small ϵ creates many small clusters; a large ϵ can merge distinct clusters.
- MinPts (Minimum Points): The minimum number of points required to form a cluster. A low MinPts generates small and noise-sensitive clusters; a high MinPts requires greater density to form a cluster.

A point is classified as a Core Point if it has at least MinPts points within the distance ϵ, as a Border Point if it is reachable from a core point but has fewer than MinPts points within the distance ϵ, and as a Noise Point if it is neither a core point nor a border point.

Applying DBSCAN Using Default Parameters

```
Number of clusters (including noise):  160
Size of each cluster:  [ 1943    27    91  2795    95    52   455   444     7    12     9
                           6  2012    45    89    12    10  5156    42   140   130    27
                          63   151   517   103    12    20     5     7    29    18   117
                         177   285    39    20     7    22   115     7    46    53    14    74
                         28     6    12     5     6    41    19    17   804  17387    37
                           7     6 160976    13     7    24   740  1221   457    26    25
                          65    36     9  1944    16     8     8     5  1203   317    65
                          117   115     6     8    21     7     5  4124  1542   467    10
                         327   124    32     5     5     9    46    13     8    10     7
                           5    25     6     9    16     5     6    14    14     7     7
                           5     7     9     6     6     8     5     7    15    16    20
                           5     6     5     6     9     6     6     5     7     7     5
                           5     6     9     8  1584  5236  1354     9    52    23     5
                         722    25     5     7     5]
Silhouette: 0.45 RI: 0.83 ARI: 0.22
```

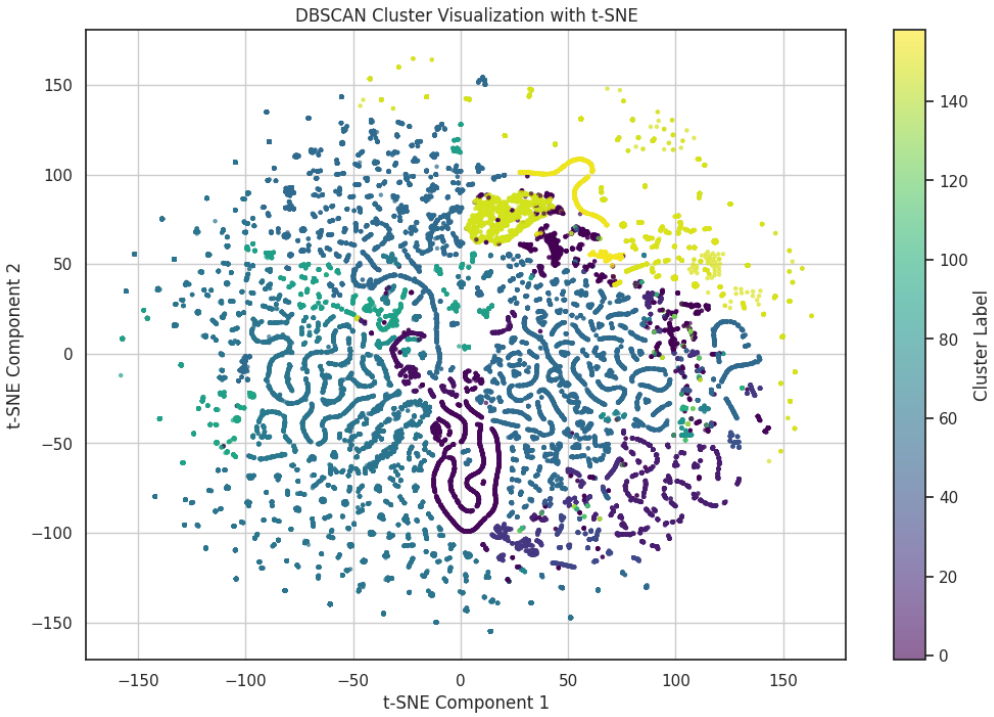


Figure 3.14: DBSCAN Cluster visualization with t-SNE

We chose to apply the DBSCAN without pre-selecting optimal parameters in order to emphasize the importance of parameter choice in the clustering process. Specifically, the analysis produced a total of 160 clusters, a considerably high number that emphasizes an overly fragmented distribution of the data. This phenomenon is further evidenced by relatively low values for both the silhouette coefficient and the Adjusted Rand Index, both indicators of suboptimal clustering quality.

Looking at figure 3.14, regarding the distribution of samples in the various clusters, it can be seen that two clusters predominate, aggregating most of the samples. This situation is not conducive to a balanced and effective representation of the heterogeneity of the data, compromising the effectiveness of the clustering model in capturing the structural subtleties of the analyzed dataset.

Determine Best Parameters of DBSCAN

To refine the selection of the epsilon and min_samples parameters of DBSCAN, we chose to conduct a grid search in two steps. In the first step, we explored a wide range for both parameters: epsilon ranged from 0.2 to 1.3 and min_samples from 20 to 45 increasing by 5 units per step. This allowed us to identify a preliminary area of interest based on cluster formation. Subsequently, in the second step, we narrowed the search by focusing on a more specific range for epsilon from 0.71 to 0.8 in increments of 0.02 and considering only values for min_samples from 30 to 34. This two-stage methodology not only helped us to optimise processing time, but also to focus the analysis on the most promising parameter ranges.

Following this detailed cross-analysis, we selected the parameters with the highest silhouette value, namely min_samples equal to 30 and epsilon equal to 0.77 (see figures 3.15 and 3.16).

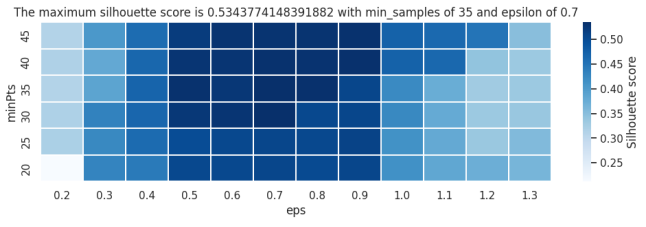


Figure 3.15: Heatmap of Silhouette Scores for Varying MinPts and Epsilon Values (min_samples = 35, $\epsilon = 0.7$)

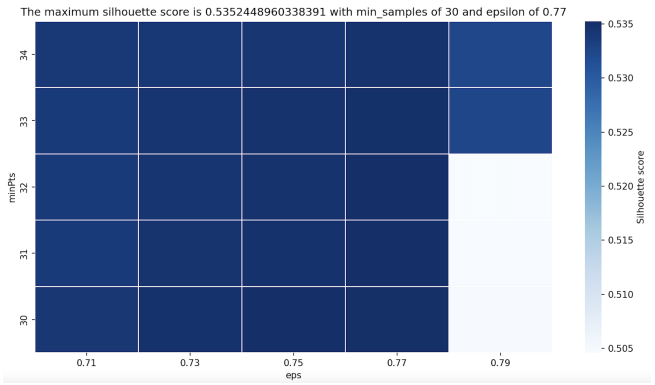


Figure 3.16: Heatmap of Silhouette Scores for Varying MinPts and Epsilon Values (min_samples = 30, $\epsilon = 0.77$)

Applying DBSCAN Using Best Parameters

```
Number of clusters (including noise):  48
Size of each cluster:  [  3187     91   2795     95     52    237    471     78     46   2059
                           330     90    145     47    140    130     59    151    502    103
                           117    177    116     48     53     73    813  17389     38    121
                         16976    740    580     65     38    129     19    115   4134   2009
                           327    161     46   1380   1555   5236     49    927]
Silhouette: 0.54
RI: 0.83
ARI: 0.23
```

ECDF of Number of Clusters

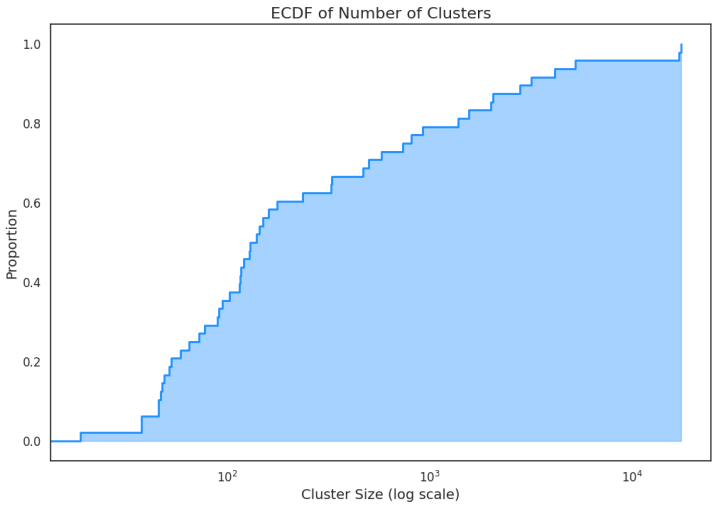


Figure 3.17: ECDF of Number of Clusters

The figure 3.17 shows considerable variability in cluster size, revealing the presence of both many very small clusters and some very large ones. In particular, the initial part of the curve shows a rapid succession of increments, indicating that a large number of clusters contain few points, suggesting possible excessive fragmentation or the presence of many outliers. Toward the end of the curve, the gradual flattening of the ECDF reveals that a few large clusters enclose a significant proportion of the data points.

