Politecnico di Torino

Machine Learning for Networking

01DSMBG

Machine Learning for Domain Name Classification in Network Traffic Analysis

Authors:

Andrea Cuzzi 317868
Luca Nepote 315234

Introduction

In this project, valid for the course "Machine Learning for Networking", we had to complete two objectives: the first one is to develop a supervised Machine Learning method in a Python environment able to classify the flows in order to predict the domain names visited by the clients, according to the statistics obtained by the network analysis. In order to do that we had to investigate different classifiers and to determine the minimum number of features (and data) before dropping of the performances. Moreover, for the second goal, we had to develop an unsupervised Machine Learning algorithm able to group domain names, looking for "families" of domain names and finding which are the most important features for clustering.
For this purpose, we used data collected by the Tstat tool, which is a passive traffic monitor that acquired statistics about flows. In the dataset, we had 125 features, of which 122 are numerical.
Finally, we are provided with two different datasets: "Project2_training.csv", which we used for taking all the decisions about the procedure and modeling, and "Project2_validation.csv", which has been used only at the end for testing the model.

Task 1: Data Characterization and Features Engineering

At first, we explored the dataset and learned about the behavior of features at different levels, such as the client IPs, the domain names, and the client and server ports, performing both data visualization and statistical analysis. The different numbers of quantities have been reported in the table below.

Client IP	Domain Names	Client Ports	Server Port
738	26	30129	[443]

Notice that we have 26 labels and only one server port, whose value is 443. Analyzing the dataset's distribution, we noted that the classes are not balanced: some have a bigger number of samples and others less. Four main classes contain the most number of samples, specifically, they are *other*, *scdn.co*, *taboola.com* and *krxd.net*, with each value bigger than 10000 samples. This imbalance has been taken into account in the supervised classification task, considering the appropriate metric.

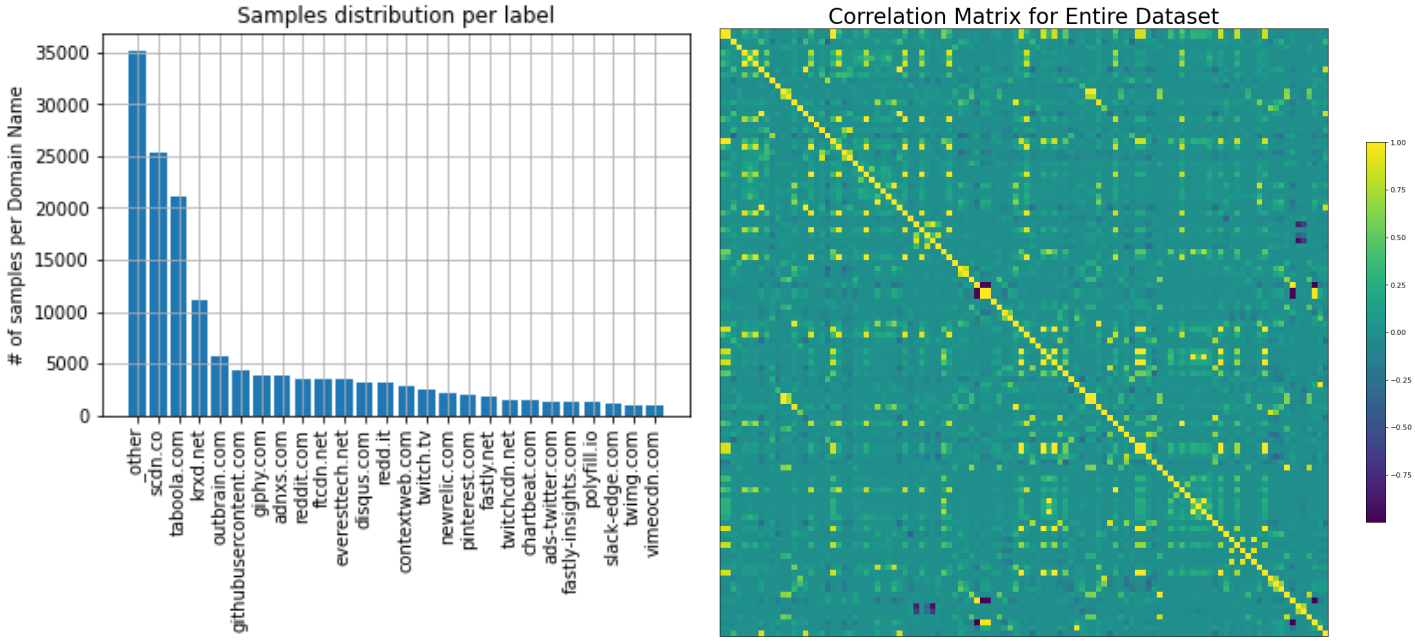


Figure 1: Samples Distribution and Correlation Matrix between the features

Taking into account the number of samples for the majority class, we considered the features for which the value with the greatest occurrence is twice the number of samples in that class. It suggests that those features may not be informative or discriminative enough to differentiate between classes. In such cases, it implies that the occurrence of those features is not unique to a specific class but is shared across multiple classes. While it suggests that those features may not be reliable for classification, it doesn't necessarily mean they are completely useless. Moreover, looking at the values distribution of these features, we can conclude that they do not carry more information or contribute in combination the ones with the others to the overall classification task: indeed, the majority part of the values are equal to 0. In this way, we got rid of all zero columns, which are not useful for the model.

Moreover, looking at the correlation matrix (Figure 1 without labels, and Figure 11), it is clear that several features in the dataset have a high correlation with others (the yellow squares): in that case, we can take into account only one of these features because the others do not carry more useful information with respect to the first one. To do so and to find which the most promising features might be, we eliminated the columns with at least 90% of correlation. If there are identical columns these would be considered only once. As a result of these considerations, we obtained the new correlation matrix (Figure 12), where we can note that the correlation between the features is much smaller with respect to the previous one.

'_c_port', '_s_port' and 'c_ip' '_tls_session_stat' are categorical features. They are not considered in the classification and clustering tasks. Because they do not affect the domain name classification and we avoided the overfitting problem. Indeed, as can be seen in Figure 13 the client ports are not useful for classifying specific domain names, because the behavior is almost the same for all of them. While the server port is always the same, hence it does not affect the classification algorithm. And the IP address of the client port is not useful to distinguish between different domain names.

Starting from the original set of 122 features we end up with 58 useful features reported in Table 3. We have discarded 34+31+3 features reported in Table 4 according to the previous arguments.

Task 2: Supervised classification

The second part of the project consists in using the whole dataset, as specified by the assignment, to build a supervised classifier able to classify the flows according to the domain name. Notice that, in order to compute this task, we divided the dataset into two parts corresponding to 70% of training and 30% of validation. We tested several machine learning models, such as Gaussian Naive Bayes, k-Nearest Neighbors, Decision Tree, and Random Forest, and used heuristic methods to tune their hyperparameters. We evaluated the models using various quality metrics, such as $accuracy$, $recall$, $precision$, and $F1-score$. However, because of the strong imbalance in our classes, we considered the $F1-score$ one, because it combines both the information given by the precision and recall, doing the harmonic mean between the two and providing a balanced measure of the model's performance by considering both false positives and false negatives. We also considered the $macro-avg$ indicator, always because of the imbalance of the classes.

The Gaussian Naive Bayes classifier has very poor performance both for the non-standardized and standardized datasets, as we can see from the pictures reported in the appendix (GNB). This can be due to the fact that the covariance matrices are not diagonalized matrices, so the assumptions valid for the GNB model do not hold anymore. Another possibility is that the classes do not have a Gaussian distribution.

For the k-NN algorithm, we first compute a parameter research looking for the best number of neighbors k according to the 'f1_score macro_avg' metric (notice that we considered the standardized case, for which we expected a better performance). The found value is $k = 3$, and we used it in the algorithm, obtaining good scores both for training and validation.

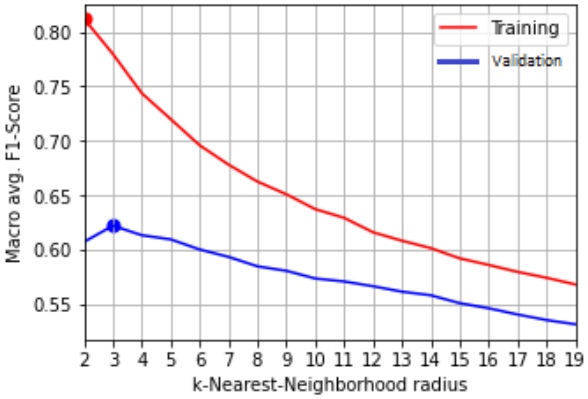


Figure 2: Research of the k parameter

Instead, for the Decision Tree and Random Forest we at first researched the best hyperparameters: for the Decision Tree Classifier they are 'max_depth', 'min_samples_split', and 'min_impurity_decrease'. Meanwhile, the hyperparameters considered for the Random Forest Classifier are the same ones plus 'n_estimators'. We computed these algorithms using the k-fold method, in order to use all the samples, especially for the classes with fewer samples.

However, doing the parameters research, we noted that the results were the same for all the combinations. Moreover, the only class that the algorithm was able to classify is *_other*. so we did not consider Decision Tree and Random Forest for the supervised classification task.

Totally, the resulting accuracy scores for the models, considering only the initial dataset, are the following ones.

	Training			**Validation**		
Classifier	F1 score	Precision	Recall	F1 score	Precision	Recall
GNB	0.08	0.15	0.17	0.08	0.15	0.16
GNB std.	0.11	0.18	0.20	0.10	0.17	0.20
3NN	0.68	0.75	0.65	0.48	0.53	0.46
3NN std.	0.78	0.83	0.75	0.62	0.66	0.60

Table 1: Accuracy metrics for complete dataset

The next step consisted in modifying the dataset trying to increase the performance of the supervised classification: to do that we eliminated the features that we did not consider relevant for the purpose, as described in the previous section, we added two more features, standardized the dataset and performed dimensionality reduction with LDA.
The two new features are:

- **'_amplitude_c_cwin'**: it is defined as the difference between '_c_cwin_max' and '_c_cwin_min' and it corresponds to the amplitude of the congestion window;

- **'_data_density'**: defined as the ratio between '_c_pkts_data_avg' and '_c_msgsize_count' and it is the density of transmitted data with respect to the mean dimension of the packets.

Because we have already eliminated some features, and so reduced the dimensionality of the dataset, we did not use PCA as it will reduce the remaining available information, but only LDA. According to this technique, we found also the most important feature per label, reported in the table 'Top Feature'.
We then recomputed the same supervised classification algorithms described above, using the new dataset and we compared the results with the previous ones.
Notice that in this case we also used the LDA as a classification algorithm, that performs as the Gaussian

Tied-Covariance model. However, looking at the figure reported in the appendix (LDA), we can see that it does not separate well the different classes. Moreover, LDA assumes that each label is associated with the same covariance matrix, but looking at the results we can state that this assumption does not hold and the classes have different distributions.

