Data Mining Project

JATIN BABBAR
SHIRIN BASHIRIAMID
MADHUBHANI RANCHA GODAGE

M2 MLDM

Supervisor:
Prof. Christine LARGERON
Prof. Baptiste JEUDY

Université Jean Monnet
Saint Étienne, France

31 January 2021

Contents

List of Figures

CHAPTER 1

Introduction

In the modern world, following the growth of number of social platforms and their users, we face some contents that are spam. Twitter as a popular platform is an attractive virtual place for spammers to post unwanted content or request. Detecting spam is a critical step to protect the servers from being overloaded with fake users by checking messages or accounts to recognize spam. This method is a classification task based on machine learning approaches that considers different features to classify the spam accounts automatically. Also it uses pattern to study the behavior of spammers and legitimate users.

In this study we consider Twitter's social spam using machine learning and data analysis techniques in python. The report is organized as follows. Chapter 2 includes data analysis part. Chapter 3 contains detail of machine learning approaches for spam detection. Next chapter presents the experiment and result analysis. Final chapter 5 is the conclusion which gives overall understanding about the achievement of projects.

CHAPTER 2

Data Analysis

2.1 Features for Spam Detection

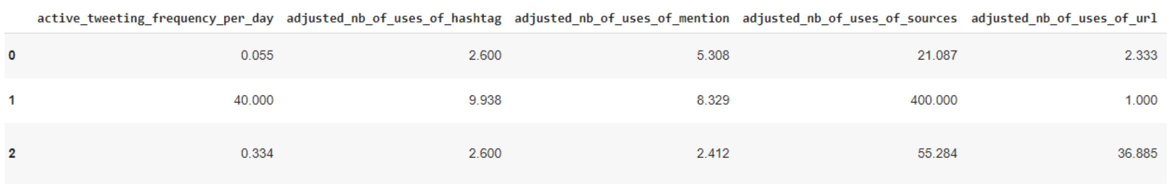
The dataset contains a set of 145 features for each account under observation, that are extracted from the users profiles, published content, behavior and relationships on the network.

2.2 Predictive Power of the Features

Selecting the best subset of feature set by calculating the predictive power of features results more interpretable model. It helps to identify the features that can impact the most to another features.

There is no one specific method for determining the predictive power of set of variables for a dependent variable and there is no universal metric which can tell the exact predictive power. It assists to find the important features that impact the dependent feature's behavior. Following two advantages will be gained by finding predictive power of features,

- Enhance the quality of the models by considering on important features
- Get rid of irrelevant features that do not contribute to the models performance



	active_tweeting_frequency_per_day	adjusted_nb_of_uses_of_hashtag	adjusted_nb_of_uses_of_mention	adjusted_nb_of_uses_of_sources	adjusted_nb_of_uses_of_url
0	0.055	2.600	5.308	21.087	2.333
1	40.000	9.938	8.329	400.000	1.000
2	0.334	2.600	2.412	55.284	36.885

FIGURE 2.1: Partition of given data

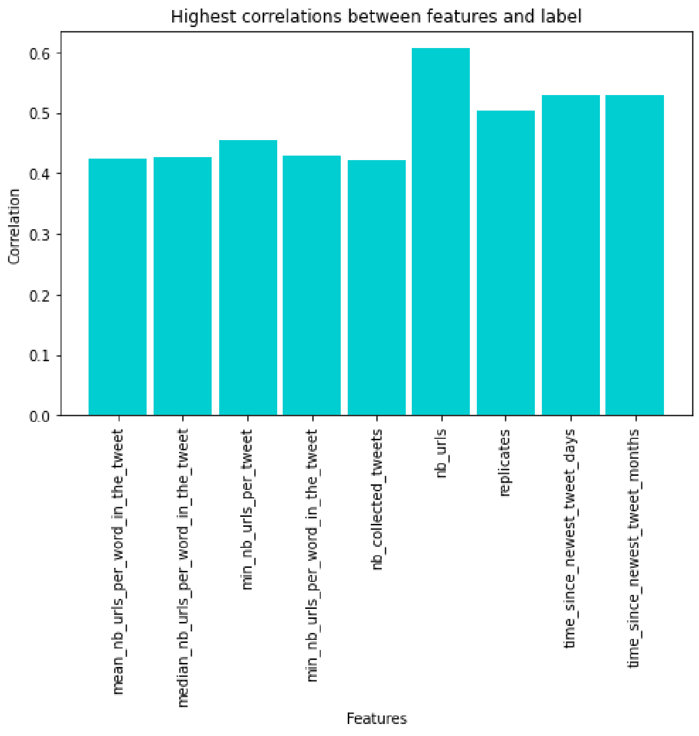


FIGURE 2.2: Feature Correlation

2.2.1 Correlation

In this practical work, first method used to find predictive power of features is the correlation model [4, 7]. It gives the possible linear relationship between variables. Pearson correlation coefficient between all the features and 'label' variable are calculated. Threshold is set to 0.4 and found nine features that have correlation is grater than 0.4.

Correlation between `nb_urls` and label is 0.6060, which gives the highest value among all. `time_since_newest_tweet_months` and `time_since_newest_tweet_days` are the next highest important features. Next one is `replicates` with the correlation of 0.504. Other five values are less than 0.5 and more than 0.4 to the dependent variable.

Other than checking correlation between each feature and label feature, we checked the correlation between features of the data set. Then we could identified predictive features of all the features of the data set. For example as shown in the 2.3 `hashtags_used_on_average` and `adjusted_nb_of_uses_of_hashtag` is 0.81. As it is a high value, we could say symmetrically they both are predictive features for them.