Cluster Visualization

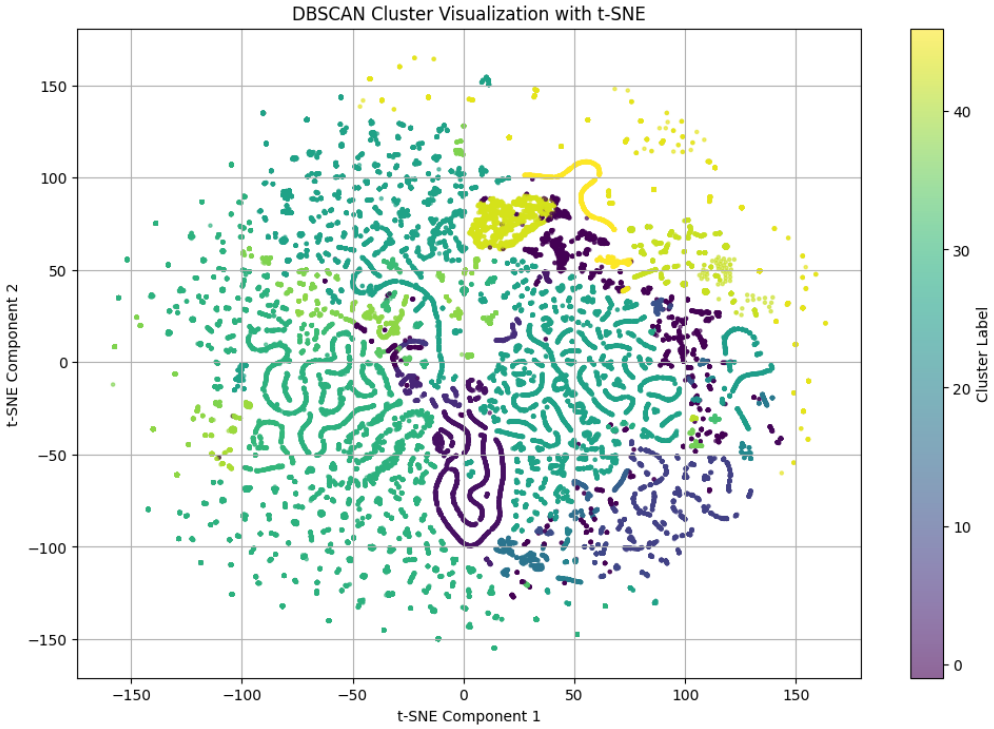


Figure 3.18: Cluster Visualization

From the analysis of the DBSCAN cluster visualization with t-SNE (as can be seen on figure 3.18), key characteristics emerge that outline the effectiveness and peculiarities of this clustering approach. DBSCAN has distinguished a considerable variety of groups, indicating its ability to identify density-based clusters in a complex dataset. The presence of numerous clusters suggests that the method has been able to recognize various data densities, isolating areas of high concentration. Moreover, DBSCAN is effective at signaling noise points, which are data that do not aggregate into dense clusters, highlighting its utility for identifying anomalies or atypical data. These results indicate that the parameters min_samples=30 and $\epsilon = 0.77$ have effectively contributed to delineating complex data structures, although they may require further optimization to refine cluster separation or reduce noise.

We note that the clusters in the t-SNE visualization described in the previous paragraphs resulting from the application of DBSCAN with default parameters appear scattered and less defined with a prevalence of noise characteristic of suboptimal parameters for distinguishing data densities. In contrast, the t-SNE with optimized parameters shows well-separated and distinct clusters. This indicates that a proper selection of values for epsilon and min_samples has significantly improved DBSCAN's ability to identify and separate clusters based on density, reducing noise and enhancing the readability of the data structure.

Section 4 Clusters explainability and analysis

Introduction In this section of the report, we will focus on the analysis and interpretation of the clusters identified in the previous section. The main objectives include the characterization of the clusters in terms of the distribution of features and activity patterns and the formulation of considerations related to the analysis of darknet traffic.

The primary goal is to examine the detected clusters and identify new patterns. This analysis will pursue four main tasks:

1. **Reflection on clusters with respect to GT labels:** We will evaluate whether the clusters reflect the ground truth (GT) labels by calculating the empirical cumulative distribution function (ECDF) of the number of clusters assigned to each class. We will verify the existence of pure clusters in which all elements belong to a single class and analyze whether there is benign traffic with characteristics similar to malicious traffic.

2. **Identification of the most important features in the obtained clusters:** We will use methods that provide feature importance or explainability techniques to identify the most relevant features within the clusters helping us understand which features most influence cluster formation.

3. **Identification of sub-attacks:** We will try to identify any sub-attacks within the clusters by analyzing the features that contribute to this identification. Additionally, we will explore the possibility of identifying new groups or similar clusters and understand the characteristics that form them.

4. **Similarity between attacks:** We will analyze which attacks are most similar to each other and according to which features to better understand the nature of the attacks and identify common patterns in malicious traffic.

The ultimate goal is to provide an in-depth understanding of the detected clusters highlighting the dynamics of darknet traffic and contributing to the improvement of threat detection techniques. For the cluster analysis we will compare the K-means and DBSCAN methods. K-means assumes spherical clusters of similar size and requires the number of clusters to be specified a priori. In contrast, DBSCAN identifies clusters based on point density not requiring the number of clusters a priori and better handling arbitrary shapes and noise. DBSCAN is particularly useful for analyzing darknet traffic characterized by significant variations in data density and the presence of noise offering a more comprehensive and robust view of the patterns in the data.

4.1 K-means

Distribution of clusters in relation to the ground truth To begin with, we visualized the distribution of clusters in relation to the ground truth (GT) labels to gain an initial understanding of how the identified clusters reflect the labels.

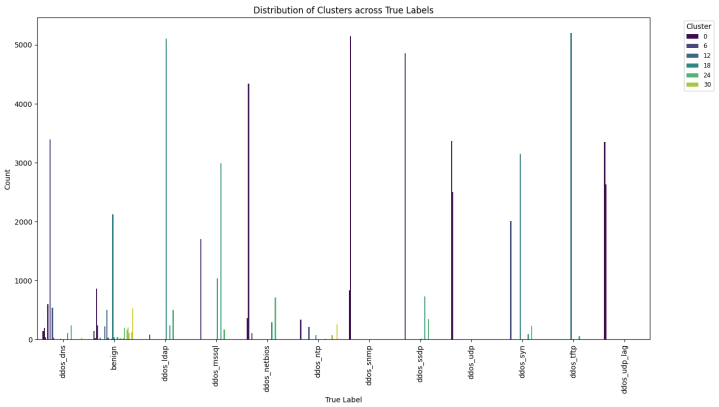


Figure 4.1: Distribution of Clusters across True Labels

The figure 4.1 shows the distribution of various clusters in relation to the ground truth labels. This visualization facilitates the identification of labels composed of a mixture of different clusters as is evident for the benign and ddos_dns labels represented by different clusters.

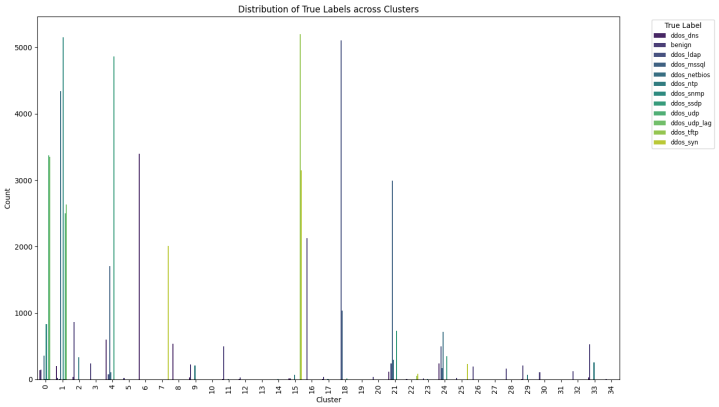


Figure 4.2: Distribution of True Labels across Clusters

The figure 4.2 shows the inverse visualization i.e. how the ground truth labels are distributed with respect to the clusters. This visualization facilitates the identification of clusters that primarily contain samples of a specific label. For example, it is evident that cluster 18 mainly contains samples belonging to the ddos_ldap label or that cluster 6 only contains ddos_dns labels.

Given the large number of clusters and labels this visualization does not allow for capturing all the nuances of interest. Therefore, the distribution of all labels within each individual cluster and the corresponding contingency matrix are presented below to achieve a much clearer understanding.