	Training			**Validation**		
Classifier	F1 score	Precision	Recall	F1 score	Precision	Recall
GNB	0.37	0.38	0.42	0.36	0.37	0.41
3NN	0.82	0.85	0.79	0.68	0.70	0.66
LDA	0.38	0.42	0.39	0.37	0.41	0.39

Table 2: Accuracy metrics for reduced dataset

Considering all the results, we can conclude that the best supervised classification algorithm corresponds to 3-NN on the modified dataset, with the confusion matrices represented below.

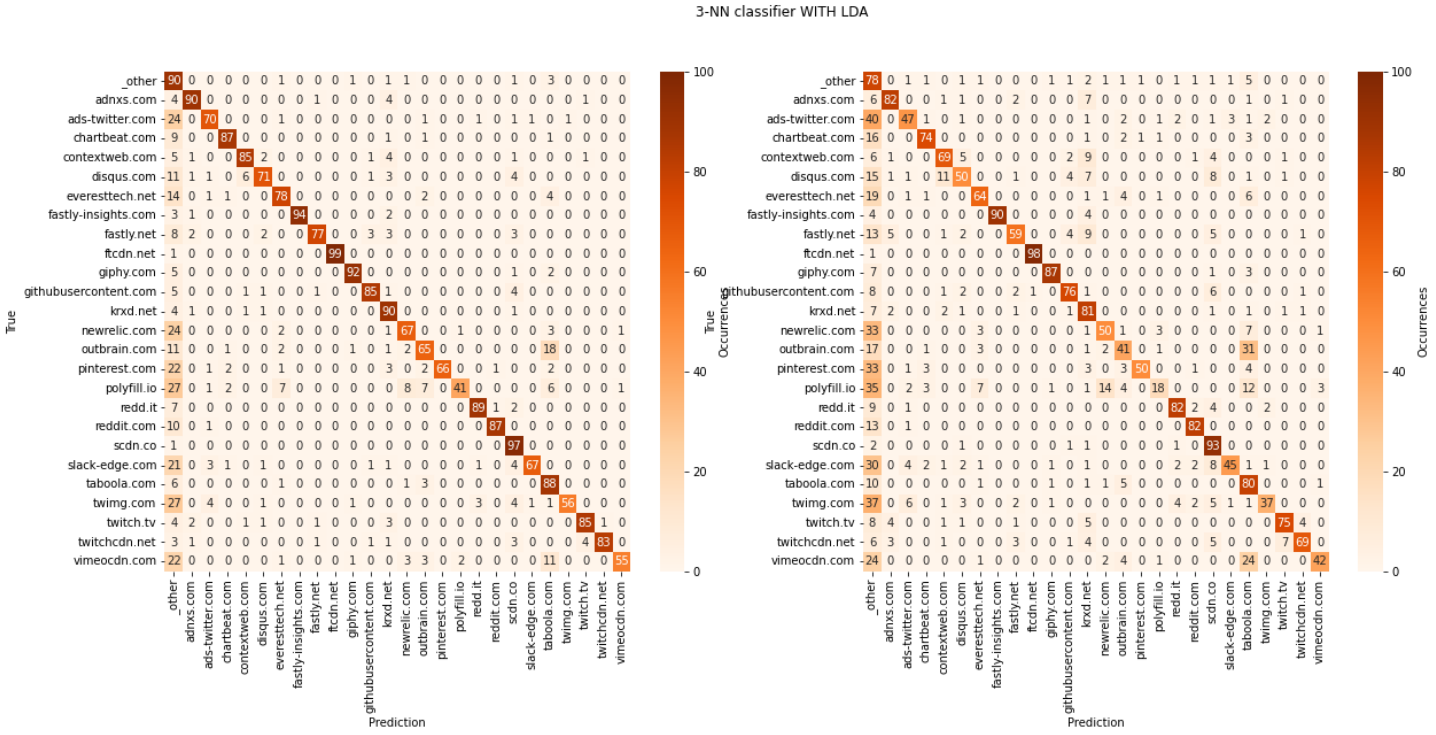


Figure 3: 3NN, reduced dataset

Final evaluation

At the end of task 2, we applied the best algorithm found on the total validation set "Project2_validation.csv". To do that, we first treated both the training and the evaluation dataset by removing the categorical data, the not useful features, and the highly correlated ones. Then we applied LDA. Finally, we fitted the algorithm on the entire training dataset and then we did the prediction on the other one, by considering the best hyper-parameter $k = 3$ find before through the grid search.

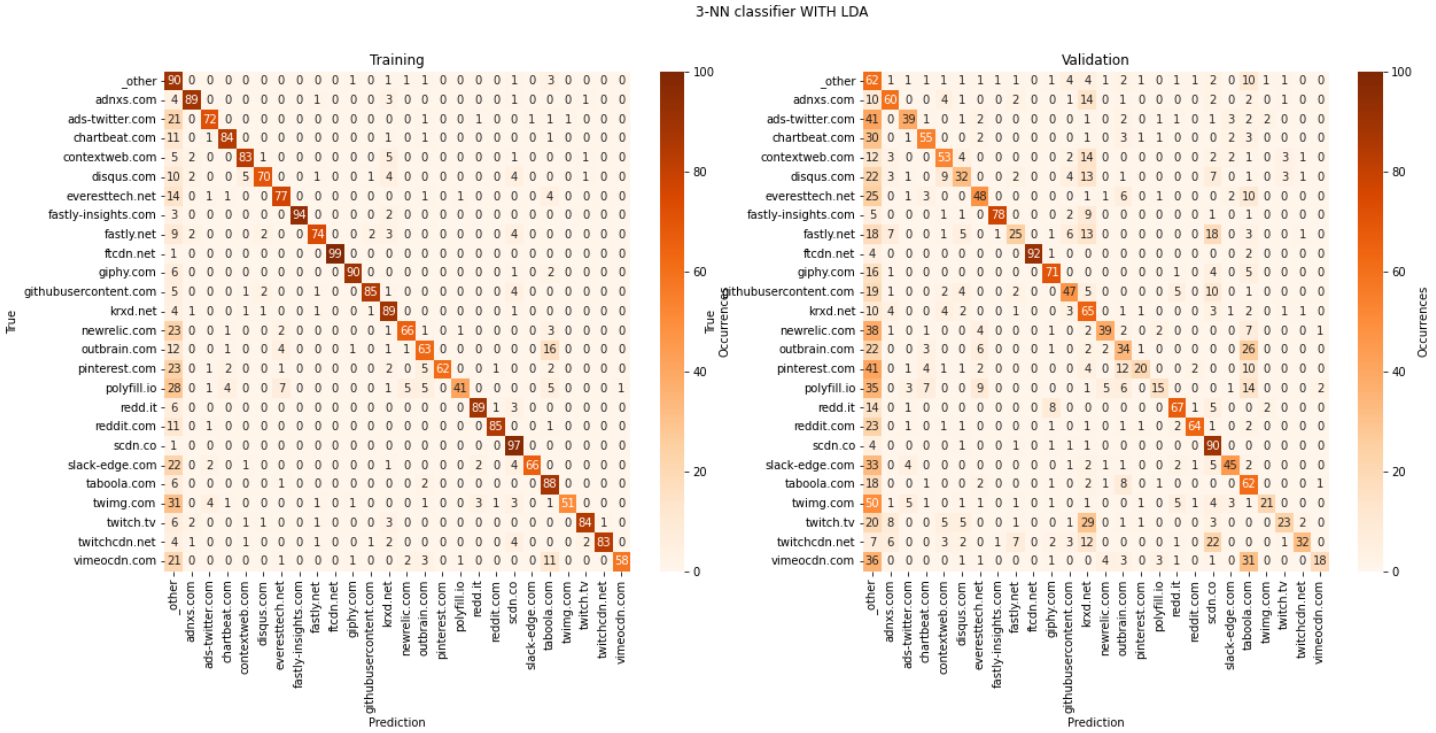


Figure 4: Final Validation

Here we report the scores obtained

	Training			Validation		
Classifier	precision	recall	f1-score	precision	recall	f1-score
3NN	0.85	0.78	0.81	0.48	0.48	0.48

Table 3: Accuracy Metrics for Total Training and Validation

From the results we can note that the performances on the training set are quite different from the ones obtained for the validation: we passed indeed from a $f1\_score = 0.81$ to a value equal to $f1\_score = 0.48$. We can conclude that the algorithm overfits the training set, classifying with a worse performance the samples belonging to the validation set.
This result is well explained in the next task: indeed we will understand which is the best number of features to avoid overfitting (Feature Elimination).
For this task, by using LDA, the number of considered features is equal to the number of labels minus 1, namely 25.

Task 3: Data and Feature Reduction

In this task, we investigated which are the most important features with a Recursive Feature Elimination algorithm, in order to use them to classify flows based on the labels. We also determined the minimum amount of training data before the performance drops: also in this case we used a Recursive Data Reduction algorithm, that step by step eliminates the same percentage of data for each label, maintaining almost the same data distribution between the labels.
Considering the features, starting from the original dataset we run the recursive algorithm, that firstly sorts them for importance, and then eliminates step by step the least important one. Notice that the order in which the features have been sorted is based on the $f1-score$ metric with the $average =' macro'$ indicator: in this way, the algorithm takes into account the imbalance of the different labels, as we already

commented in the precious tasks. As a comparison, we repeat the same evaluation by considering the accuracy metrics: the results for this case have been reported in the appendix.

Initially, we tried to maximize the accuracy of the results on the validation set by using different combinations of hyperparameters.