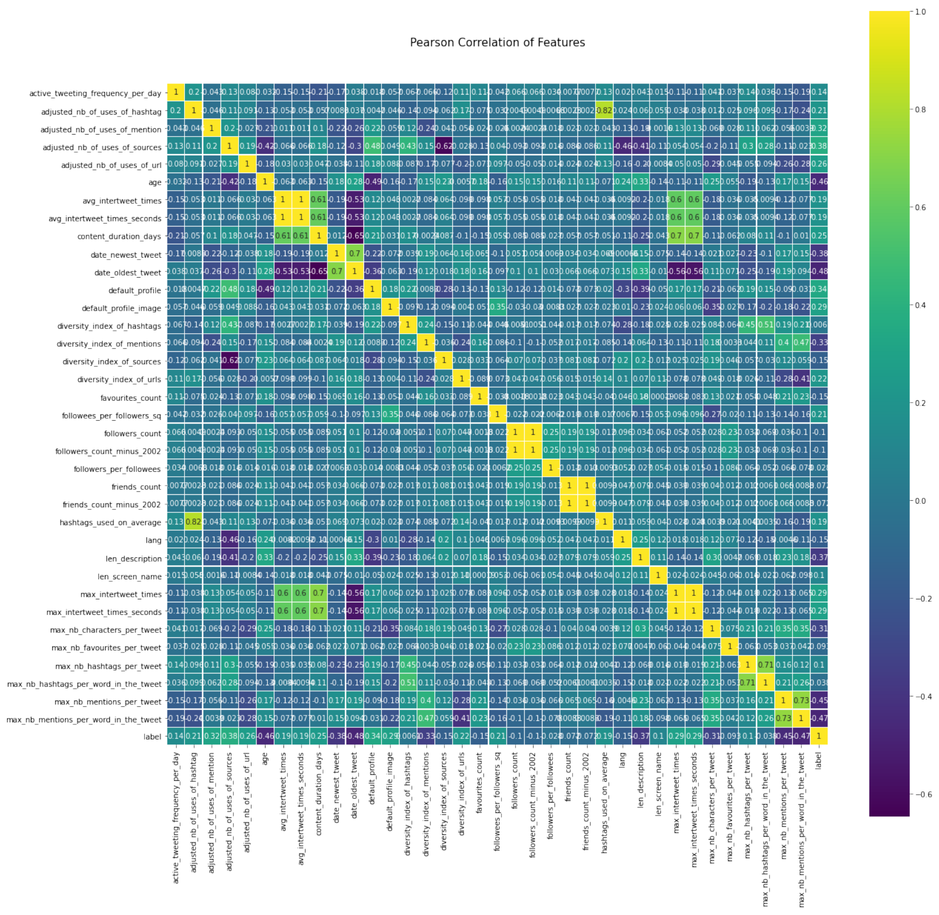


FIGURE 2.3: Part of Heatmap

2.2.2 Random Forest

In random forest, a subset of data (bootstrap) is fed to decision trees with a large depths randomly. The final result is obtained by counting the majority of the vote from all the decision trees. This technique is called as bootstrapping or bagging. This also helps in overcoming bias [5]. Using the default feature importance of Scikit-learn, we identified most important features with the values as shown in Figure 2.4.

From this Figure 2.4, we can identify that, `favourites_count` and `nb_urls` shows the highest predictive power over the other features.

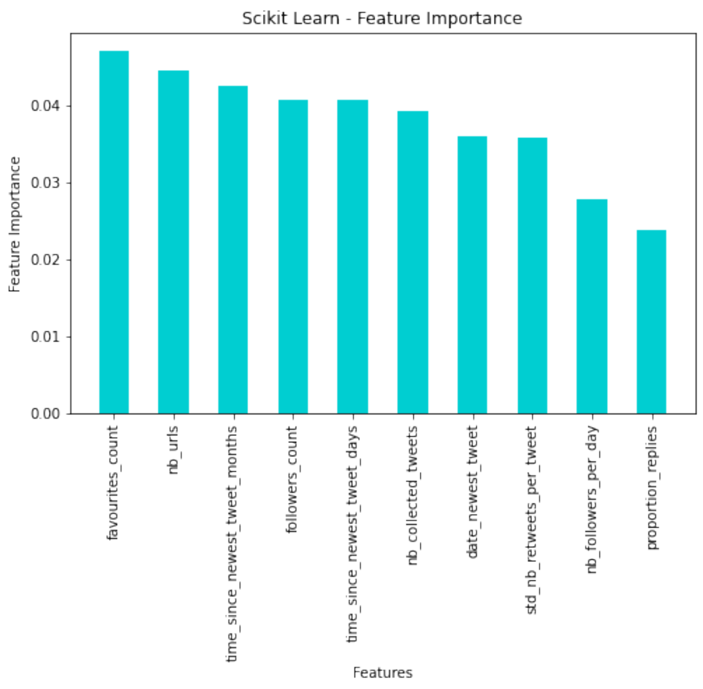


FIGURE 2.4: Random Forest

2.2.3 Predictive Power Score (PPS)

The Predictive Power Score is an another method to find predictive power of feature. Not like correlation, PPS is complex. It finds more patterns in the data set by detecting both linear and non-linear relationships between features. PPS is a normalized metric that assists to understand the importance of variables to predict a variable. Values are between 0 and 1 showing the importance. If the value is high the feature is important [6].

From this Figure 2.5, we can identify the features, which affects more on the prediction as `nb_collected_tweets` and `nb_urls`.

Finally we can conclude that, we get almost similar features as most affecting features on the label from all these methods.

2.3 Additional features

We create additional features in two different files.

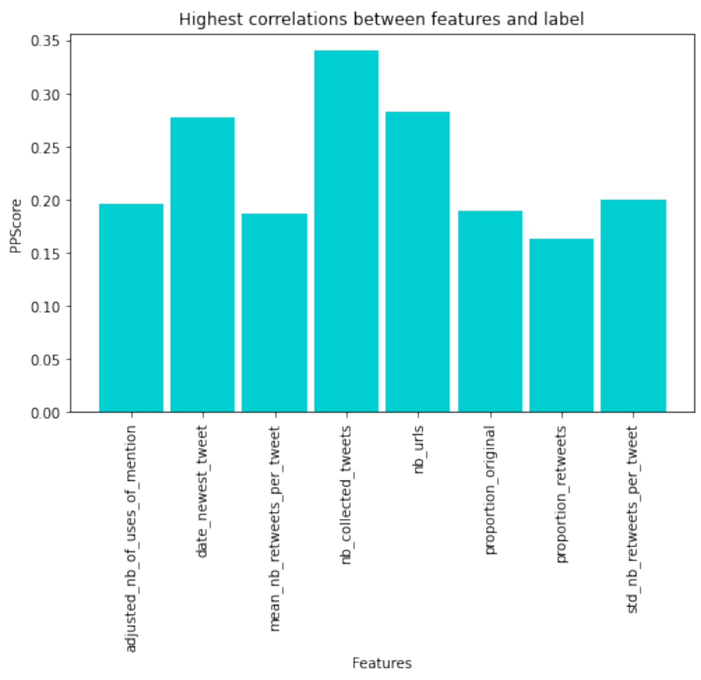
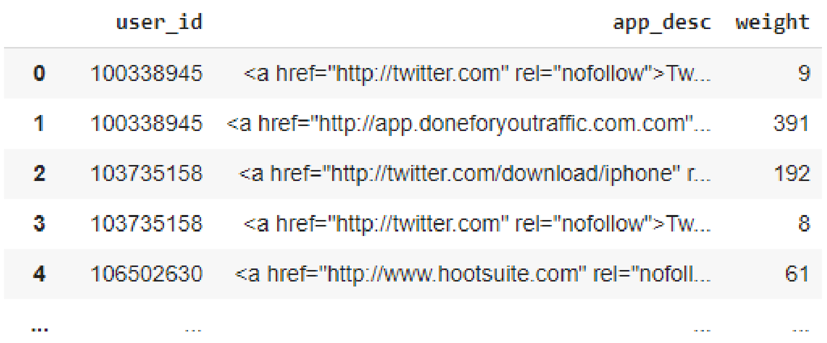