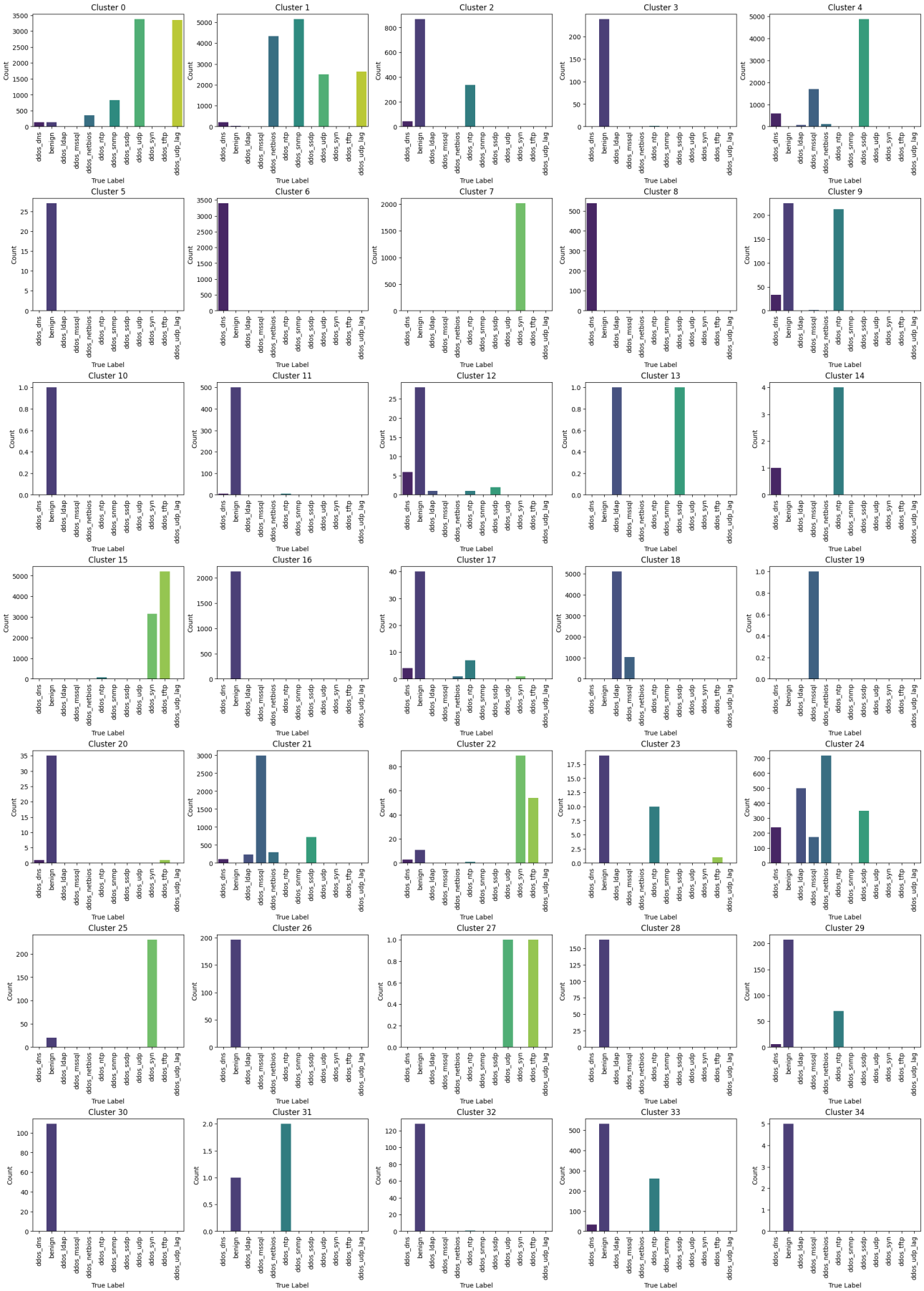


Figure 4.3: Distribution of true labels and their counts for each cluster

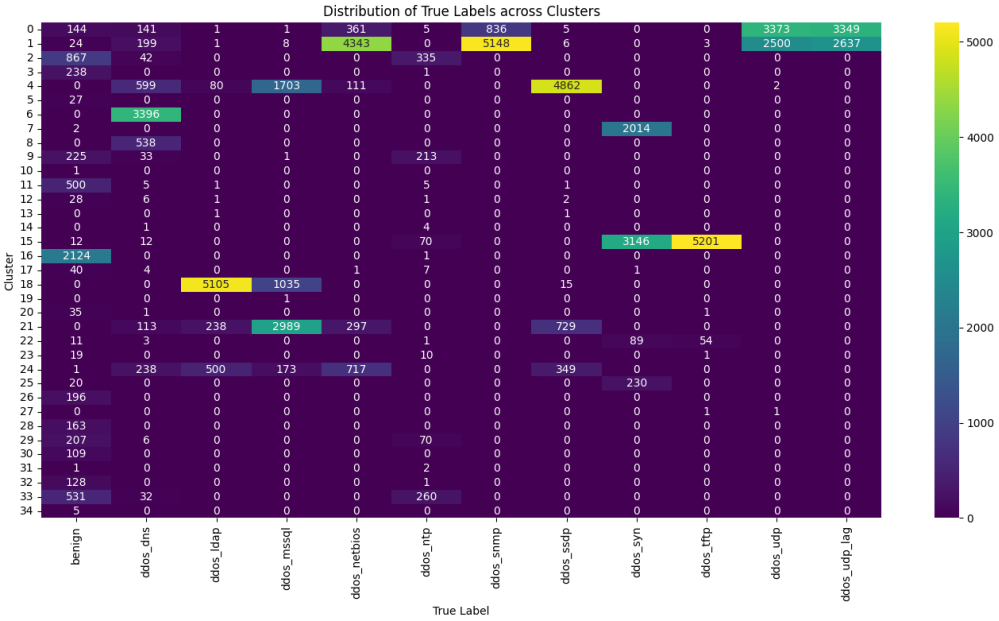


Figure 4.4: Distribution of True Labels across Clusters

With reference to the figures 4.3 and 4.4, a number of considerations can be made.

Table 4.1: Noise (less than 10 samples per cluster)

Cluster
10
13
14
19
27
31
34

Table 4.2: Pure (or nearly pure) clusters

Cluster	Label
3	benign
5	benign
6	ddos_dns
7	ddos_syn
8	ddos_dns
11	benign
12	benign
16	benign
17	benign
20	benign
25	ddos_syn
26	benign
28	benign
30	benign
32	benign

Important note: in all visualizations the noise clusters will still be present but they will be ignored.

As was reasonably expected, the analysis revealed the presence of numerous clusters representing benign traffic. This phenomenon can be explained by considering that generic traffic is inherently diverse characterized by a wide range of attributes that reflect the different ways in which users navigate and interact on the network. This diversity results in a multiplicity of traffic patterns each of which can be distinctly categorized within a specific cluster.

Looking at the figure, it can be seen that some specific clusters such as 2, 9, 23, and 33 exclusively include traffic classified as benign and ddos_ntp type attacks, the latter known to be an underrepresented class in the data. To understand the reason for this distinct aggregation, it is essential to conduct a deeper analysis of the key characteristics that define these clusters.

Investigation of the most relevant characteristics may reveal distinctive patterns or common attributes that justify the coexistence of benign traffic and ddos_ntp attacks within the same clusters.

At this point in the discussion, although it is possible to start making preliminary remarks regarding underrepresented attacks and similarities between the different clusters, it is considered more appropriate to defer detailed analysis of these aspects to later sections of the paper.

Cluster mapping From the initial analysis, it emerged that the clusters more or less tend to reflect the assigned labels. However, to conduct a more meticulous investigation, we adopted a specific approach. Using a technique known as cluster mapping, we aimed to overcome the challenge presented by the coexistence of two or more labels within single clusters as observed for example in cluster 0. The goal was to avoid automatically selecting the most frequent label without considering other significantly represented labels.

For this, we developed a script that analyzes each cluster by identifying the frequencies of the various labels present. The algorithm selects the label with the highest frequency and concurrently identifies and includes all those labels whose frequency is at least 60% of the highest recorded value.

Applying this method to cluster 0, the algorithm selected labels with frequencies of 3373 and 3349 excluding those with frequencies below 60% of 3373.

After defining the functioning of the algorithm, we move on to evaluating the results of this mapping. We use the confusion matrix (Shown on figure 4.5) as an evaluation tool. The confusion matrix allows us to visualize the effectiveness with which the algorithm has categorized traffic in the clusters relative to the original labels thus highlighting the accuracy and any discrepancies in classification. This analysis will provide us with a clear picture of the accuracy of our mapping strategy and help us identify areas that could benefit from further improvements or a different approach.

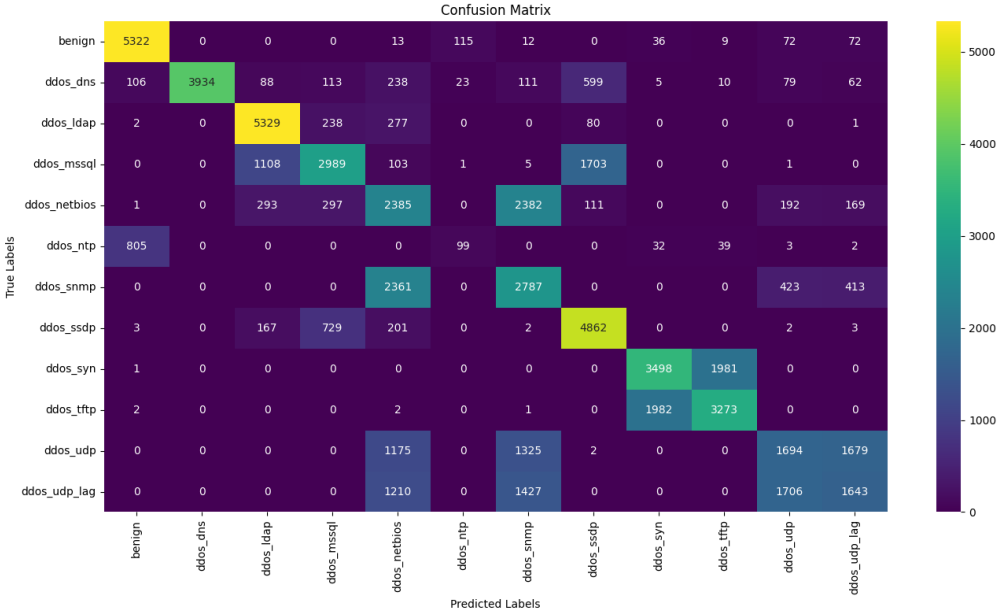


Figure 4.5: Confusion Matrix – Cluster mapping

Table 4.3: Cluster Mapping – Classification Report

Label	Precision	Recall	F1-Score	Support
benign	0.85	0.94	0.89	5651
ddos_dns	1.00	0.73	0.85	5368
ddos_ldap	0.76	0.90	0.83	5927
ddos_mssql	0.68	0.51	0.58	5910
ddos_netbios	0.30	0.41	0.35	5830
ddos_ntp	0.42	0.10	0.16	5894
ddos_snmp	0.35	0.47	0.40	5980
ddos_ssdp	0.61	0.41	0.49	5969
ddos_syn	0.63	0.64	0.63	5260
ddos_tftp	0.42	0.41	0.42	5875
ddos_udp	0.41	0.29	0.34	5986
ddos_udp_lag	0.41	0.27	0.34	5986
Accuracy		0.59 (64220)		
Macro avg	0.59	0.56	0.56	64220
Weighted avg	0.60	0.59	0.58	64220

Cluster	Labels with Weights
0	ddos_udp: 50.18%, ddos_udp_lag: 49.82%
1	ddos_netbios: 45.76%, ddos_snmp: 54.24%
2	benign: 100.00%
3	benign: 100.00%
4	ddos_ssdp: 100.00%
5	benign: 100.00%
6	ddos_dns: 100.00%
7	ddos_syn: 100.00%
8	ddos_dns: 100.00%
9	benign: 51.37%, ddos_ntp: 48.63%
11	benign: 100.00%
12	benign: 100.00%
15	ddos_syn: 37.69%, ddos_tftp: 62.31%
16	benign: 100.00%
17	benign: 100.00%
18	ddos_ldap: 100.00%
20	benign: 100.00%
21	ddos_mssql: 100.00%
22	ddos_syn: 62.24%, ddos_tftp: 37.76%
23	benign: 100.00%
24	ddos_ldap: 41.08%, ddos_netbios: 58.92%
25	ddos_syn: 100.00%
26	benign: 100.00%
28	benign: 100.00%
29	benign: 100.00%
30	benign: 100.00%
32	benign: 100.00%
33	benign: 100.00%

Table 4.4: Clusters to Label Mapping with Weights

```
Clusters Eliminated (frequency < 10):
['14', '34', '31', '13', '27', '19', '10']
```

From the data provided, it is evident that the clusters have been mapped with varying accuracy to different labels.