Feature Elimination

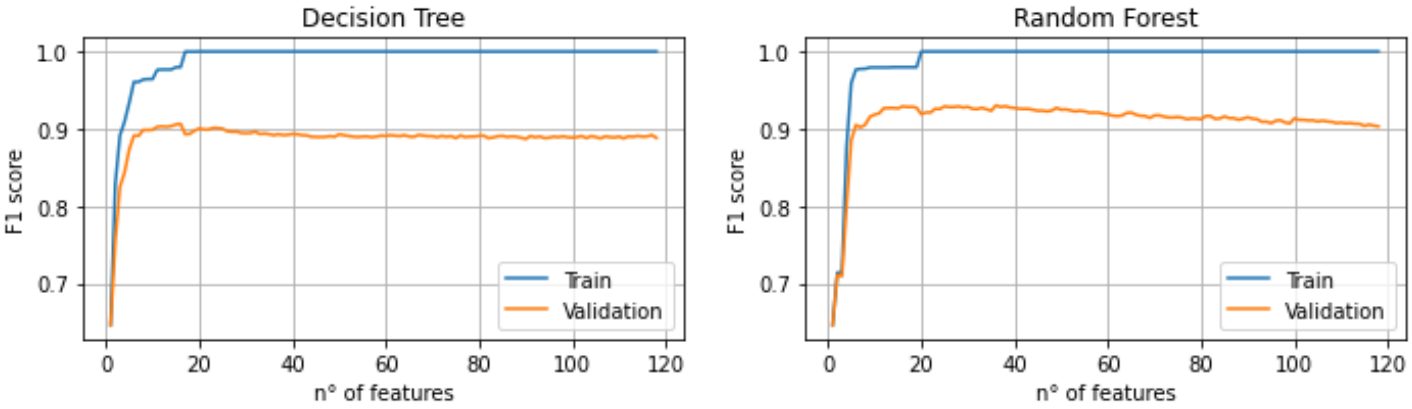
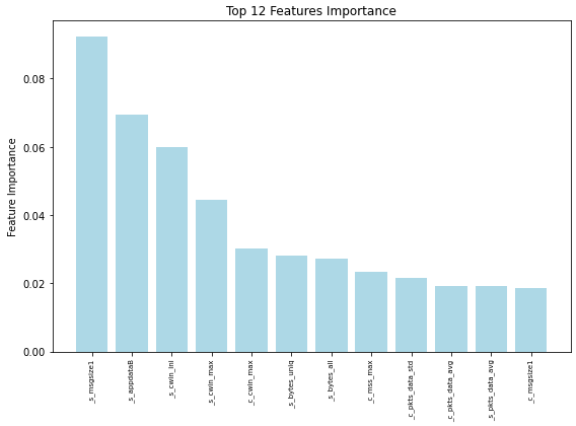


Figure 5: Feature Elimination with Decision Tree and Random Forest

As seen from the charts, the accuracy of the validation set achieved with the Random Forest Classifier is slightly better compared to that obtained with the Decision Tree Classifier. In fact, in the case of the Decision Tree, the maximum accuracy reached on the validation set is around 90%, while the one obtained with the Random Forest Classifier is approximately 93%.
Moreover, we can note that at a certain point (i.e. around $N = 18 - 20$ features), the performance on the validation does not increase: when it happens, using, more features can cause overfitting over the training set. From the graph, it is clearly shown in this case that the training value for $f1 - score$ reaches the value 1, whereas the validation value remains constant at ≈ 0.90.
In order to avoid overfitting we have to take a value less than the limit discussed before, but for which the performances are still good: a possible number is $N = 12$. We reported in the following the most important 12 features, with the corresponding computed importance.



feature	value [%]
_s_msgsize1	9.23
_s_appdataB	6.93
_s_cwin_ini	6.01
_s_cwin_max	4.44
_c_cwin_max	3.03
_s_bytes_uniq	2.82
_s_bytes_all	2.73
_c_mss_max	2.33
_c_pkts_data_std	2.16
_c_pkts_data_avg	1.91
_s_pkts_data_avg	1.91
_c_msgsize1	1.86

Figure 6: Comparison of the Top 12 Features with Random Forest classifier and f1-score metric

Among the features that best generalize the Random Classifier, there are some that were discarded in the first task due to high correlation with other features. These features are '_s_bytes_uniq', '_s_bytes_all', and '_c_msgsize1'.

While the features that are discarded based on the reasoning of maximum occurrence are not among the Top 12.

Data Reduction

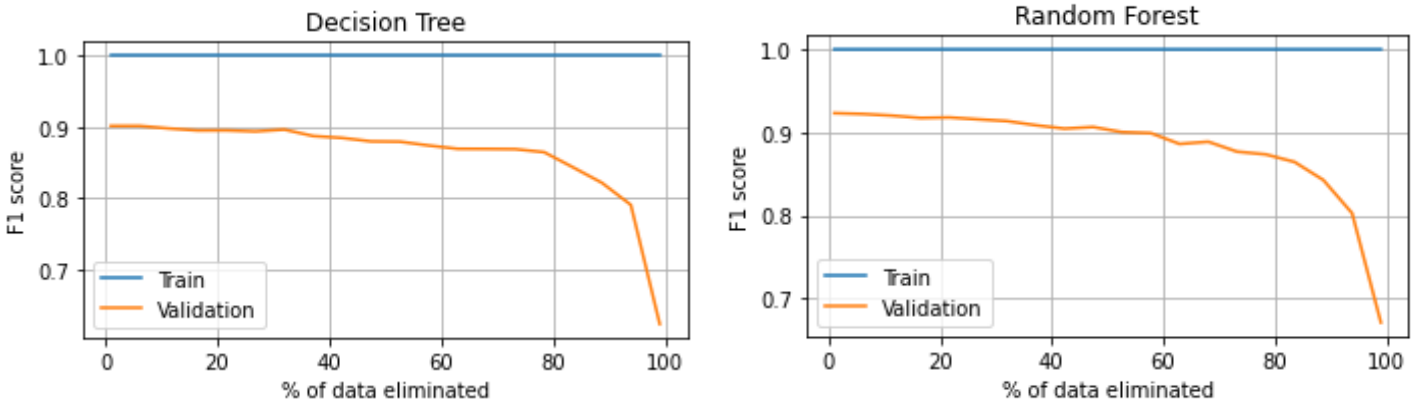


Figure 7: Data Reduction with Decision Tree and Random Forest - f1 score

The graphs shown above validate the classifier for a step size of 5 percent.

It can be observed that as the percentage of samples decreases, in the same way for each label, the accuracy obtained for the training set does not change. In fact, this is also confirmed by the graphs obtained during Features Reduction: if there are many features that distinguish the samples, then the classifier overfits and fails to generalize the results for the validation dataset.

Referring to the accuracy of the validation set, it can be noticed that as the number of samples used during training decreases, the classifier's performance drops out.

Task 4: Unsupervised Clustering

In this task, we grouped flows into clusters according to the considered features, in order to identify homogeneous "families" of domain names.
We first applied PCA Dimensionality Reduction to take advantage of the different variances among the classes, to better distinguish them. We also considered a lower number of features (10) with respect to considered 12 Initial Features to reduce the noise due to possible outliers.
We tested several clustering algorithms, such as k-Means, Hierarchical Clustering, and DBSCAN, and evaluated their best hyperparameters according to the silhouette score. Indeed, this is an unsupervised metric as if we don't know in advance the sample labels.

For what regards **K-Means**, we chose the value of k corresponding to the highest value of the silhouette: in this case, it corresponds to 25 clusters (Silhouette Score). On the other hand, we did not consider the Elbow Method because we are not able to identify a well-defined elbow in the curve; indeed, this method works well if the clusters are well separated and they have a spherical shape, differently from our case. Moreover, We can note that the number of clusters corresponds approximately to the number of labels.

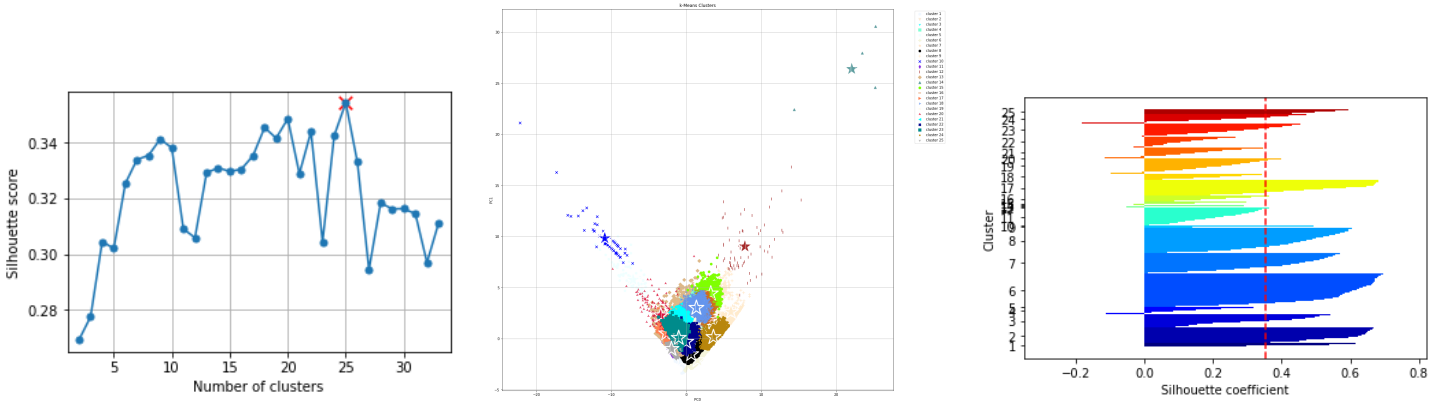


Figure 8: Silhouette heuristic approach - Training Samples

We can test the accuracy of the obtained result by looking at the Silhouette per sample graph reported above: we can see how the silhouette values have very different lengths, in fact comparing them with the mean value (red dotted line) we can note that some classes have all the samples below the mean value and other classes have all the samples well above the mean value. In this case, the clustering is not optimal, namely, the classes are not well separated from each other.
Furthermore, it should be noted that there are samples for which the silhouette takes on a negative value: these samples are on average closer to the neighboring groups than to one's own: it is therefore poorly classified.

Moreover, we expected that **DBSCAN** might be the one with the poorest performance because we are dealing with a dataset with different clusters' densities: indeed, as we studied during the course, DBSCAN could have issues when the classes have different density distributions, and when the dataset has a large dimensionality. In our case, the data points are not uniformly distributed, and there are regions of varying densities, DBSCAN may not identify the clusters correctly; indeed, our samples are strongly overlapped. If there are a lot of noise points (outliers) they may be considered as separate clusters, and this is not good.
The best hyper-parameters according to the maximum of the Silhouette Score are $metric$ = 'cosine', $eps = 0.15$, and $mins = 10$.

As can be seen from graphs reported in the appendix, it is clearly visible that the grid search for the best hyperparameters through the Silhouette Score doesn't provide good results. Indeed, looking at the supervised metric ARI, the best hyper-parameters are much different than the one obtained. And finally, this result does not separate the samples, they are still grouped together (Confusion Matrix).
We get a better performance with k-Means because it suits well larger dataset and it is more robust against outliers.
Notice that, in the picture, the stars represent the centroids of the clustering algorithm.

To evaluate the performance of the **Hierarchical Agglomerative clustering**, differently from the other two clustering algorithms, we have used fewer samples because of the complexity of the model, so the final result may not perform well. In order to do so, we used a technique of subsampling of the majority classes by choosing a number of samples equal to those in the least numerous class. In this way, we considered a balanced dataset for the analysis.
Then through a grid search, we chose the best hyper-parameters according to the maximum Silhouette score. We obtained $affinity$ = 'euclidean', $linkage$ = 'single', $n\_cluster = 10$. The obtained result is not good, as can be seen from the obtained clustering.

We Finally evaluate the clustering algorithm with the corresponding best hyper-parameters, obtained through the Silhouette score, on the validation dataset.