FIGURE 2.5: PPS



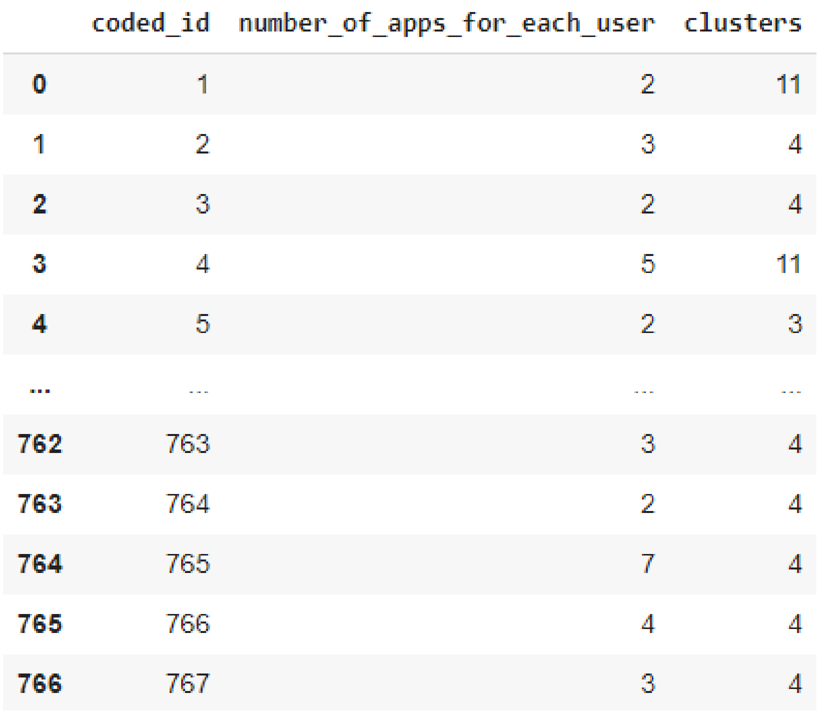
	user_id	app_desc	weight
0	100338945	<a href="http://twitter.com" rel="nofollow">Tw...	9
1	100338945	<a href="http://app.doneforyoutraffic.com.com"...	391
2	103735158	<a href="http://twitter.com/download/iphone" r...	192
3	103735158	<a href="http://twitter.com" rel="nofollow">Tw...	8
4	106502630	<a href="http://www.hootsuite.com" rel="nofoll...	61
...	...	...	...

FIGURE 2.6: weighted_app_user_edges.csv

As the first feature, we introduce number of applications used by each user from the given data. We have a table with the name of weighted_app_user_edges.csv (Figure 2.6).

By studying this dataset, we realize that one user uses more than one application. So number_of_apps_for_each_user_df feature is introduced.

As the second feature, we create clusters and introduce the cluster number from sim_matrix.txt file.



	coded_id	number_of_apps_for_each_user	clusters
0	1	2	11
1	2	3	4
2	3	2	4
3	4	5	11
4	5	2	3
...	...	...	...
762	763	3	4
763	764	2	4
764	765	7	4
765	766	4	4
766	767	3	4

FIGURE 2.7: Generated new features

To perform this task, we use SpectralClustering() method in sklearn and set number of clusters as 12. Cluster number for each user is added as a new feature.

Figure 2.7 shows the generated new features.

2.4 Build / visualize the graph of the users

Data visualization is the graphical representation of data and information extracted from data mining. They can be display as graphs, charts and maps. Data visualization helps to analyze data and information to make decisions [2]. Under this topic we describe how we represent given data in graph elements. We found several tools for graph representation such as graph-tool, igraph and NetworkX. In this practical work, we use NetworkX library[1].

Task 1. c_combined_edges_w_1_s_0.9.csv file contains the weight and similarity between two users. Weight is number of same messages sent between them and similarity is calculated according to application they used tweet. This file contains only the similarity >= 0.9 occasions. Nodes will represent all the users. If the similarity is higher two nodes are closer,

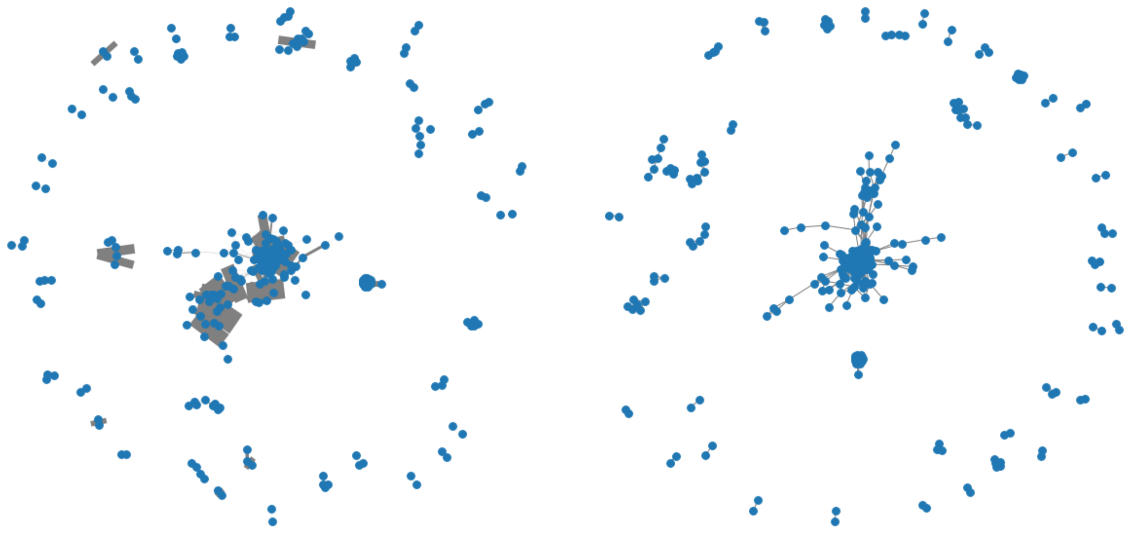


FIGURE 2.8: Network Graph with weight and without weight

that means the similarity is representing as the edge length. If the weight is higher, the thickness of edge is higher. Edge width is representing the weight of two users. Figure 2.8 shows the network graph of this data with and without considering the weight.