For example, cluster 0 shows an almost equal distribution between the labels ddos_udp and ddos_udp_lag suggesting a similarity in the characteristic traits between these categories of traffic. On the other hand, clusters like 11 and 17 are clearly identified as benign traffic with a 100% match. However, some clusters like 9 present an almost even balance between contrasting labels such as benign and ddos_ntp indicating the possibility of confusion or overlap in the traffic characteristics or even a strong resemblance.

The 60% accuracy achieved through the k-means clustering algorithm represents an acceptable result considering its inherent limitations especially in complex contexts like network traffic analysis. K-means tends to perform better with homogeneous and simple data.

The similarities observed between various pairs of attacks such as ddos_syn and ddos_tftp and between categories such as benign and ddos_ntp as well as ddos_udp and ddos_udp_lag indicate the presence of common characteristics that can induce confusion in the clustering process. These affinities may arise from similar traffic patterns, overlapping features, or a lack of capability of distance-based algorithms like k-means to distinguish between subtly different patterns.

Therefore, strong similarities are highlighted between ddos_syn and ddos_tftp, benign and ddos_ntp, ddos_udp and ddos_udp_lag, ddos_netbios and ddos_snmp, ddos_ldap and ddos_netbios.

To obtain a visual representation that contrasts the predicted label mapping with the actual labels, we applied the t-SNE technique to both datasets.

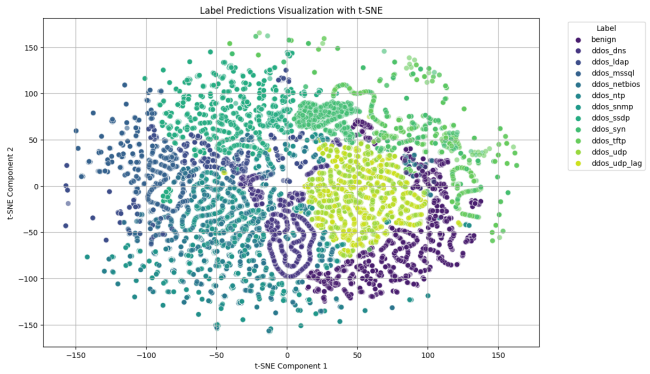


Figure 4.6: Label Predictions Visualization with t-SNE

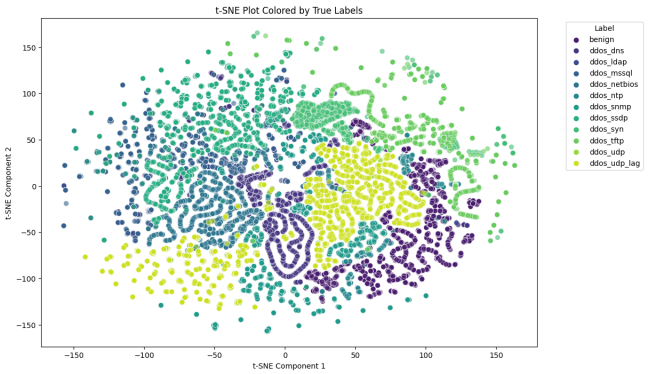


Figure 4.7: t-SNE plot Colored by True Labels

From the visual analysis of the t-SNE graphs (figures 4.6 and 4.7), similarities and overlaps between the classifications of various traffic types clearly emerge. In particular, two cases of misclassification attract attention because of their implications:

- **Misclassification of ddos_udp_lag:** This traffic type is divided into two distinct groupings. The upper portion of ddos_udp_lag is correctly identified but also shows significant overlap with ddos_udp. This suggests that the characteristics of these two traffic types are similar enough to confuse the clustering algorithm. The other clustering of ddos_udp_lag is completely misclassified being identified as a combination of ddos_netbios and ddos_snmp. This observation indicates a more serious problem of distinguishing between features.
- **Misclassification of ddos_ntp:** In this case, we note that at several specific coordinates ddos_ntp traffic is misclassified. At coordinate point (10050) the traffic is classified as benign while at coordinate point (4010) it is identified as ddos_tftp. These misclassifications could result from subtle variations in the traffic patterns of ddos_ntp that make it similar to benign traffic or other types of DDoS attacks such as ddos_tftp.

These are just some of the evidences we wanted to highlight that confirm the data analyzed earlier.

ECDF of number of cluster assigned to each label

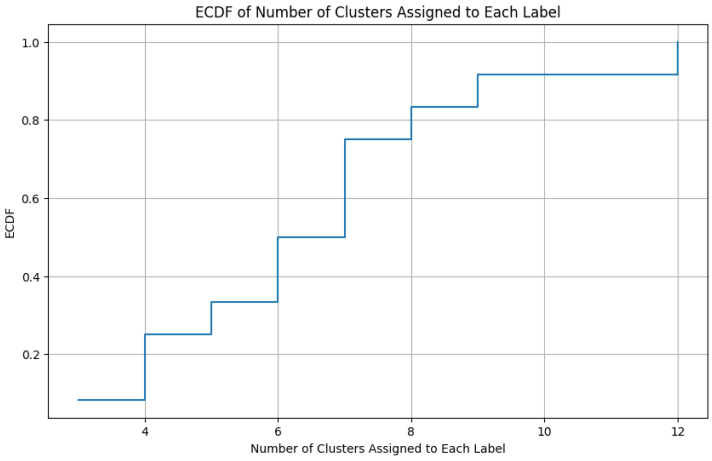


Figure 4.8: ECDF of number of cluster assigned to each label

On figure 4.8, the graph of the empirical cumulative distribution function (ECDF) shows how the number of clusters assigned varies among different true labels (True Labels) in the dataset. The information we can glean is that the median is between 6 and 8 clusters per label with the total range ranging from 4 to 12. About 40 percent of the labels have 6 or fewer clusters assigned which may help to assess the effectiveness of the clustering process.

Feature importance In the context of clustering, assessing the importance of features is critical to understanding how various attributes influence cluster formation. Two effective methods for analyzing this importance are multi-class classification and intra-cluster variable similarity, both of which are useful for refining and interpreting clustering results.

- Multi-class classification (check figure 4.9) treats objects in a given cluster as belonging to one class while those in other clusters as members of a second class. Using classification algorithms such as XGBoost (more faster than other)

combined with interpretation techniques such as SHAP it is possible to identify the contribution of each individual feature to the classification decision. This approach not only clarifies which attributes distinguish one cluster from others but also provides a quantitative basis for assessing the importance of each feature. This process is iterated for each cluster allowing detailed and specific analysis.

- Intra-cluster variable similarity method (check figures 4.10 and 4.11) focuses on how much each variable contributes to the internal cohesion of the cluster. By averaging the similarities between each feature and its cluster centroid for each variable a direct measure of the impact of each feature on cluster compactness is obtained. The use of the k-means WCSS_min technique which measures the sum of the inner squares of the cluster for each feature makes it possible to highlight those variables that contribute most to minimizing internal variance suggesting greater significance in aggregating the data.

The combined implementation of these methods provides a thorough and balanced understanding of the importance of features in clustering.

Figure 4.9: Multi-class classification

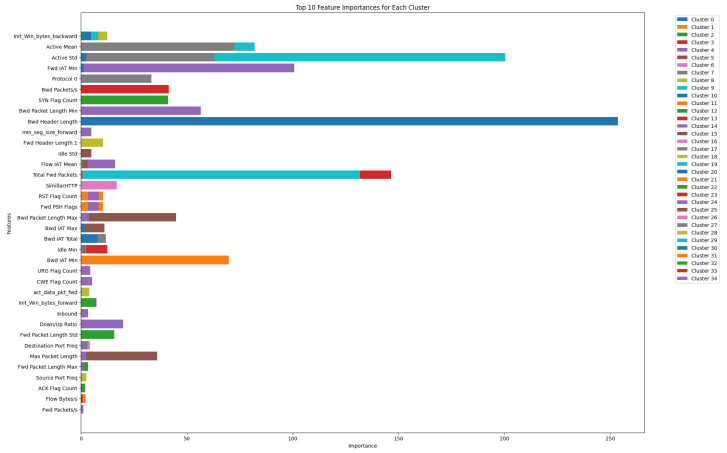


Figure 4.10: Intra-cluster variable similarity (I)

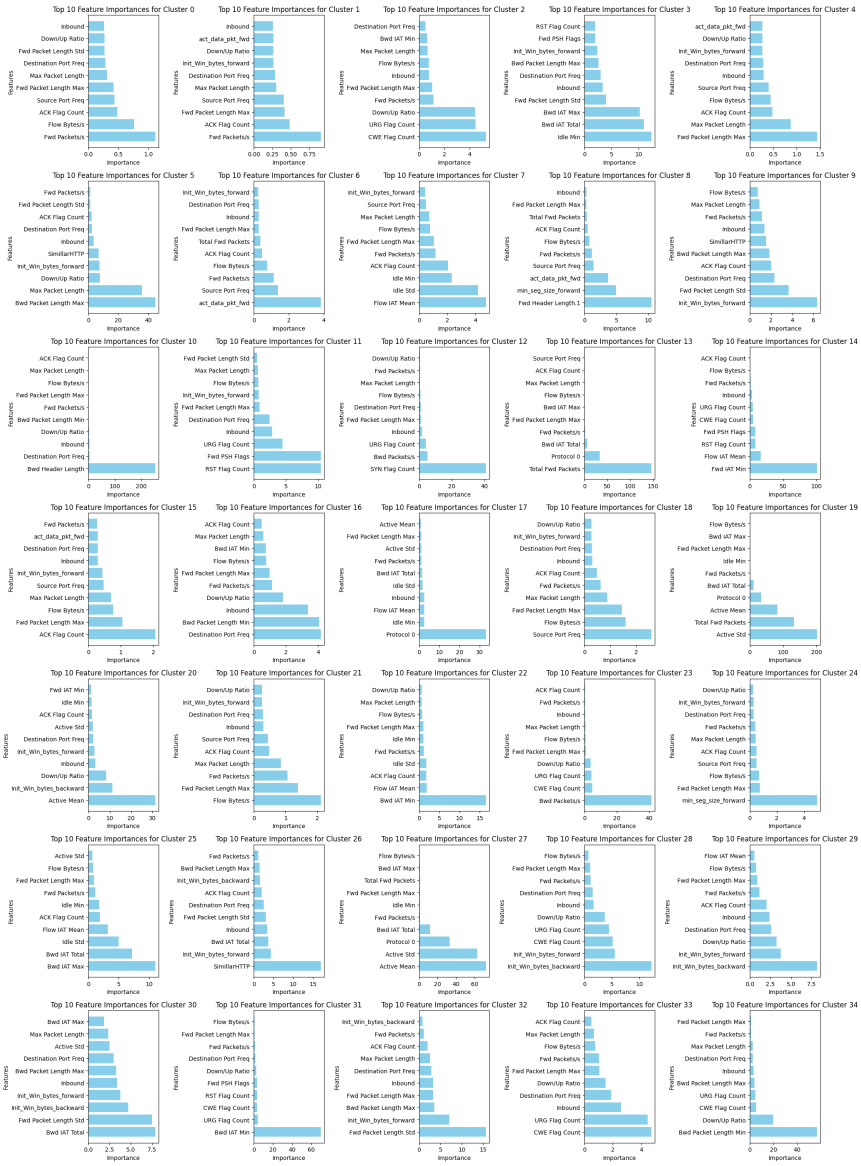


Figure 4.11: Intra-cluster variable similarity (II)