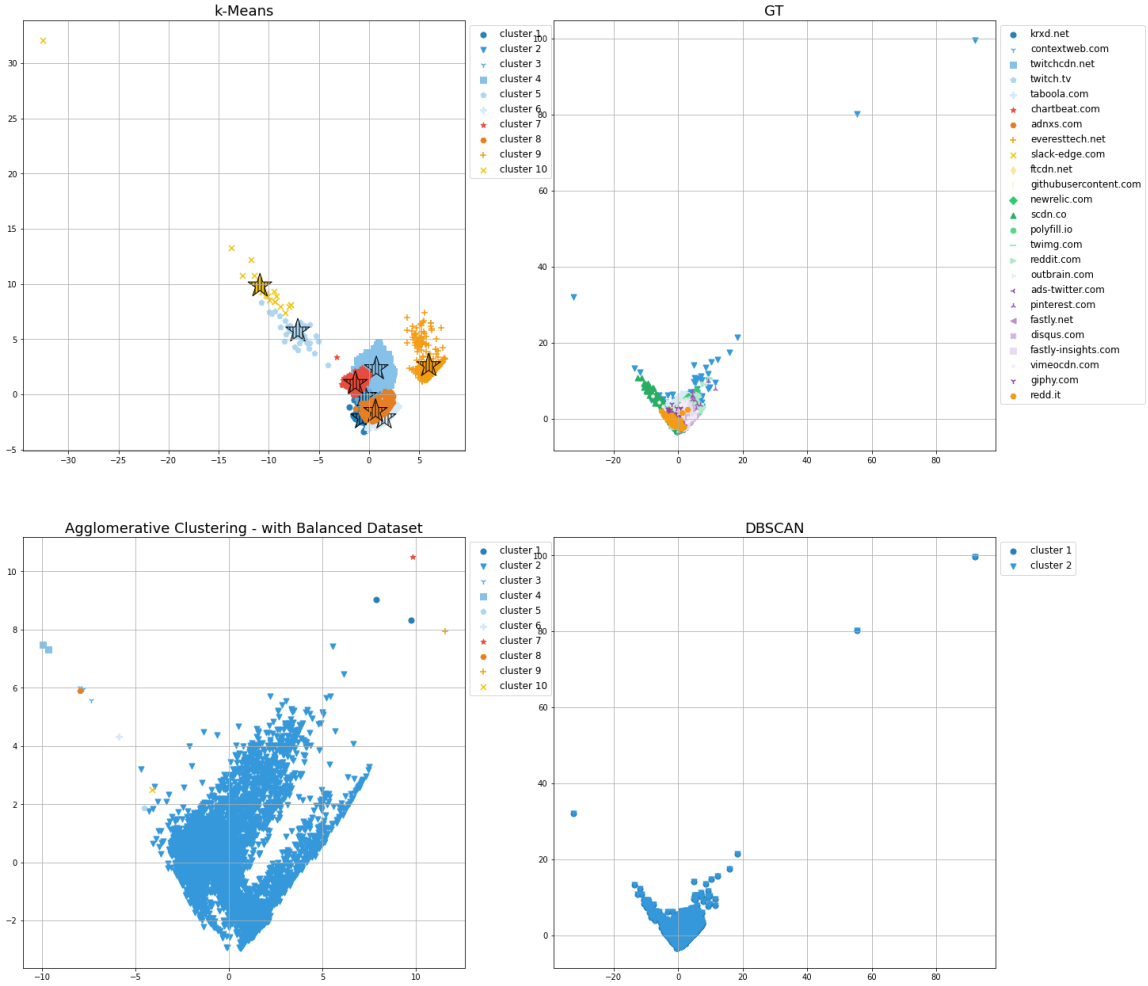


Figure 9: Comparison between clustering methods

Can be clearly seen that the best result is obtained with the k-Means algorithm, also if it is not accurate, looking at the values obtained for the supervised metrics.

	Silhouette	Homogeneity	Purity	ARI
K-Means	0.36	0.21	0.34	5.05e-02
DBSCAN	0.64	1.21e-3	3.96e-2	2.79e-07
Agglomerative	0.09	4.1e-5	0.24	-1.43e-05

Table 4: Clustering results on the validation dataset

Final evaluation

The best clustering method to use is the k-Means algorithm according to the Supervised Metric. Whereas the Unsupervised metric suggests using DBSCAN, according to the highest Silhouette Score.

We decided to test the validation dataset by using k_Means, although the silhouette is lower than in DBSCAN. In this way, the chosen algorithm is better able to group samples belonging to the same class.

In order to obtain the final result we considered the validation dataset and perform the same operation done to the training one. So we have eliminated the categorical features and considered the best 12 Features. Then we Transform the data by using the scaler and PCA method trained on the training dataset.

And Finally, we used the k-Means algorithm fitted in the training set with the best hyper-parameters obtained before.

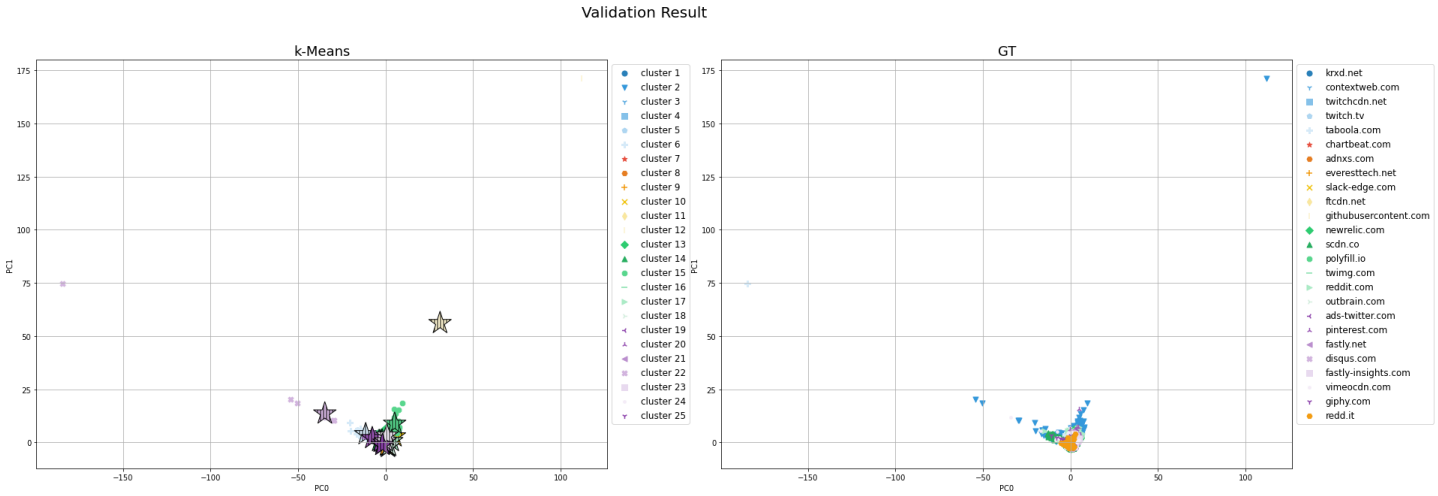


Figure 10: Final Validation

Here we report the score obtained

	Silhouette	Homogeneity	Purity	ARI
K-Means	0.307101	0.167348	0.345566	3.476019e-02

Table 5: Clustering results on the validation dataset

The final evaluation is not much different from the one obtained considering the training dataset. Therefore the validation Dataset is consistent with the training dataset.

Conclusions

Concluding, the obtained results are poor: indeed the best supervised classification algorithm suffers of overfitting, and so it is not able to classify correctly unseen samples. A better solution could be to use the features found in task 3 to avoid overfitting.
In the classification task, we used LDA which finds 25 discriminant directions. However, the maximum number of features to avoid overfitting (it has been found in the third task) is 15. So we can conclude that according to the overfitting obtained in the second task by using the discriminant features, the different classes are strongly overlapped.

For what concern the clustering method we obtained not very good results, as can be seen by both the supervised and unsupervised Scores. Moreover, the best clustering algorithm, according to the supervised metrics, is not able to separate very well the samples, because they are strongly overlapped. A possible solution can be to use DBSCAN by considering the best hyperparameters according to the supervised metric (ARI) instead of the unsupervised one (Silhouette) reported in the appendix.