Task 2. c_combined_edges_w_2_s_0.9.csv file contains the same data as c_combined_edges_w_1_s_0.9.csv file. In addition to that, this file contains data only the weight >= 2. Nodes will represent all the users.

Task 3. To draw the app_user_interaction_graph, we have two csv files. As both have the same data, we used coded_weighted_edges.csv with user_id, app_id and weight. user_id and app_id both had labels of common integers that. So we have changed the labels of app_ids by concatenating a string. Then we draw bipartatite graphs. To make the edges visible, we can reduce the data size as shown in Figure 2.10.

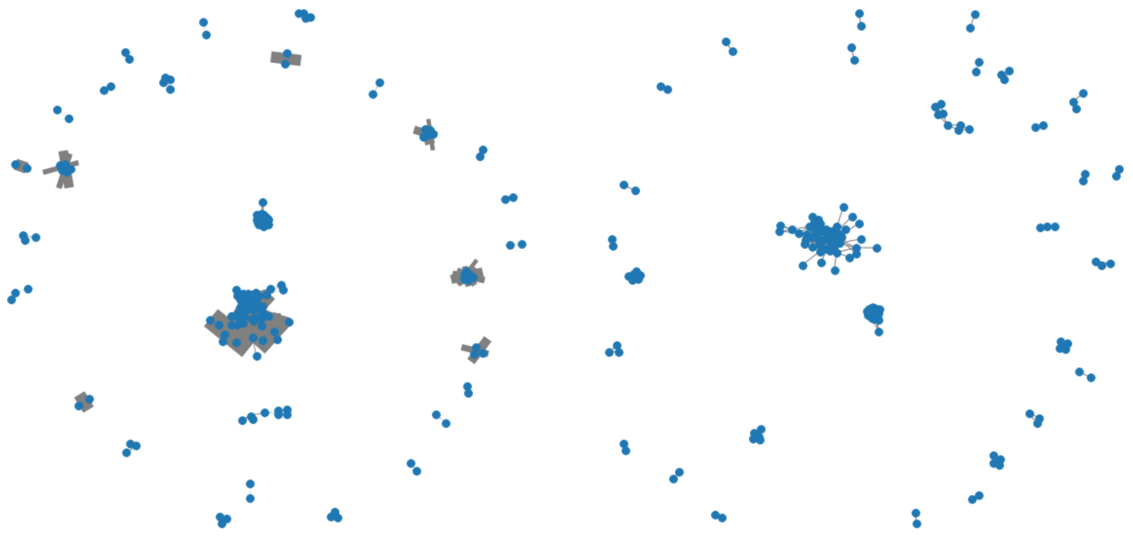


FIGURE 2.9: Network Graph with weight and without weight

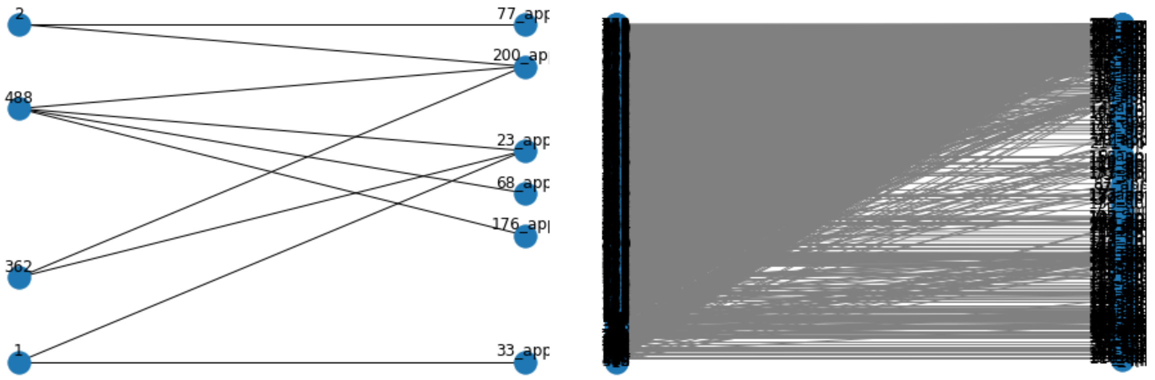


FIGURE 2.10: Bipartite Graph with less data and with full data

2.5 Identify groups of users or communities and characterize them

The interest of using networks is finding a way to divide them in communities. We have different approaches to characterize users' group and we select two of popular methods "Modularity" and "Louvain Algorithm".

First we apply modularity to measure the structure of users based on their similarities. In this approach all nodes are assign to a community and the modularity is the fraction of edges that