Before proceeding with the analysis, it is interesting to note how these two methods return very similar results. In fact, one only has to look at a few clusters to see that the most important features selected by the models are the same. Now we can combine all the previously collected data with the knowledge of the weights that the features have in the individual clusters in order to make the similarities and misclassifications in our k-means clustering algorithm meaningful.

Important note: in these visualizations the noise clusters will be present because they could not be removed; when reading the graphs it is good to ignore them.

Detection of Sub-Attack and Similarities The next goal is to identify any sub-attacks within the clusters analyzing the distinguishing characteristics and explore the formation of new similar groups or clusters based on these characteristics. In addition, a similarity analysis will be conducted among the attacks identifying common features in order to better understand the nature of the attacks and detect recurring patterns in malicious traffic.

Cluster
10
13
14
19
27
31
34

Table 4.5: Noise clusters with less than 10 samples

Table 4.6: Pure (or nearly pure) clusters

Cluster	Label
3	benign
5	benign
6	ddos_dns
7	ddos_syn
8	ddos_dns
11	benign
12	benign
16	benign
17	benign
20	benign
25	ddos_syn
26	benign
28	benign
30	benign
32	benign

Cluster
0
1
2
4
9
15
18
21
22
23
24
29
33

Table 4.7: Clusters to be classified

- **Cluster 0:** we have seen to be mostly composed of ddos_udp and ddos_udp_lag. Looking at the most important features we notice that predominating are Fwd Packets/s and Flow Bytes/s. If we look at the graph obtained with t-SNE we notice that the ddos_udp and ddos_udp_lag clusters overlap in a sparse manner so in this case we can say the kmeans algorithm is not able to capture the subtleties such that these two labels can be distinguished well.
- **Cluster 1:** this cluster is very peculiar because it contains so many samples within it and because it identifies as many as 4 labels. Nevertheless, from the feature analysis and also by viewing the t-SNE it is difficult to understand why the algorithm magnifies this cluster.
- **Clusters 2, 9, 23, 29, 33:** these clusters are all characterized by the predominant presence of two labels benign and ddos_ntp. We look at the most important features and notice a very interesting recurrence in fact all clusters have as main features: CWE Flag Count, URG Flag Count, Down/Up Ratio, Fwd Packet Length Max, Init_Win_bytes_forward, Inbound. This is very important information because it makes us realize that there is a strong similarity between the two labels.
- **Cluster 4:** this cluster mostly represents ddos_ssdp but to a small extent also ddos_mssql. The relationship between the two is not seen in any other cluster excluding the possibility of strong similarities between the two.
- **Clusters 15, 22:** these two clusters represent the ddos_syn and ddo_tftp labels. We also note here a similarity between the most important features although less pronounced than in benign and ddos_ntp.
- **Cluster 18:** this cluster consists mainly of ddos_ldap although other labels appear in significantly smaller and negligible amounts.
- **Cluster 21:** this cluster consists mainly of ddos_mssql although other labels appear in significantly smaller and negligible amounts.
- **Cluster 24:** this cluster has within it with non-negligible amounts 5 labels so we can say that it does not carry important information such as to identify strong similarities or under attacks.

Regarding sub-attacks, we can look at the contingency matrix vertically and notice that:

- ddos_syn appears with many samples in two main clusters so the fact that are 1 label strongly represented by two clusters makes us think that there are two sub-attacks.
- The same reasoning applies to ddos_udp and ddos_udp_lag only that the fact that cluster 1 is so large makes us more insecure.

4.2 DBSCAN

Distribution of clusters in relation to the ground truth As already said, DBSCAN identifies clusters based on the density of points with points in high-density regions forming clusters and points in low-density regions being labeled as noise. The `eps` parameter (0.77 in our case) sets the maximum distance between points in a cluster and `min_samples` (30) sets the minimum number of points required to form a cluster. In DBSCAN, any point that does not belong to any cluster is labeled as -1. These points are considered noise because they are not within a dense region of points. By labeling and removing noise points, DBSCAN helps in forming more accurate and meaningful clusters. Noise points, if included in clusters, can distort the true structure of the data.

As already done with k-means, the figure 4.12 shows the distribution of various clusters in relation to the ground truth labels.

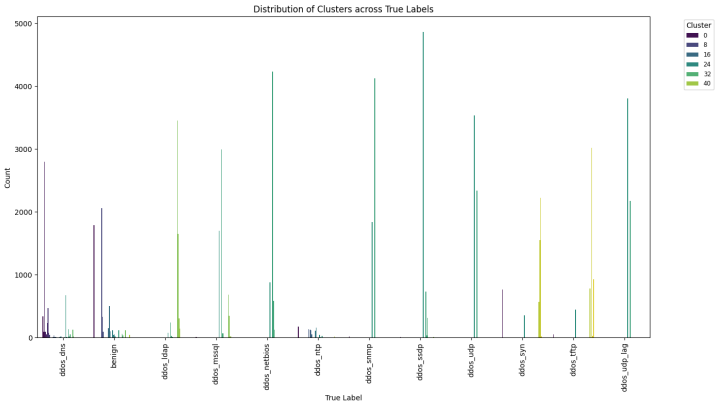


Figure 4.12: Distribution of Clusters across True Labels

Mirroring the analysis with K-means, the labels composed of a mix of different clusters again are benign and ddos_dns but with the addition of ddos_ntp which however shows a significant decrease in the number of samples involved (y-axis) compared with the first two.

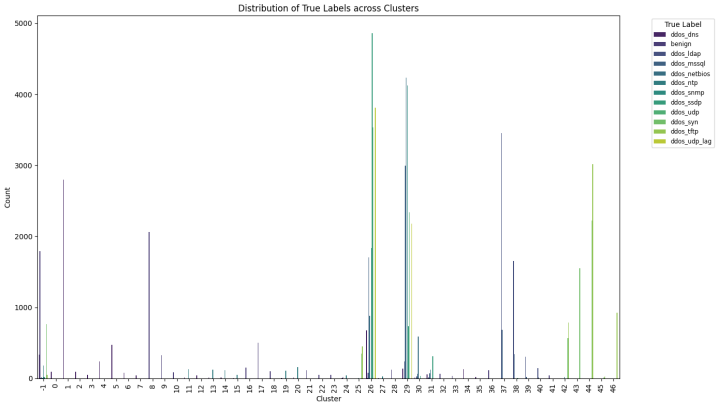


Figure 4.13: Distribution of True Labels across True Labels

In the figure 4.13, the one related to the inverse visualization, sizable clusters that contain only one "pure" label are: cluster 1 that contains only “ddos_dns” labels, cluster 8 that contains only “benign” ones, and cluster 43 that contain only “ddos_syn”.