Appendix A: Figures and Tables

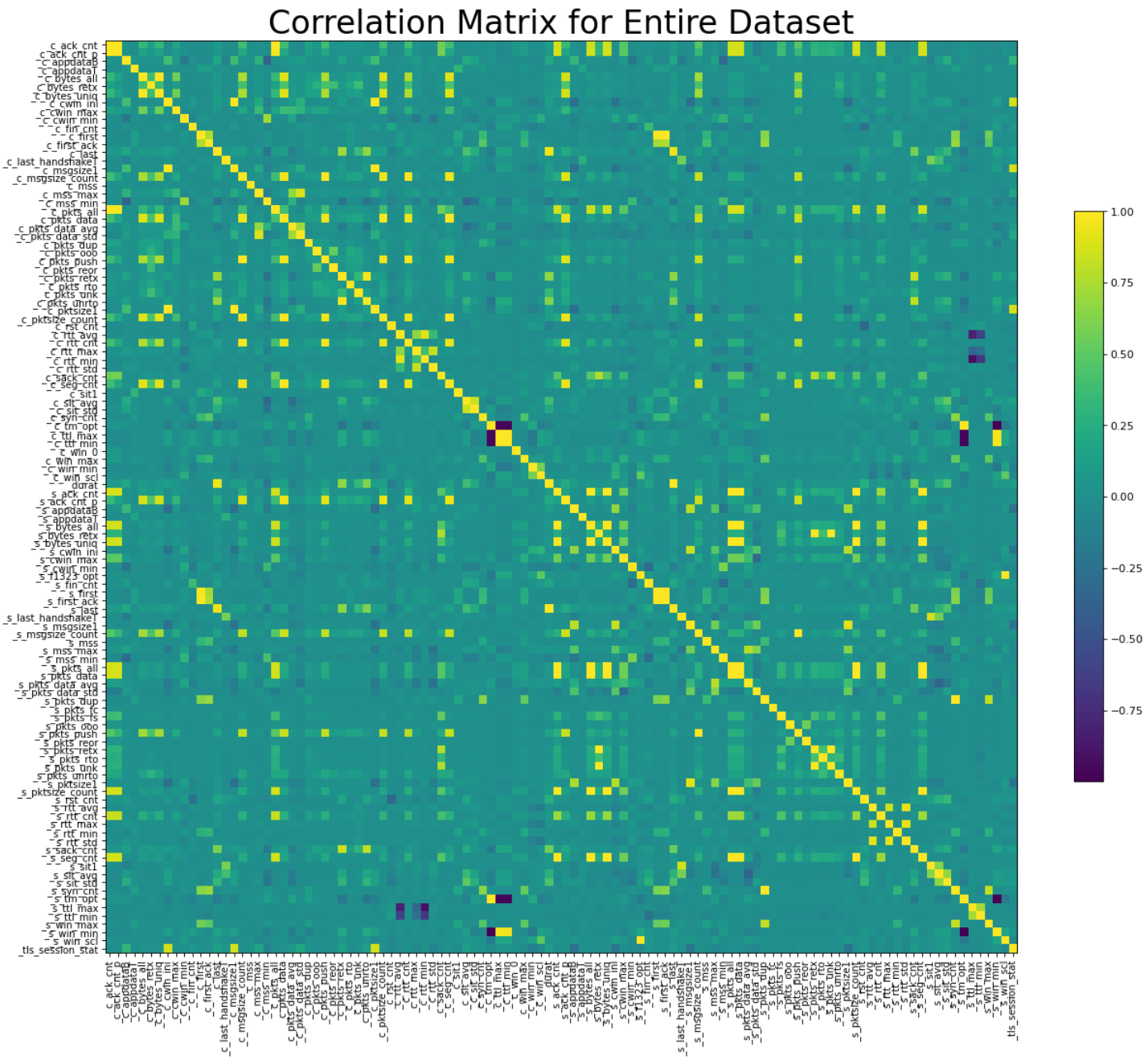


Figure 11: Correlation Matrix Complete

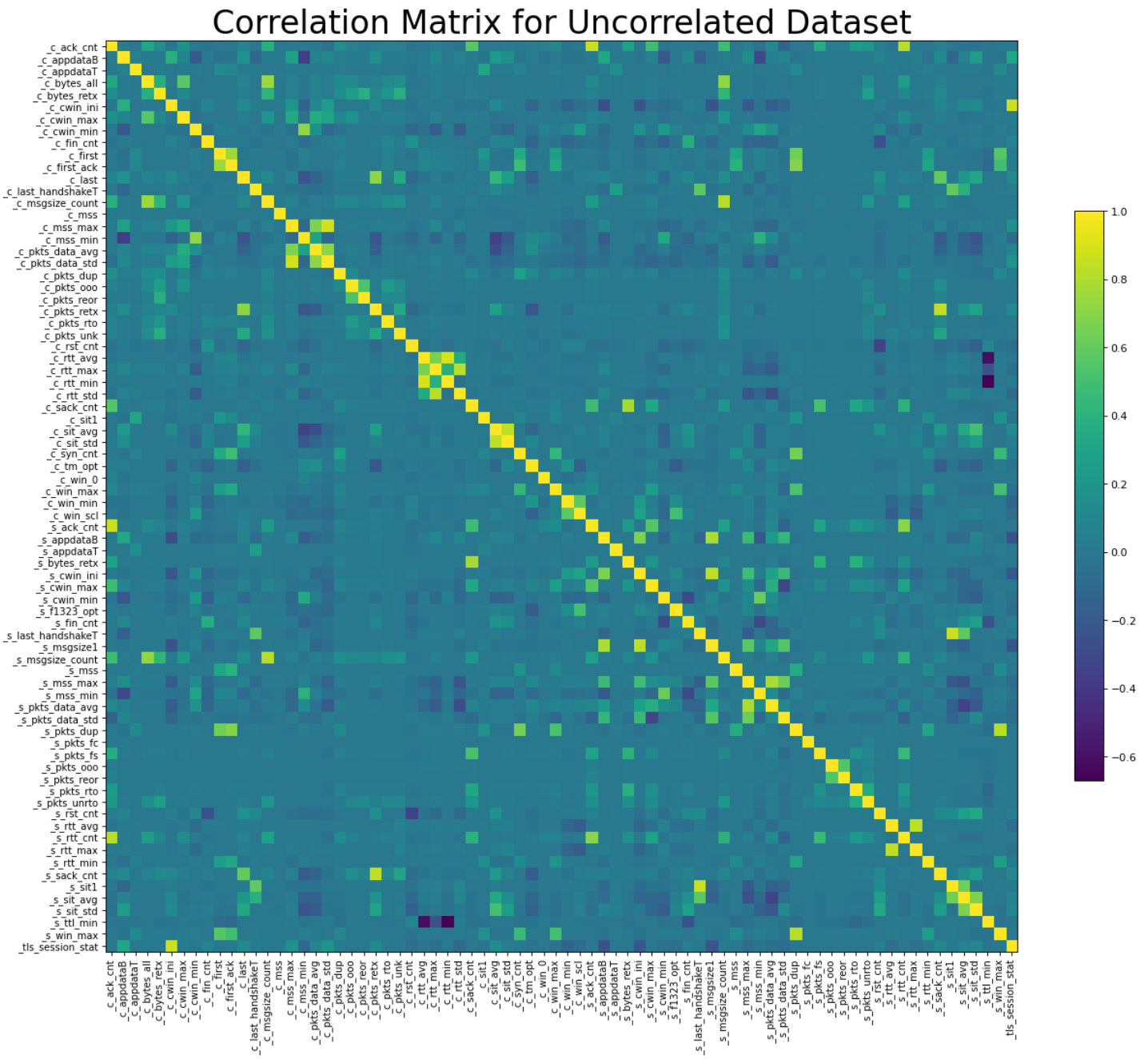


Figure 12: Correlation Matrix for Uncorrelated Dataset

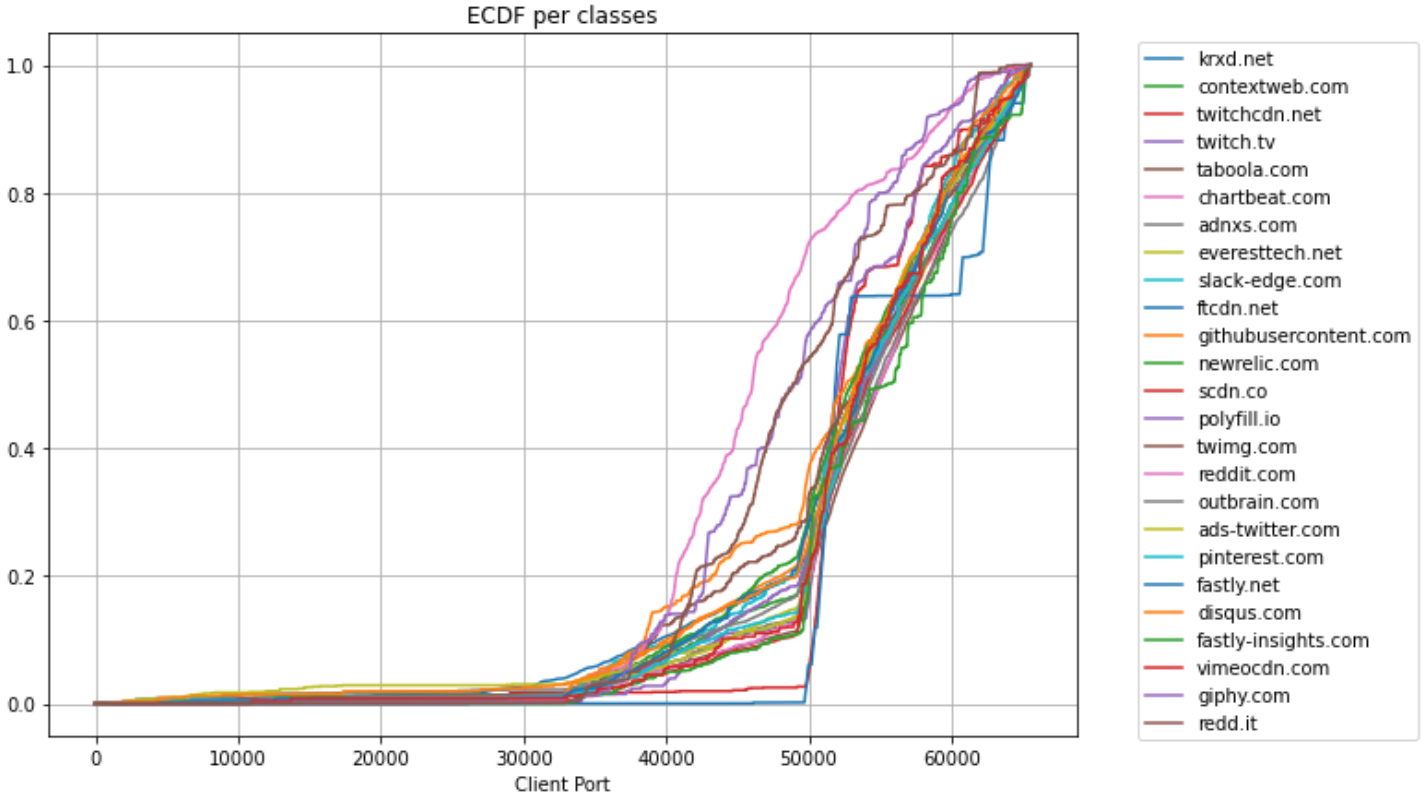


Figure 13: Client Port ECDF

Table 6

Useful Features				
_c_ack_cnt	_c_appdataB	_c_appdataT	_c_bytes_all	_c_cwin_ini
_c_cwin_max	_c_cwin_min	_c_fin_cnt	_c_first	_c_first_ack
_c_last	_c_last_handshakeT	_c_msgsize_count	_c_mss	_c_mss_max
_c_mss_min	_c_pkts_data_avg	_c_pkts_data_std	_s_win_max	_c_rst_cnt
_c_rtt_avg	_c_rtt_max	_c_rtt_min	_c_rtt_std	_c_sack_opt
_c_sit1	_c_sit_avg	_c_sit_std	_c_syn_cnt	_c_win_max
_c_win_min	_c_win_scl	_s_ack_cnt	_s_appdataB	_s_appdataT
_s_cwin_ini	_s_cwin_max	_s_cwin_min	_s_f1323_opt	_s_fin_cnt
_s_last_handshakeT	_s_msgsize1	_s_msgsize_count	_s_mss	_s_mss_max
_s_mss_min	_s_pkts_data_avg	_s_pkts_data_std	_s_ttl_min	_s_rtt_avg
_s_rtt_cnt	_s_rtt_max	_s_rtt_min	_s_sack_opt	_s_sit_avg
_s_sit_std				

Table 7

Not useful features	**high correlated features**	**categorical data**
_c_bytes_retx	_c_ack_cnt_p	_c_port
_c_pkts_dup	_c_bytes_uniq	_s_port
_c_pkts_fc	_c_msgsize1	_c_ip
_c_pkts_ooo	_c_pkts_all	_tls_session_stat
_c_pkts_reor	_c_pkts_data	
_c_pkts_retx	_c_pkts_push	
_c_pkts_rto	_c_pkts_unrto	
_c_pkts_unfs	_c_pktsize1	
_c_pkts_unk	_c_pktsize_count	
_c_pkts_unrto	_c_rtt_cnt	
_c_sack_cnt	_c_seg_cnt	
_c_syn_retx	_c_ttl_max	
_c_tm_opt	_c_ttl_min	
_c_win_0	_durat	
_s_bytes_retx	_s_ack_cnt_p	
_s_pkts_dup	_s_bytes_all	
_s_pkts_fc	_s_bytes_uniq	
_s_pkts_fs	_s_first	
_s_pkts_ooo	_s_first_ack	
_s_pkts_reor	_s_last	
_s_pkts_retx	_s_pkts_all	
_s_pkts_unk	_s_pkts_data	
_s_pkts_push	_s_pkts_push	
_s_pkts_rto	_s_pkts_retx	
_s_pkts_unfs	_s_pkts_unk	
_s_pkts_unk	_s_pktsize1	
_s_pktsize_count	_s_pktsize_count	
_s_rst_cnt	_s_rtt_std	
_s_sack_cnt	_s_seg_cnt	
_s_sit1	_s_syn_cnt	
_s_syn_retx	_s_tm_opt	
_s_tm_opt	_s_ttl_max	
_s_win_min	_s_win_min	
_s_win_0	_s_win_scl	