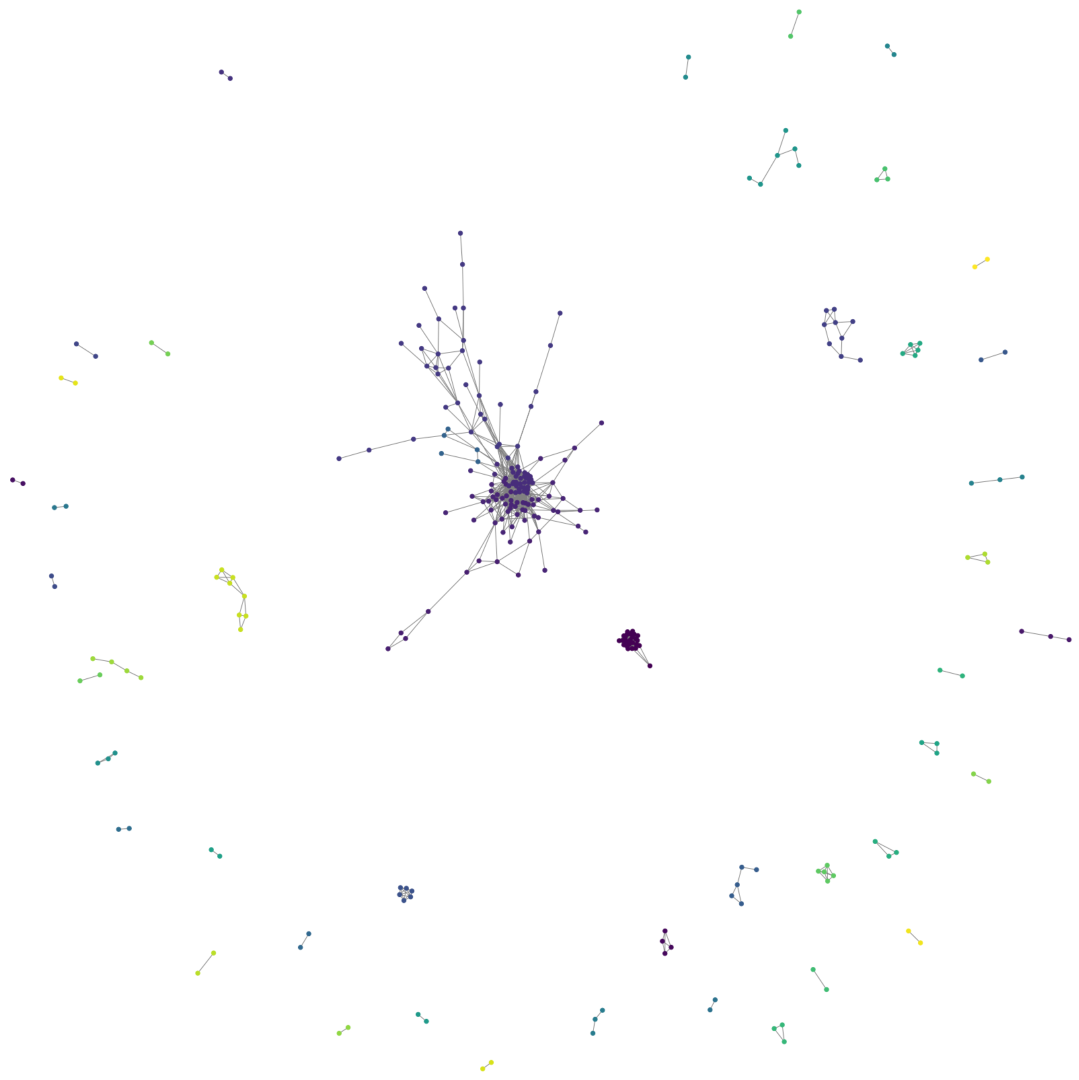


FIGURE 2.11: Community Detection using Modularity

fall in a community minus the number of expected edges in the community over the number of the whole edges in the network.

The modularity equals zero means that nodes are in the same community and it computes the optimal number of communities.

The figure 2.11 shows the communities based on Modularity. The number of communities are 48 and assigned in different colors.

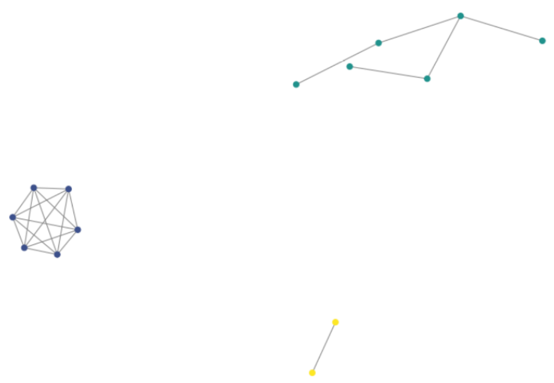


FIGURE 2.12: Part of a Community Detection graph

"Louvain algorithm" is a hierarchical clustering algorithm and maximizes a modularity scores for each community.

The first step of detection with Louvain is considering each nodes as a community. So, the number of communities are equal the number of nodes.Then each nodes check the neighbors for the overall modularity if the modularity increases by moving a certain nodes to neighbor's partition. The aim is gaining the highest modularity and this procedure will repeat until no improvements can be achieved.

The figure 2.13 shows the communities of our network based on Louvain algorithm. It assigns 48 communities to our users based on their similarity that are shown in different colors.

2.5.1 Hierarchical Clustering Dendogram

To study our records based on sequences of merges or splits we will use Hierarchical Clustering Dendogram. We will be working on Similarities file. We will segment users based on the Weight and Similarities of application they use. The values represent number of messages with the same content sent by the users whereas the app-based similarity evaluates the similarity according to the applications used to post their tweets by the users. Our aim is to make clusters from this data that can segment similar users together. We will use Hierarchical Clustering for this problem [3].

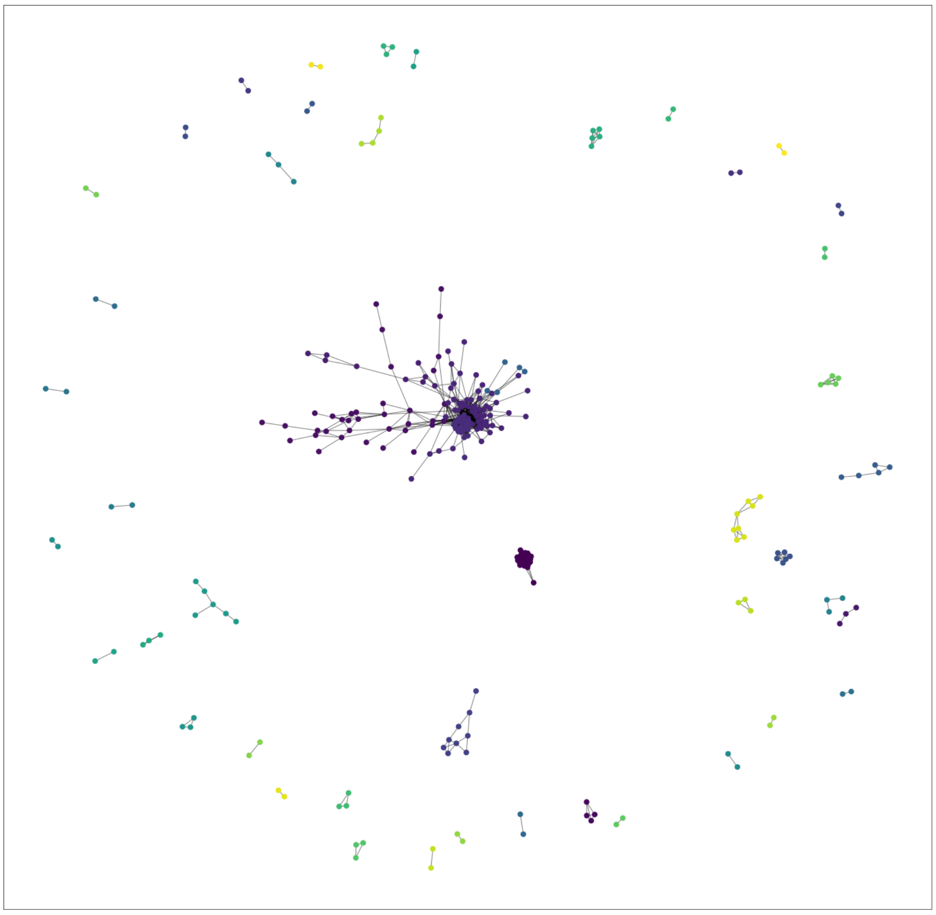
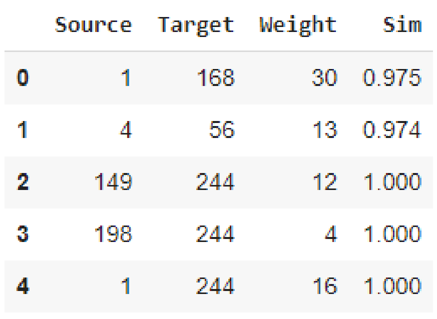