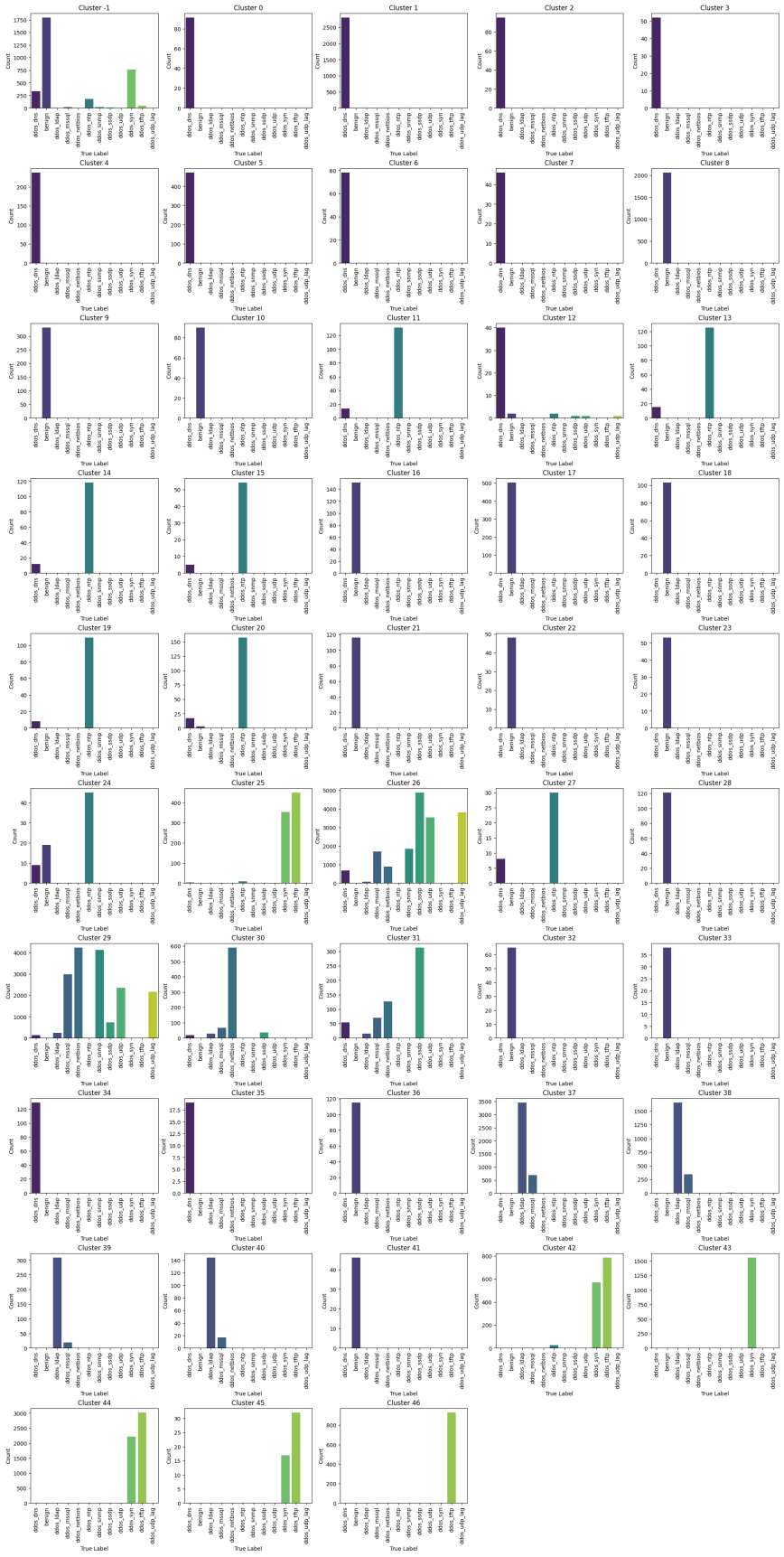


Figure 4.14: Bar charts showing the distribution of true labels within each DBSCAN cluster from -1 to 46, illustrating cluster purity and label composition

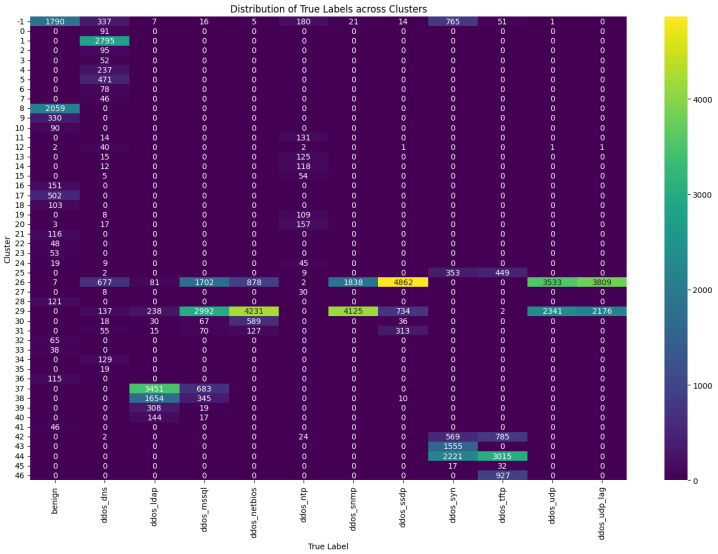


Figure 4.15: Distribution of True Labels across Clusters

The contingency table shows the distribution of true labels (see figures 4.14 and 4.15) within each cluster including the noise points. We proceed as we did with K-means.

Cluster	Label
0	ddos_dns
1	ddos_dns
2	ddos_dns
3	ddos_dns
4	ddos_dns
5	ddos_dns
6	ddos_dns
7	ddos_dns
8	benign
9	benign
10	benign
11	ddos_ntp
12	ddos_dns
13	ddos_ntp
14	ddos_ntp
15	ddos_ntp
16	ddos_ntp
17	benign
18	benign
19	benign
20	ddos_ntp
21	benign
22	benign
23	benign
24	ddos_ntp
27	ddos_ntp
28	benign
32	benign
33	benign
34	ddos_dns
35	ddos_dns
36	benign
41	benign
43	ddos_syn
46	ddos_tftp

Table 4.8: Pure (or nearly pure) clusters

It is important to note that there are two clusters (26 and 29) that capture an exceptionally high number of samples, highlighting the model's difficulty in identifying well-defined clusters. Further in-depth analysis on this matter will be conducted subsequently.

Now, to allow the analysis to focus on well-defined clusters and have an interpretation of the results is easier and more reliable, the noise has been removed from the DataFrame.

Given the relative frequencies of true labels within each cluster, a probabilistic approach was adopted to assign labels to the clusters. This process involved several key steps:

- **Calculating Relative Frequencies:** A contingency table was created from the DataFrame to show the distribution of true labels within each cluster. The relative frequencies of these labels were then calculated, providing a distribution that indicated how likely each label was to appear in a given cluster.
- **Filtering and Normalizing Probabilities:** To ensure meaningful probabilistic assignment, only labels that represented at least 60% of the maximum frequency within a cluster were considered. These probabilities were then normalized to sum to one.
- **Assigning Labels Probabilistically:** For each cluster, labels were assigned based on the filtered and normalized probabilities. This method allowed for a nuanced and statistically informed assignment of labels, particularly useful for clusters containing a mix of true labels.
- **Evaluation of Refined Clusters:** To assess the performance of the probabilistic label assignment, the confusion matrix below was created to compare the true labels with the predicted labels assigned probabilistically. This matrix provided a detailed view of the clustering performance, highlighting areas of agreement and discrepancy between the true and assigned labels.

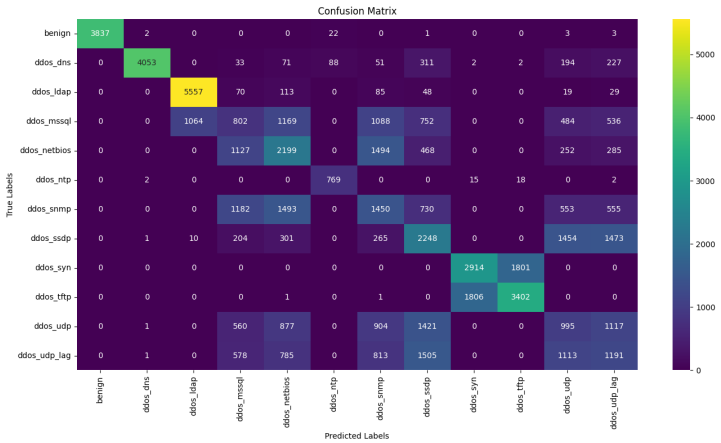


Figure 4.16: Confusion Matrix

From figure 4.16, it is evident that certain clusters, such as those labeled as benign, ddos_dns, and ddos_ldap, exhibit high purity with a significant majority of points correctly labeled. In contrast, other clusters have been assigned labels with varying degrees of accuracy, indicating a mix of true labels within those clusters.

Table 4.9: Classification Report

	Precision	**Recall**	**F1-Score**	**Support**
benign	1.00	0.99	1.00	3868
ddos_dns	1.00	0.81	0.89	5032
ddos_ldap	0.84	0.94	0.89	5921
ddos_mssql	0.18	0.14	0.16	5895
ddos_netbios	0.31	0.38	0.34	5825
ddos_ntp	0.87	0.95	0.91	806
ddos_snmp	0.25	0.26	0.26	5946
ddos_ssdp	0.30	0.38	0.34	5956
ddos_syn	0.61	0.61	0.61	4715
ddos_tftp	0.64	0.65	0.64	5210
ddos_udp	0.21	0.18	0.19	5875
ddos_udp_lag	0.22	0.20	0.21	5986
accuracy			0.48	61052
macro avg	0.54	0.54	0.54	61052
weighted avg	0.48	0.48	0.48	61052

Cluster	Labels with Weights
0	ddos_dns: 100.00%
1	ddos_dns: 100.00%
2	ddos_dns: 100.00%
3	ddos_dns: 100.00%
4	ddos_dns: 100.00%
5	ddos_dns: 100.00%
6	ddos_dns: 100.00%
7	ddos_dns: 100.00%
8	benign: 100.00%
9	benign: 100.00%
10	benign: 100.00%
11	ddos_ntp: 100.00%
12	ddos_dns: 100.00%
13	ddos_ntp: 100.00%
14	ddos_ntp: 100.00%
15	ddos_ntp: 100.00%
16	ddos_ntp: 100.00%
17	benign: 100.00%
18	benign: 100.00%
19	benign: 100.00%
20	benign: 100.00%
21	benign: 100.00%
22	benign: 100.00%
23	benign: 100.00%
24	ddos_ntp: 100.00%
25	ddos_syn: 44.01%, ddos_tftp: 55.99%
26	ddos_ssdp: 39.84%, ddos_udp: 28.95%, ddos_udp_lag: 31.21%
27	ddos_ntp: 100.00%
28	benign: 100.00%
29	ddos_mssql: 26.37%, ddos_netbios: 37.28%, ddos_snmp: 36.35%
30	ddos_netbios: 100.00%
31	ddos_ssdp: 100.00%
32	benign: 100.00%
33	benign: 100.00%
34	ddos_dns: 100.00%
35	ddos_dns: 100.00%
36	benign: 100.00%
37	ddos_ldap: 100.00%
38	ddos_ldap: 100.00%
39	ddos_ldap: 100.00%
40	ddos_ldap: 100.00%
41	benign: 100.00%
42	ddos_syn: 42.02%, ddos_tftp: 57.98%
43	ddos_syn: 100.00%
44	ddos_syn: 42.42%, ddos_tftp: 57.58%
45	ddos_tftp: 100.00%
46	ddos_tftp: 100.00%

Table 4.10: Clusters to Label Mapping with Weights

This classification report provides a comprehensive performance evaluation, indicating high precision and recall for certain labels such as benign, ddos_dns, and ddos_ldap while showing lower performance for labels such as ddos_mssql, ddos_snmp, and ddos_udp. Specifically, the benign label achieved near-perfect precision and recall, reflecting the model's strong ability to correctly identify benign instances with minimal false positives and negatives. The ddos_dns and ddos_ldap labels also performed well with high precision and recall, suggesting that these types of attacks were distinctly recognized by the clustering model. Conversely, labels like ddos_mssql, ddos_snmp, and ddos_udp exhibited significantly lower performance with both precision and recall values, indicating higher rates of misclassification. This disparity suggests potential overlaps or confusion among these attack types within the feature space. The overall accuracy of the model was 48%, which while moderate, highlights the challenges in clustering and classifying complex datasets with multiple attack types. The macro average F1-score of 0.54 and the weighted average F1-score of 0.48 reflect the model's varied performance across different

labels, with the weighted average taking into account the support (number of true instances) for each label. In summary, DBSCAN performed significantly worse than K-means.