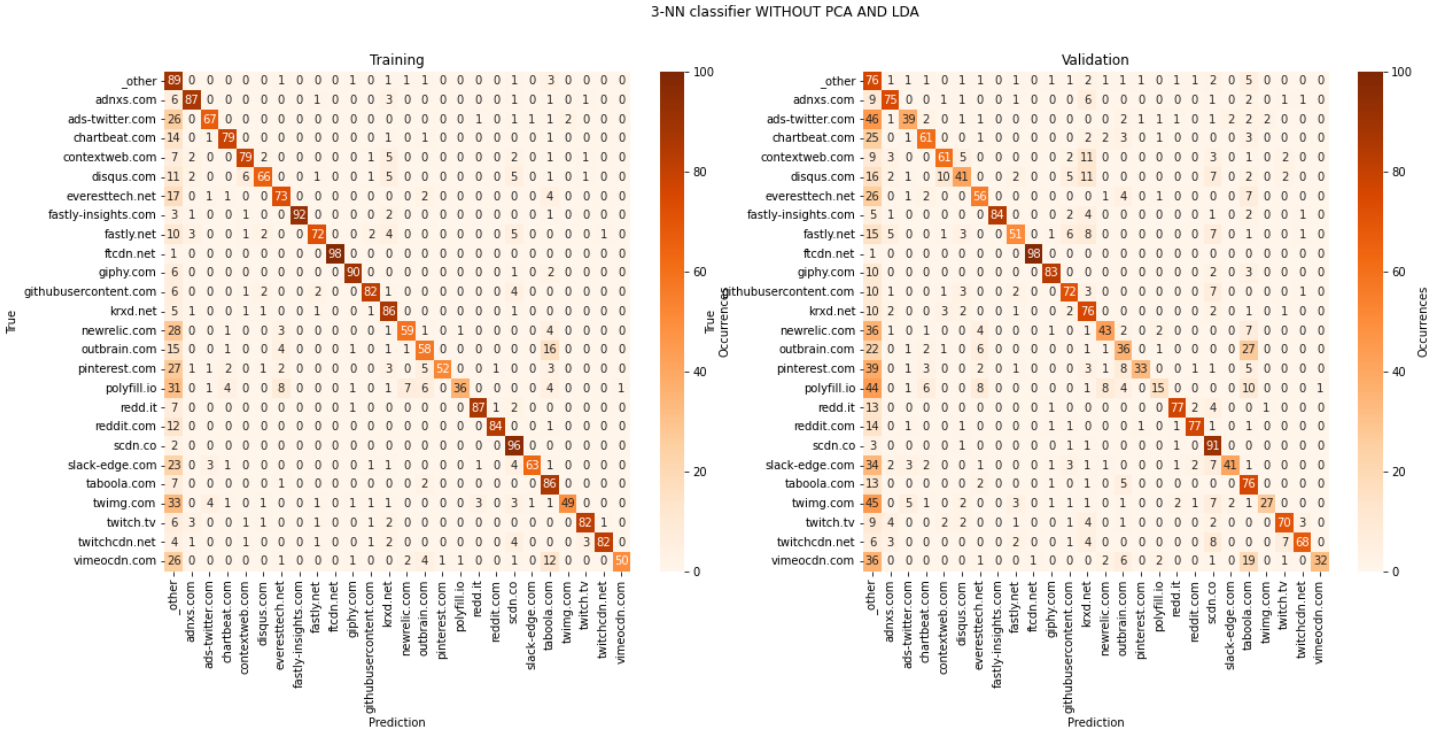


Figure 14: 3NN, standardized dataset

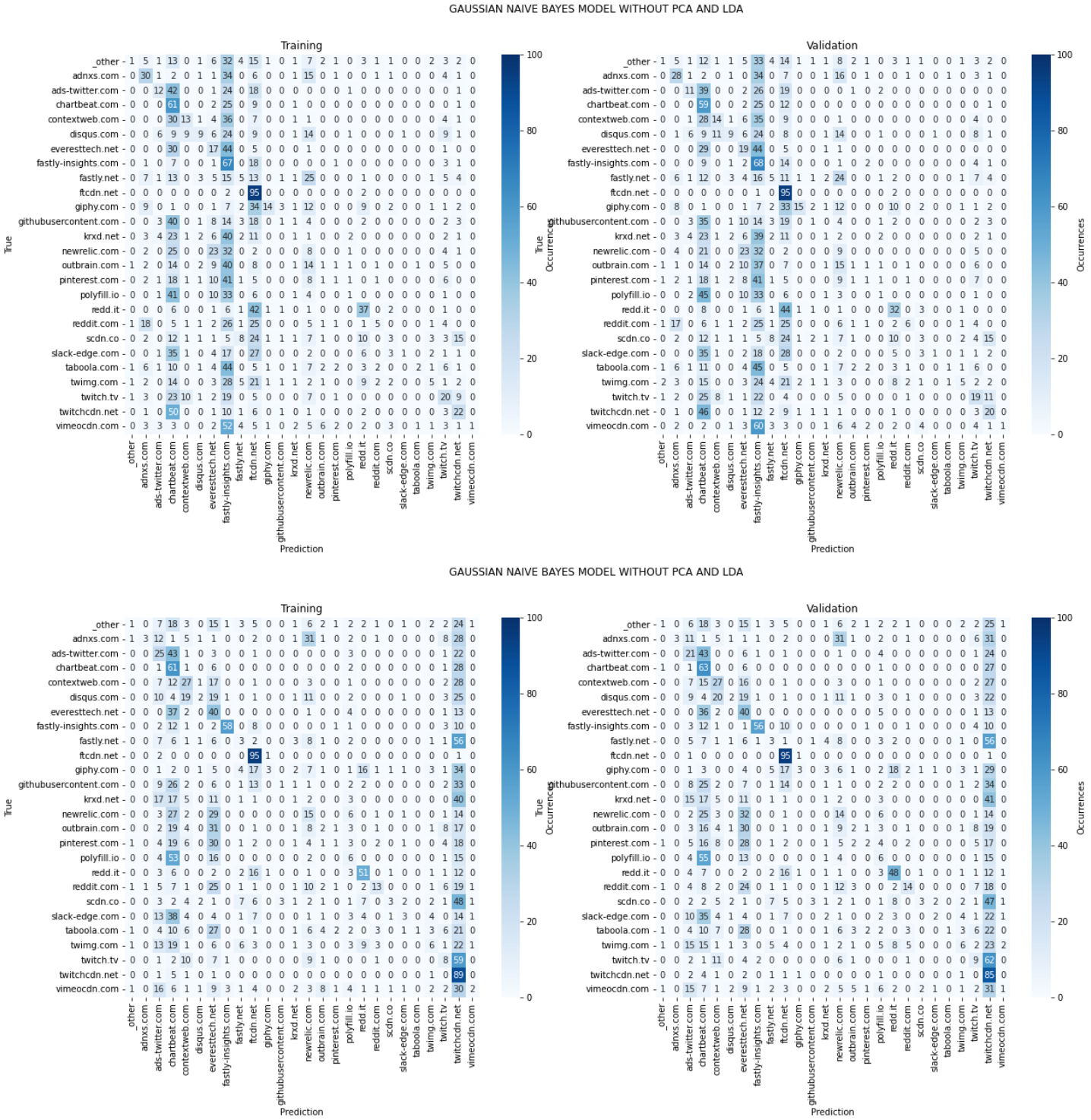


Figure 15: GNB, not standardized and standardized cases

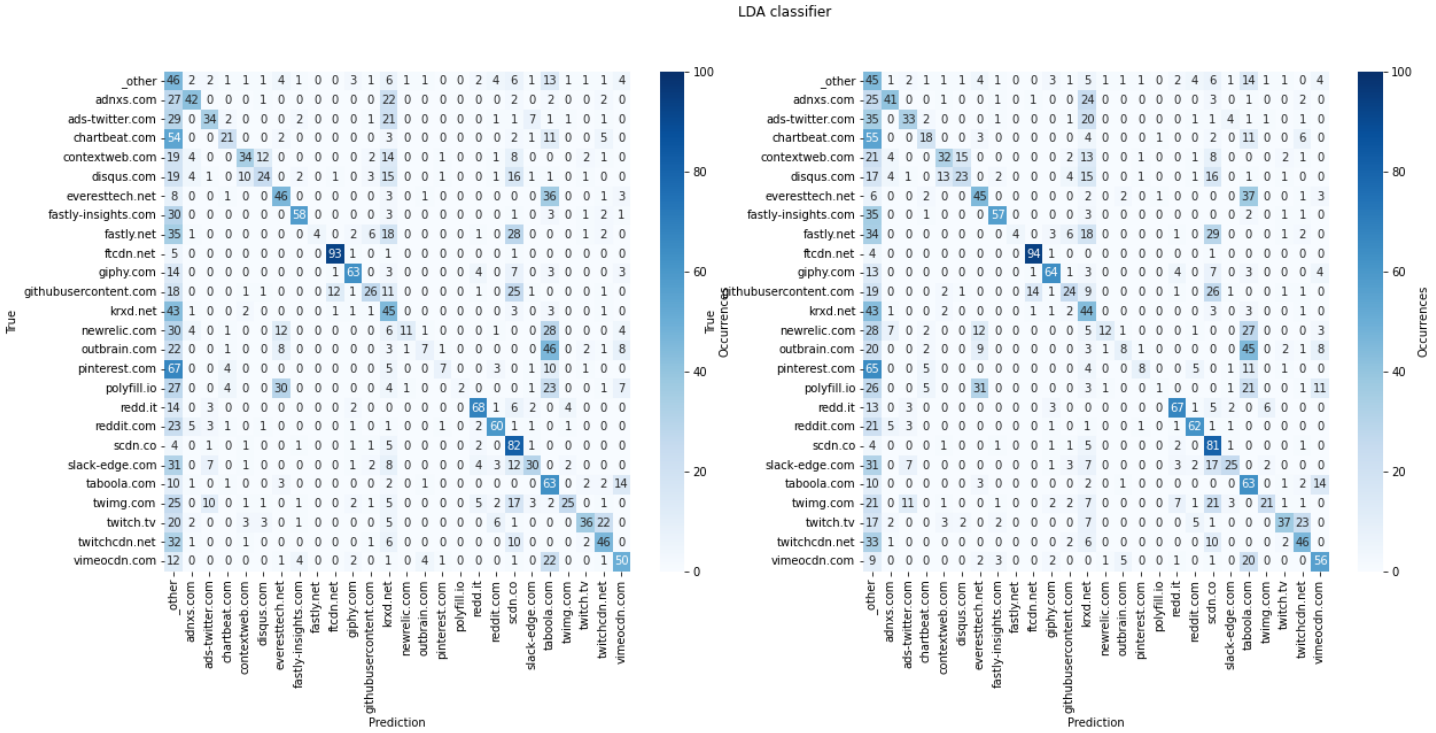


Figure 16: LDA, reduced dataset

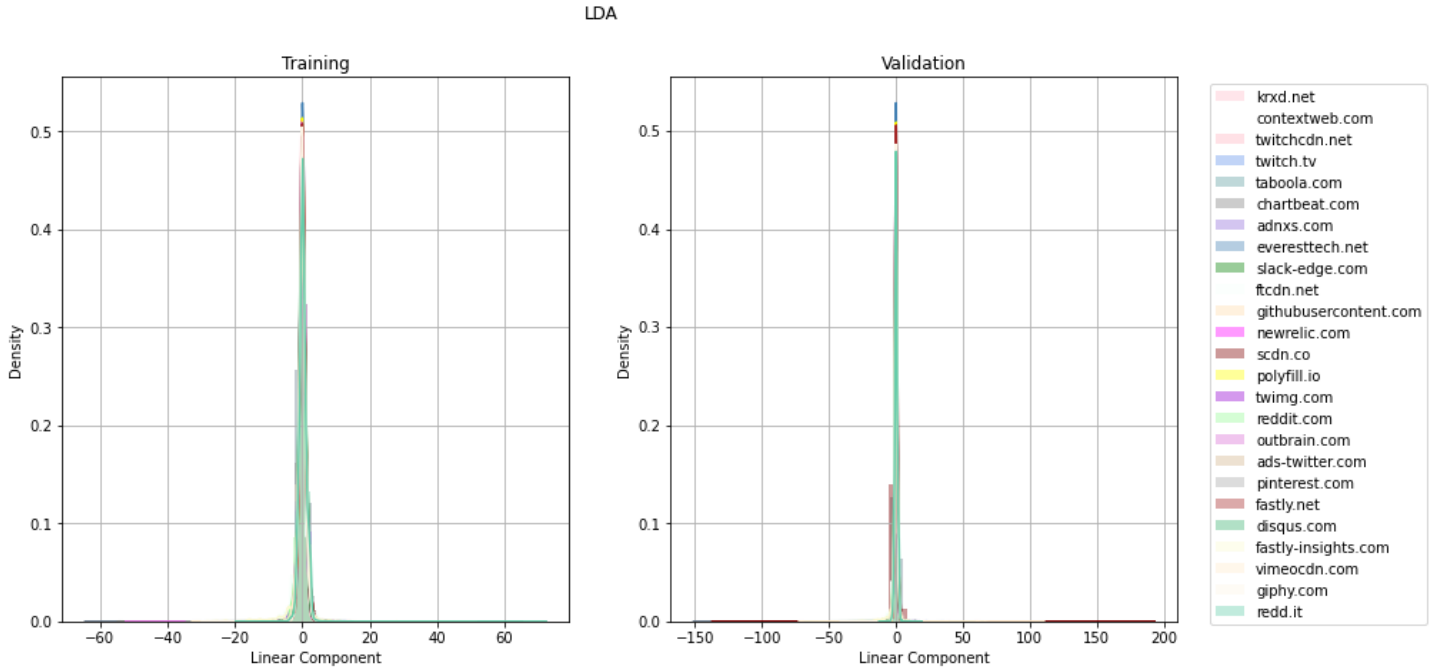


Figure 17: LDA distribution

label	top feature
_other	_c_ack_cnt_p
adnxs.com	_c_ack_cnt_p
ads-twitter.com	_c_ack_cnt_p
chartbeat.com	_c_pkts_all
contextweb.com	_c_seg_cnt
disqus.com	_c_ack_cnt_p
everesttech.net	_c_ack_cnt
fastly-insights.com	_c_ack_cnt
fastly.com	_c_ack_cnt
ftcdn.net	_c_seg_cnt
giphy.com	_c_pkts_all
githubusercontent.com	_c_pkts_all
krxd.net	_c_seg_cnt
newrelic.com	_c_ack_cnt_p
outbrain.com	_c_ack_cnt
pinterest.com	_c_ack_cnt_p
polyfill.io	_c_ack_cnt
redd.it	_c_ack_cnt
reddit.com	_c_ack_cnt_p
scdn.co	_c_pktsize_count
slack-edge.com	_c_ack_cnt_p
taboola.com	_c_pktsize_count
twimg.com	_c_pkts_all
twitch.tv	_c_ack_cnt
twitchcdn.net	_c_ack_cnt
videocdn.com	_c_ack_cnt_p