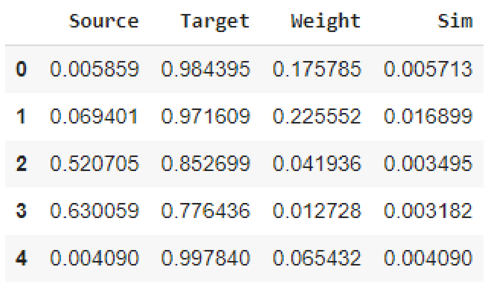
FIGURE 2.13: Community Detection graph with Louvain algo



	Source	Target	Weight	Sim
0	1	168	30	0.975
1	4	56	13	0.974
2	149	244	12	1.000
3	198	244	4	1.000
4	1	244	16	1.000

FIGURE 2.14: Similarities table

But before applying Hierarchical Clustering, we have to normalize the data so that the scale of each variable is the same. if the scale of the variables is not the same, the model might become biased towards the variables with a higher magnitude.



	Source	Target	Weight	Sim
0	0.005859	0.984395	0.175785	0.005713
1	0.069401	0.971609	0.225552	0.016899
2	0.520705	0.852699	0.041936	0.003495
3	0.630059	0.776436	0.012728	0.003182
4	0.004090	0.997840	0.065432	0.004090

FIGURE 2.15: Normalized Similarities table

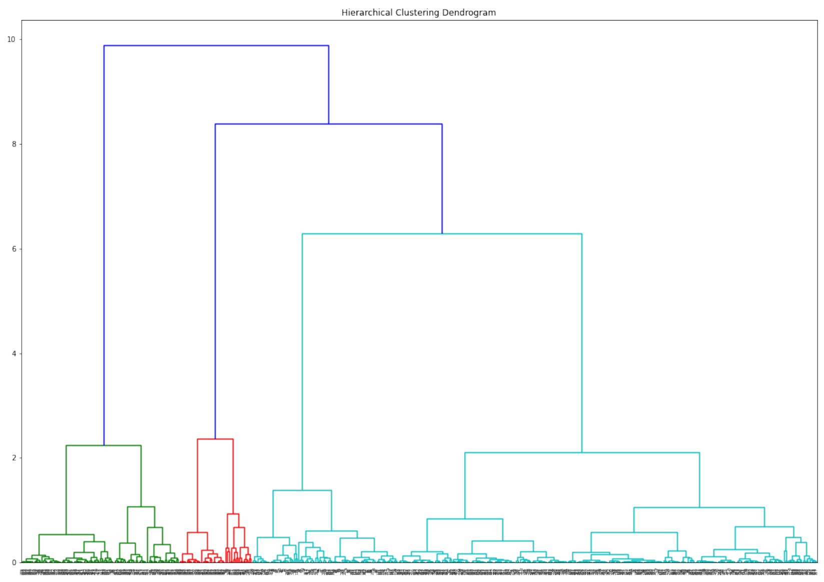


FIGURE 2.16: Hierarchical Clustering Dendogram

Then we will draw the dendrogram to help us decide the number of clusters.

The vertical line with maximum distance is the blue and the blue black lines and hence we can decide a threshold of 5 and cut the dendrogram.

We have Four clusters as this line cuts the dendrogram at Four points. Now we apply hierarchical clustering for 4 clusters.

We can see the values of 0,1,2,3 in the output since we defined 4 clusters. 0 represents the points that belong to the first cluster, 1 represents points in the second cluster, 2 represents points in the third cluster, 3 represents points in the fourth cluster (Figure 2.19).