Additionally, the table displaying the weights of labels for each cluster shows a high degree of certainty for many clusters, such as those entirely composed of benign or ddos_dns labels. For instance, clusters 10, 16, 17, 18, 21, 22, 23, 28, 32, 33, 36, 41, and 8 were all composed entirely of benign points, indicating a strong uniform distribution and high confidence in the label assignment for these clusters. Similarly, clusters 0, 1, 2, 3, 4, 5, 6, 7, 12, and 34 were entirely composed of ddos_dns points, reflecting a clear and distinct clustering of this attack type.

However, some clusters contained a mix of labels, reflecting the probabilistic nature of the assignment method and the inherent overlap in feature space among certain attack types. For example, clusters 25, 42, and 44 all contained a mix of ddos_syn and ddos_tftp labels with weights of 44.01% and 55.99%, 42.02% and 57.98%, and 42.42% and 57.58% respectively. The similarities observed across these clusters indicate the presence of common characteristics that can induce confusion in the clustering process. The nearly identical distributions of ddos_syn and ddos_tftp labels in these clusters suggest that these attack types share significant similarities in the dataset's feature space. This overlap could be due to several factors, including feature overlap, where the features used for clustering might not be distinctive enough to separate ddos_syn from ddos_tftp, as both attack types might exhibit similar behavior in terms of packet size, frequency, or other network characteristics. Additionally, similar traffic patterns might be followed by ddos_syn and ddos_tftp attacks, making it difficult for the clustering algorithm to differentiate between them. For instance, both might target the same network vulnerabilities or use comparable methods to overwhelm network resources.

Cluster 26 displayed a more complex mix with ddos_ssdp, ddos_udp, and ddos_udp_lag labels distributed at 39.84%, 28.95%, and 31.21% respectively, while cluster 29 exhibited a mixed distribution with ddos_mssql, ddos_netbios, and ddos_snmp labels at 26.37%, 37.28%, and 36.35% respectively.

This mapping of clusters to labels with their respective weights provides a view of the reliability and distribution of labels within the clusters. Clusters with uniform distributions demonstrate the model's effectiveness in distinctly identifying certain attack types or benign traffic. In contrast, mixed clusters reveal areas where the feature space might be shared among different labels.

Further analysis will be conducted to investigate these findings in more detail.

To obtain a visual representation that contrasts the predicted label mapping with the actual labels, we applied the t-SNE technique to both of them (see figures 4.17 and 4.18)

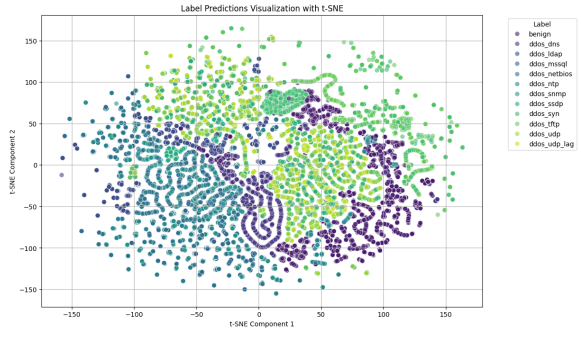


Figure 4.17: Label Predictions Visualization with t-SNE

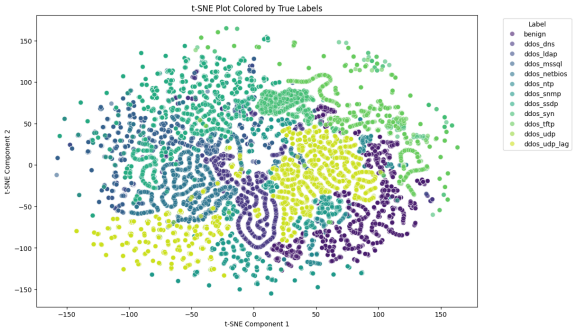


Figure 4.18: t-SNE Plot Colored by True Labels

By comparing these plots, we can identify areas where DBSCAN performed well and where it encountered difficulties. In both plots, benign traffic (colored purple) appears as distinct and relatively compact clusters. This indicates that DBSCAN was able to effectively identify benign traffic, which is characterized by its uniform behavior, making it easier to separate from other types of traffic. The clusters of ddos_dns and ddos_ldap are also well-defined in both the predicted and true label plots. DBSCAN successfully captured these attack types, which suggests that they have distinctive features that the algorithm can easily recognize.

Areas of Misclassification The t-SNE plot of predicted labels shows significant overlap between ddos_mssql, ddos_snmp, and ddos_udp. This overlap is less pronounced in the plot of true labels, indicating that DBSCAN struggled to differentiate between these types of attacks. This misclassification likely arises from similar features in these attacks, leading to confusion in the clustering process.

Clusters for ddos_syn and ddos_tftp exhibit substantial overlap in the predicted labels plot. The near-identical distribution in the predicted plot indicates common characteristics that might not be adequately differentiated by the current feature set.

ECDF of number of clusters assigned to each label

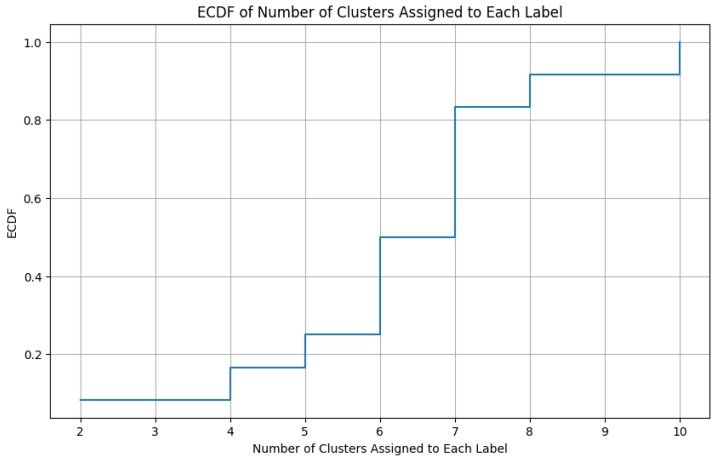
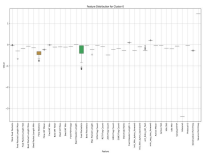


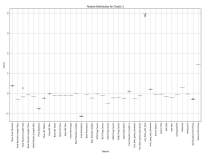
Figure 4.19: ECDF of Number of Clusters Assigned to Each Label

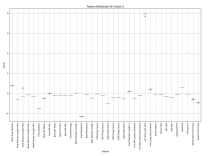
On figure 4.19 in shown that approximately 20% of the labels are assigned to as few as 2 clusters while around 60% of the labels are assigned to 6 or fewer clusters. This suggests that the majority of the labels are distributed among a moderate number of clusters.

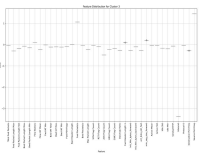
Feature importance As done with k-means, the two methods used for feature analysis are multi-class classification and intra-cluster variable similarity.

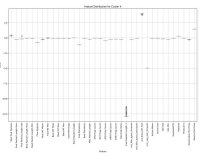
Intra-cluster variable similarity The intra-cluster variable similarity analysis using box plots for each cluster provides a detailed visualization of how different features vary within each cluster. This helps in identifying the most important features that contribute to the clustering process, understanding the distinct characteristics of each cluster, and pinpointing areas where the clustering algorithm might need improvement.