Table 8: Top feature for label according to LDA

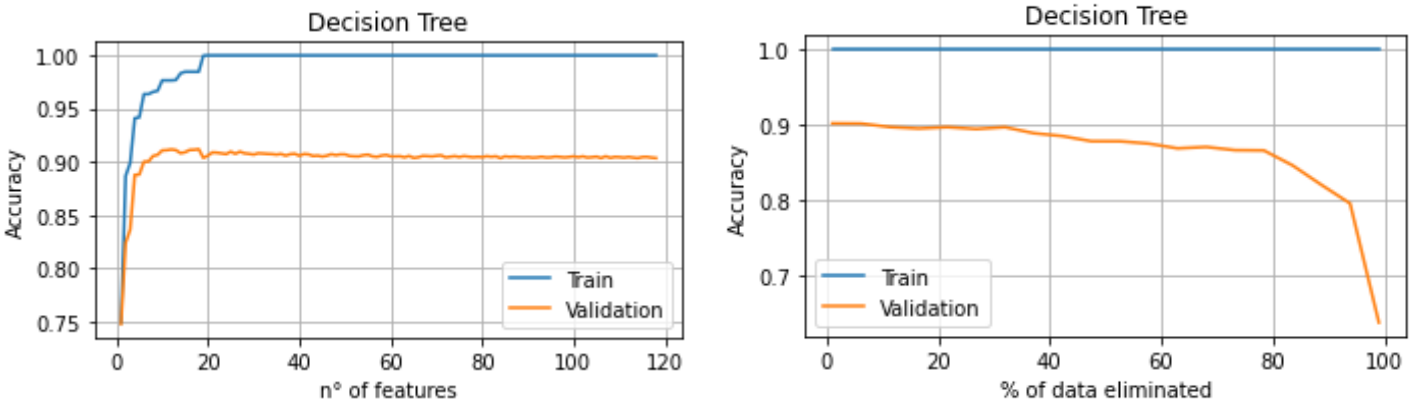


Figure 18: Feature and Data Elimination with Decision Tree and accuracy metric

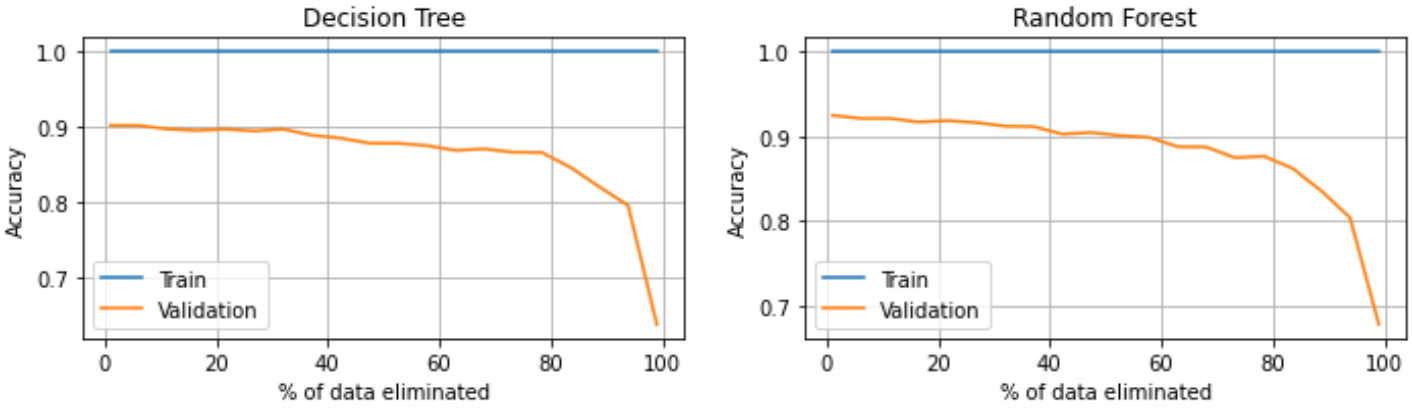


Figure 19: Data Reduction with Decision Tree and Random Forest - Accuracy score

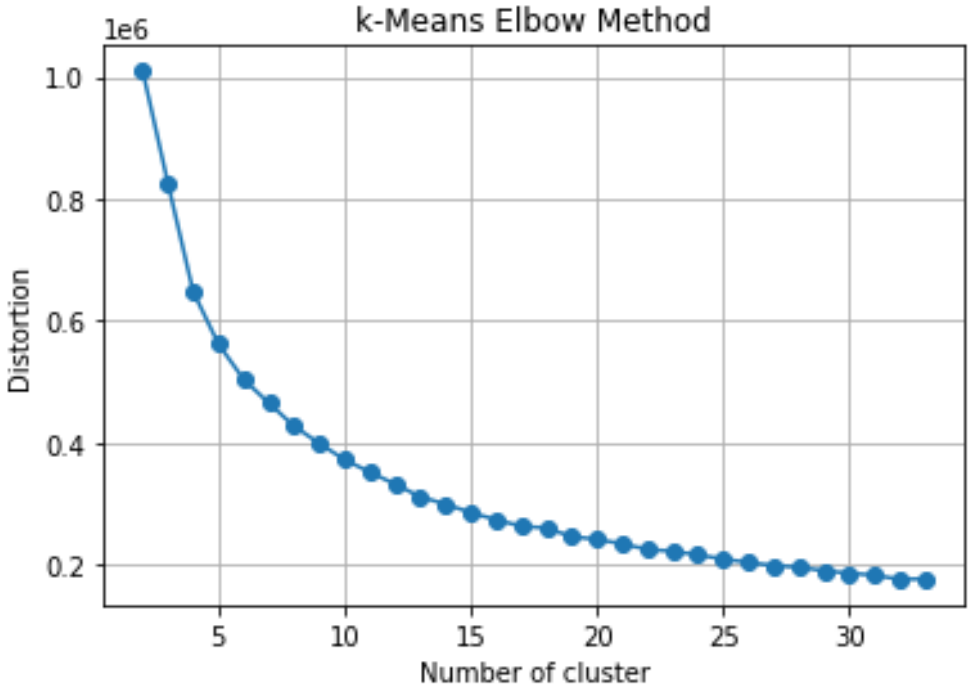


Figure 20: Elbow Method