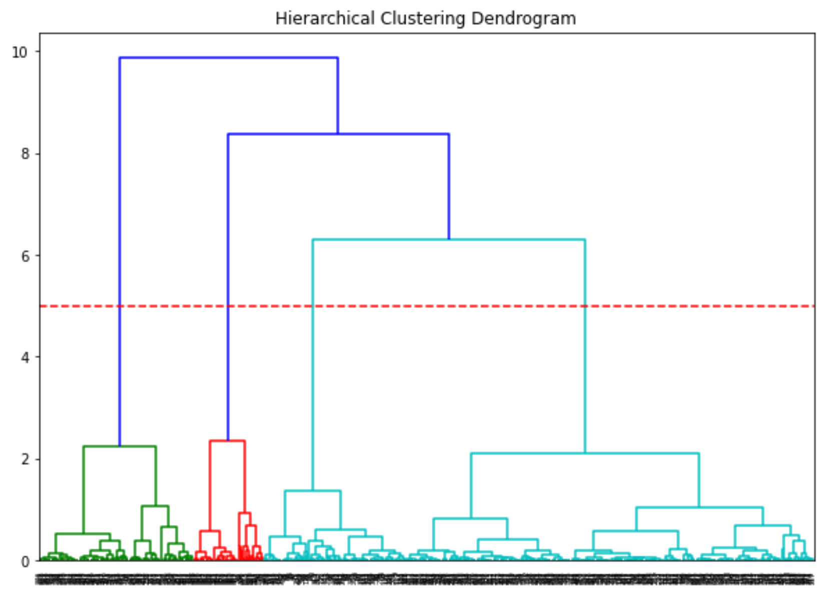
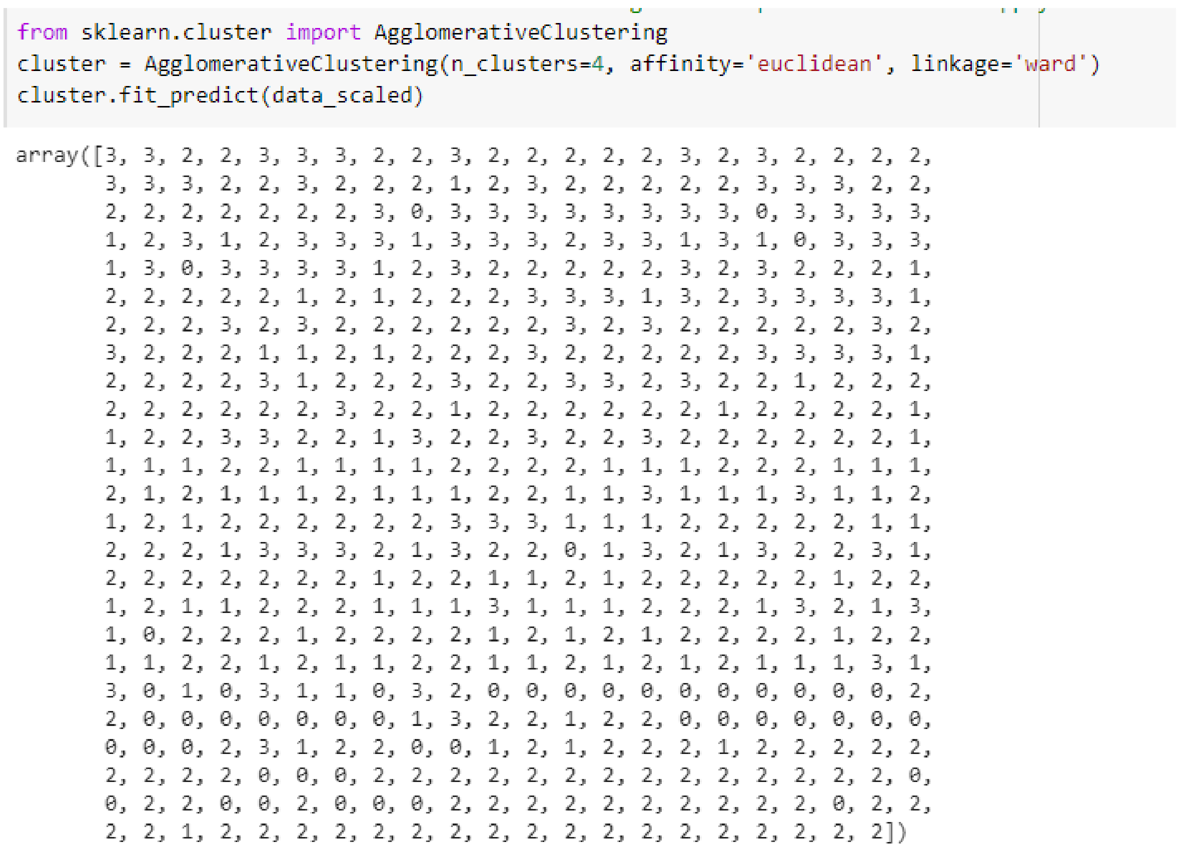