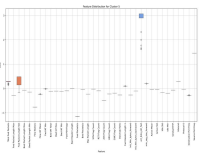

Feature Distribution for Cluster 0

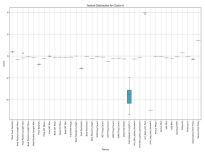

Feature Distribution for Cluster 1

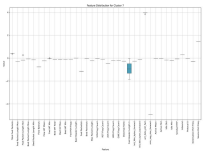

Feature Distribution for Cluster 2

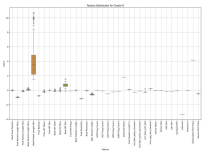

Feature Distribution for Cluster 3

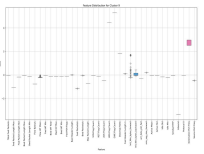

Feature Distribution for Cluster 4

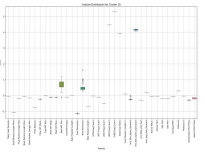

Feature Distribution for Cluster 5

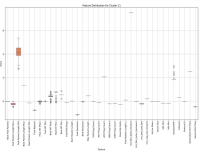

Feature Distribution for Cluster 6

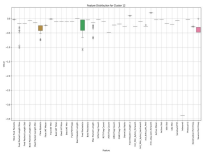

Feature Distribution for Cluster 7

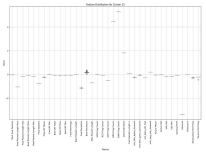

Feature Distribution for Cluster 8

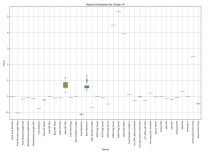

Feature Distribution for Cluster 9

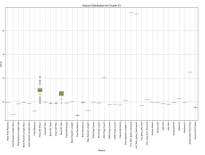

Feature Distribution for Cluster 10

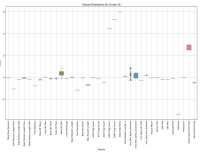

Feature Distribution for Cluster 11

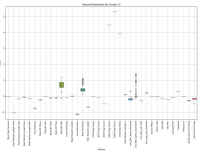

Feature Distribution for Cluster 12

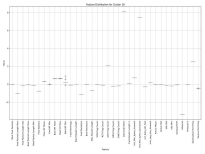

Feature Distribution for Cluster 13

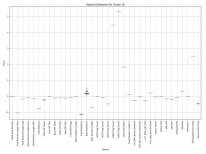

Feature Distribution for Cluster 14

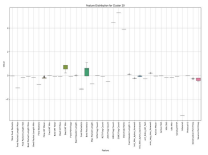

Feature Distribution for Cluster 15

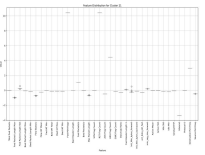

Feature Distribution for Cluster 16

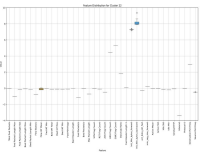

Feature Distribution for Cluster 17

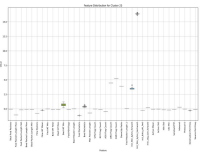

Feature Distribution for Cluster 18

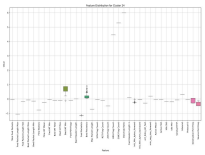

Feature Distribution for Cluster 19

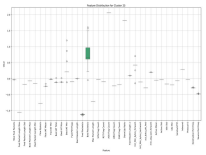

Feature Distribution for Cluster 20

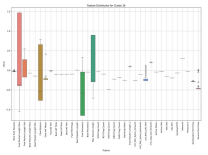

Feature Distribution for Cluster 21

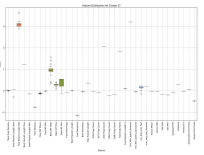

Feature Distribution for Cluster 22

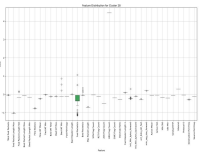

Feature Distribution for Cluster 23

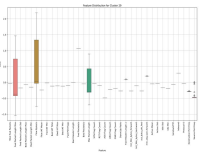

Feature Distribution for Cluster 24

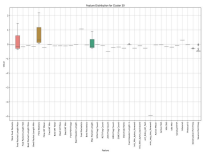

Feature Distribution for Cluster 25

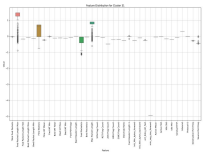

Feature Distribution for Cluster 26

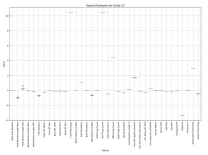

Feature Distribution for Cluster 27

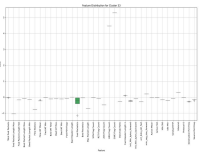

Feature Distribution for Cluster 28

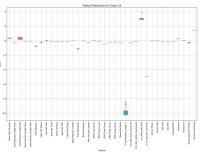

Feature Distribution for Cluster 29

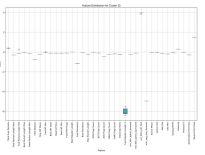

Feature Distribution for Cluster 30

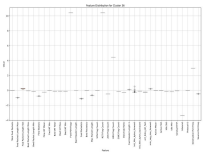

Feature Distribution for Cluster 31

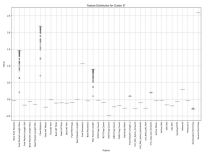

Feature Distribution for Cluster 32

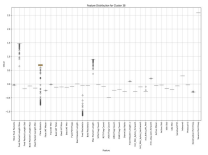

Feature Distribution for Cluster 33

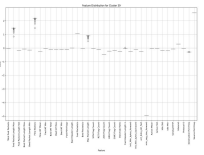

Feature Distribution for Cluster 34


Feature Distribution for Cluster 35


Feature Distribution for Cluster 36


Feature Distribution for Cluster 37


Feature Distribution for Cluster 38


Feature Distribution for Cluster 39

Feature Distribution for Cluster 40

Feature Distribution for Cluster 41

Feature Distribution for Cluster 42

Feature Distribution for Cluster 43

Feature Distribution for Cluster 44

Feature Distribution for Cluster 45

Feature Distribution for Cluster 46

Figure 4.20: Feature Distributions for Clusters 0-46

Multi-class classification Also here, as done in k-means, the combination of SHAP with XGBoost was adopted to identify the contribution of each individual feature to the classification decision (see figure 4.21).

Figure 4.21: SHAP value plots displaying the impact on model output on the x-axis and the features on the y-axis, organized by different clusters

As with K-Means, the two methods also return consistent results here. Indeed, a quick glance at a few clusters reveals that the most significant features chosen by both models are identical. We can now integrate all the previously gathered data with the understanding of feature weights within the individual clusters to highlight the similarities and address the

misclassifications in our DBSCAN clustering algorithm effectively.

Detection of Similarities The next step is a similarity analysis that will be conducted among the attacks, identifying common features in order to better understand the nature of the attacks and detect recurring patterns in malicious traffic.

Cluster	Label
0	ddos_dns
1	ddos_dns
2	ddos_dns
3	ddos_dns
4	ddos_dns
5	ddos_dns
6	ddos_dns
7	ddos_dns
8	benign
9	benign
10	benign
11	ddos_ntp
12	ddos_dns
13	ddos_ntp
14	ddos_ntp
15	ddos_ntp
16	ddos_ntp
17	benign
18	benign
19	benign
20	ddos_ntp
21	benign
22	benign
23	benign
24	ddos_ntp
27	ddos_ntp
28	benign
32	benign
33	benign
34	ddos_dns
35	ddos_dns
36	benign
41	benign
43	ddos_syn
46	ddos_tftp

Table 4.11: Pure (or nearly pure) clusters (previously identified)

Cluster
25
26
29
30
31
37
38
39
40
42
44
45

Table 4.12: To be classified

- **Clusters 25, 42, 44, 45:** they are all characterized by an almost identical distribution of labels between ddos_syn and ddos_tftp. As previously analyzed, the nearly identical distributions of ddos_syn and ddos_tftp labels in these clusters suggest that these attack types share significant similarities in the dataset's feature space. By closely examining the cluster boxplots (intra-cluster) and the SHAP plots, we observe that the most important shared features are ACK flag count, Flow IAT Mean, and FWD Packets/s. Cluster 45 is also included in this discussion albeit with many fewer samples involved. Furthermore, the t-SNE graph reveals that the ddos_syn and ddos_tftp clusters overlap in a sparse manner, indicating that the DBSCAN algorithm struggled to distinguish between these attack types effectively.
- **Clusters 25, 42, 44, 45**: they are all characterized by an almost identical distribution of labels between ddos_syn and ddos_tftp. As previously analyzed, the nearly identical distributions of ddos_syn and ddos_tftp labels in these clusters suggest that these attack types share significant similarities in the dataset's feature space. By closely examining the cluster boxplots (intra-cluster) and the SHAP plots, we observe that the most important shared features are ACK flag count, Flow IAT Mean, and FWD Packets/s. Cluster 45 is also included in this discussion albeit with many fewer samples involved. Furthermore, the t-SNE graph reveals that the ddos_syn and ddos_tftp clusters overlap in a sparse manner, indicating that the DBSCAN algorithm struggled to distinguish between these attack types effectively.

- **Clusters 26, 29:** For these two clusters, the similarity in the contingency table (see figure 4.22) values suggests a commonality in the feature distributions within these clusters. The provided boxplots and SHAP graphs for Clusters 26 and 29 visually reinforce this observation, indicating that these clusters share similar feature distributions. Specifically, features like Total Fwd Packets, Max Packet Length, Flow Bytes/s, FWD Packets/s, and Source Port Frequency exhibit similar distributions in both clusters, further demonstrating their significance.

- **Clusters 30, 31:** These clusters have an almost identical distribution of the percentage between the labels (both containing ddos_syn and ddos_tftp at 44.01% and 55.99% for one cluster and 42.02% and 57.98% for the other). Examining the features, both the boxplots and SHAP graphs confirm this strong similarity, highlighting the same important features: Fwd Packet Length Max, Flow Bytes/s Max, Packet Length, Fwd Header Length.1, and Source Port Frequency. The similarity in feature importance and distribution indicates that the underlying characteristics of the attacks within these clusters may overlap significantly. For instance, the flow and packet metrics exhibit similar patterns for ddos_syn and ddos_tftp attacks, making it challenging for the clustering algorithm to differentiate between them based solely on these features. As evidence of this, looking at the t-SNE graph, it is noticeable how the two labels overlap a lot.

- **Clusters 37, 38, 39, 40:** The boxplots and SHAP graphs for Clusters 37, 38, 39, and 40 visually reinforce the observation that these clusters share similar feature distributions. The boxplots show that features like Source Port Frequency and Flow Bytes/s have similar distributions across these clusters, while the SHAP graphs highlight the impact of these features on the model's output, further demonstrating their significance.

Figure 4.22: Clusters 26 and 29 similarities

Conclusion

In conclusion, this project successfully demonstrated the potential of machine learning models in the classification of network traffic, distinguishing in particular between benign traffic and DDoS (Distributed Denial of Service) attacks. By meticulously following a structured approach, we achieved important milestones in understanding, processing and analysing network data.

The initial data exploration and pre-processing phase was crucial in laying the foundation for effective model training. By normalising features and resolving dimensionality issues, we ensured that our models could learn from the data efficiently and effectively. This phase also involved removing highly correlated features, which helped to reduce redundancy and improve model performance.

In the supervised learning phase, different classification models were trained and evaluated to identify the most effective approach to detect and classify network traffic. Through rigorous performance analysis and tuning of hyperparameters, we were able to improve the accuracy and reliability of our models, ensuring their robustness in real-world scenarios.

The unsupervised learning phase added a further layer of understanding by using clustering algorithms to discover natural groupings in network data. This approach was instrumental in identifying patterns that were not immediately obvious, providing a deeper understanding. By examining the characteristics and distributions of these clusters, we gained valuable insights into the nature of DDoS attacks and the behaviour of benign traffic.

The techniques and results of this project can be exploited to improve the security and resilience of network systems against DDoS attacks, helping to safeguard digital infrastructures.

Aresca Massimo (328743), Carcagnì Andrea (331618),
de Maria Giovanni (331031), Rizzo Leonardo (328764)
Turin, June 14, 2024