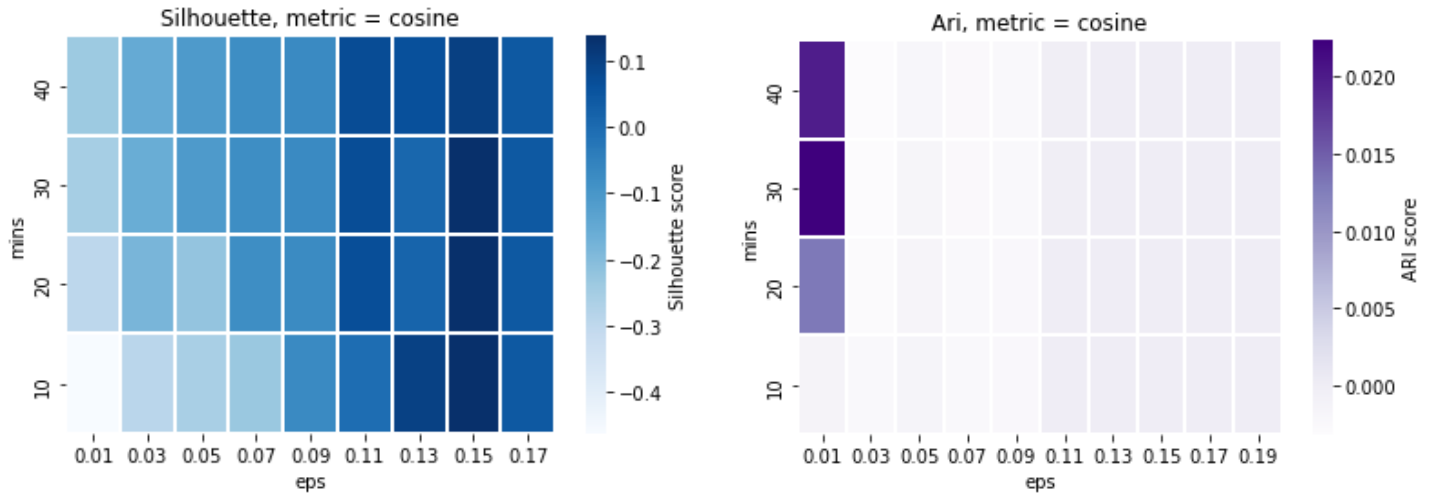


Figure 21: DBSCAN with cosine metric- Training Samples

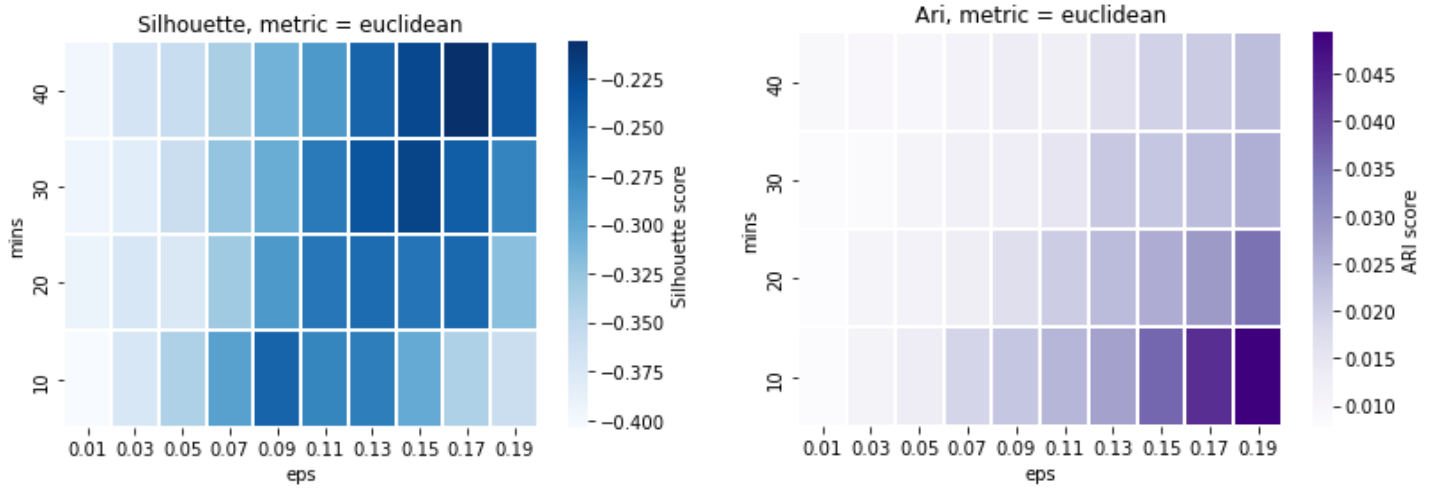


Figure 22: DBSCAN with euclidean metric- Training Samples

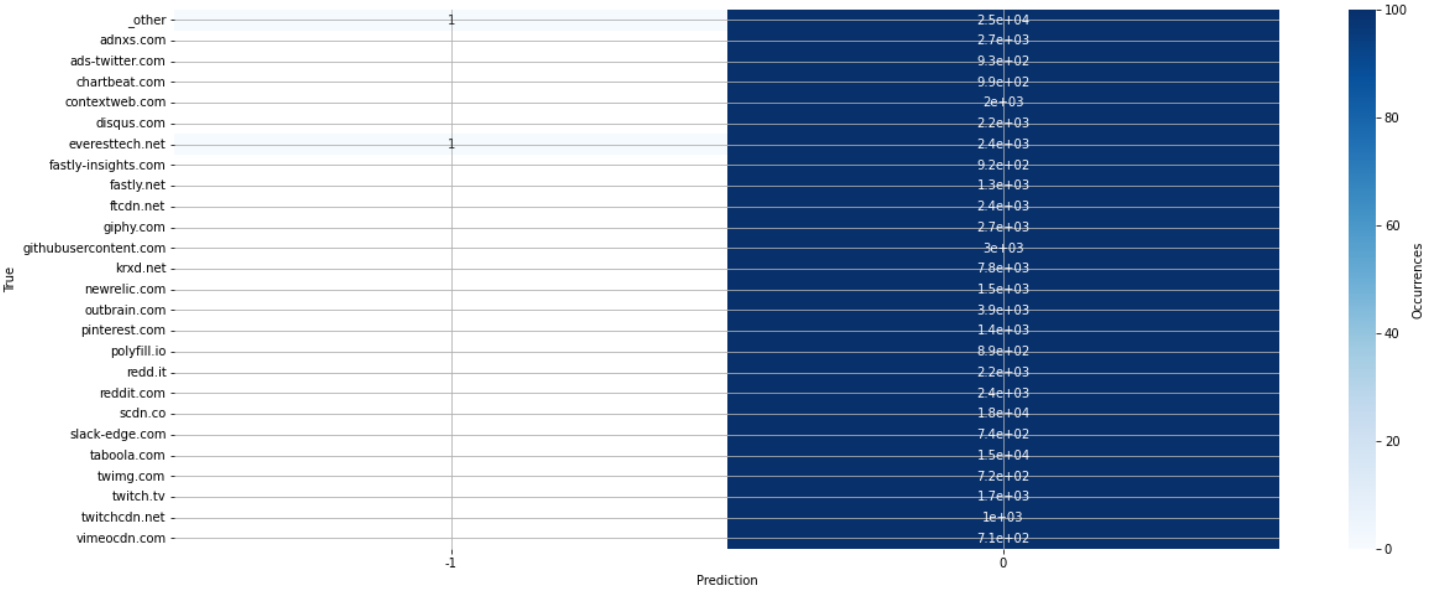


Figure 23: DBSCAN Confusion Matrix- Training Samples

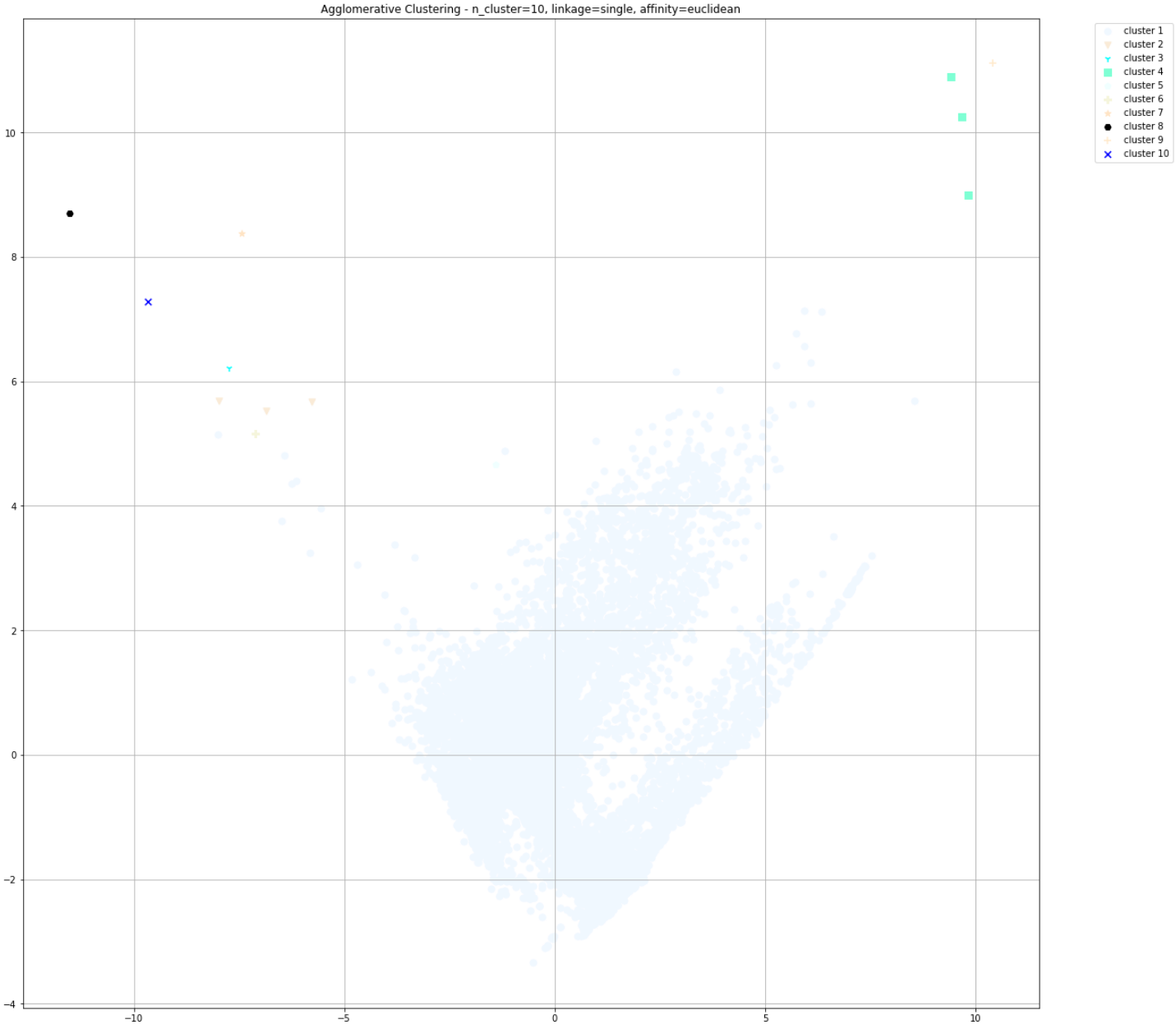


Figure 24: Agglomerative Clustering- Training Samples