FIGURE 2.17: Dendrogram with clusters



```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='ward')
cluster.fit_predict(data_scaled)
```

```
array([3, 3, 2, 2, 3, 3, 3, 2, 2, 3, 2, 2, 2, 2, 2, 3, 2, 3, 2, 2, 2, 2,
       3, 3, 3, 2, 2, 3, 2, 2, 2, 1, 2, 3, 2, 2, 2, 2, 2, 3, 3, 3, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 3, 0, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3, 3,
       1, 2, 3, 1, 2, 3, 3, 3, 1, 3, 3, 3, 2, 3, 3, 1, 3, 1, 0, 3, 3, 3,
       1, 3, 0, 3, 3, 3, 3, 1, 2, 3, 2, 2, 2, 2, 2, 3, 2, 3, 2, 2, 2, 1,
       2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 3, 3, 3, 1, 3, 2, 3, 3, 3, 3, 1,
       2, 2, 2, 3, 2, 3, 2, 2, 2, 2, 2, 2, 3, 2, 3, 2, 2, 2, 2, 2, 3, 2,
       3, 2, 2, 2, 1, 1, 2, 1, 2, 2, 2, 3, 2, 2, 2, 2, 2, 3, 3, 3, 3, 1,
       2, 2, 2, 2, 3, 1, 2, 2, 2, 3, 2, 2, 3, 3, 2, 3, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 3, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1,
       1, 2, 2, 3, 3, 2, 2, 1, 3, 2, 2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 2, 1,
       1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 1, 1, 1,
       2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 3, 1, 1, 1, 3, 1, 1, 2,
       1, 2, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1,
       2, 2, 2, 1, 3, 3, 3, 2, 1, 3, 2, 2, 0, 1, 3, 2, 1, 3, 2, 2, 3, 1,
       2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2,
       1, 2, 1, 1, 2, 2, 2, 1, 1, 1, 3, 1, 1, 1, 2, 2, 2, 1, 3, 2, 1, 3,
       1, 0, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2,
       1, 1, 2, 2, 1, 2, 1, 1, 2, 2, 1, 1, 2, 1, 2, 1, 2, 1, 1, 1, 3, 1,
       3, 0, 1, 0, 3, 1, 1, 0, 3, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,
       2, 0, 0, 0, 0, 0, 0, 0, 1, 3, 2, 2, 1, 2, 2, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 2, 3, 1, 2, 2, 0, 0, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0,
       0, 2, 2, 0, 0, 2, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2,
       2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

FIGURE 2.18: Hierarchical clustering for 4 clusters

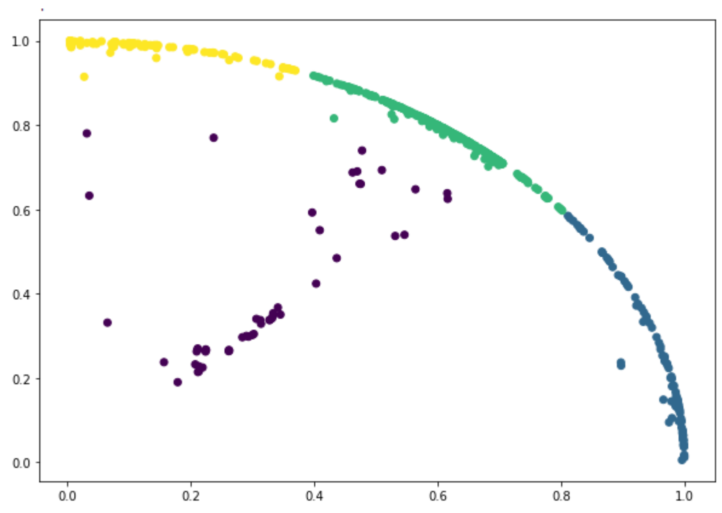


FIGURE 2.19: Visualizing the 4 clusters

With out having pre-define number of clusters we came up to conclude that their are 4 clusters of users based on number of messages with the same content sent by the users and similarity according to the applications used to post their tweets by the users.

2.6 Study if there are applications mainly used by abusive users

Figure 2.20 shows the graphical representation of applications for the label of 0 (legitimate user) and 1 (spam user). As we can see, there is a set of applications only used by legitimate users and there is another set of applications only used by spam users. And also, there is a separate set of applications used by both legitimate users and spam users.

So, we can clearly identify that there are applications mainly used by abusive users.

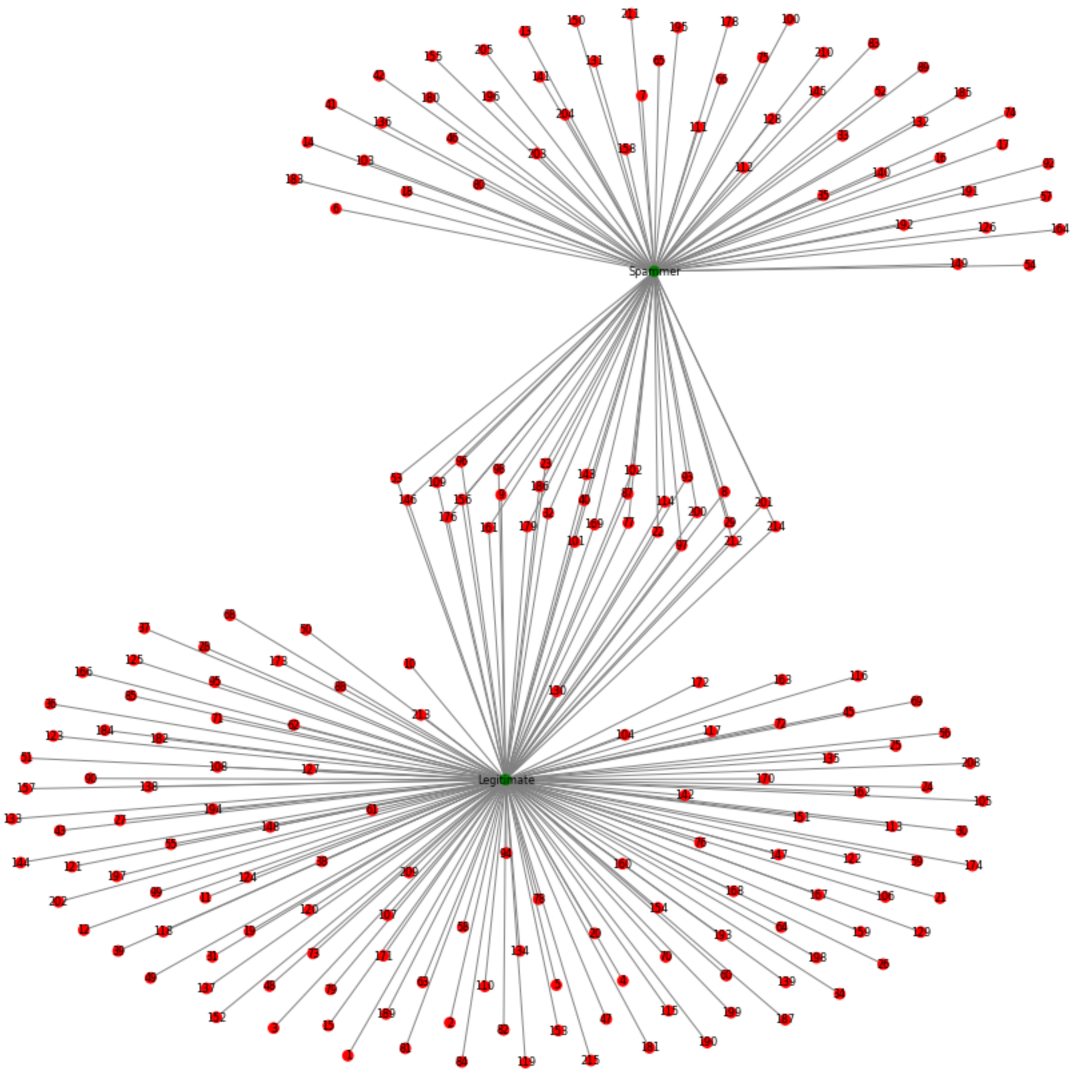


FIGURE 2.20: Applications of abusive users

CHAPTER 3

Spam Detection using Machine Learning

In this section, we discuss the machine learning methods and other techniques we use for the task of Spam Detection and also predicting the classes for the unlabeled data.

Out of a total of 767 samples, there are 81 unlabeled samples. This set of 81 unlabeled samples forms the test set. Out of the remaining 686 labeled samples, we use 600 for training the machine learning model, and 86 samples as the validation set.

So, the three datasets have the following size:

- Train set: 600 samples
- Validation set: 86 samples
- Test set: 81 samples

3.1 Data Cleaning

While exploring the datasets, we observe that the dataset has a combination of features of numerical type, categorical type, and date-time type. Out of the 145 features, we try to use the features that have more importance with respect to making predictions for the target variable.

First, we observe that the column *spam_in_screen_name* has a missing value for each sample in the train and the validation set. Consequently, we remove this feature.

Next, we observe that there are a few features that have multiple instance, just in a different form or unit, which are mentioned as follows:

- avg_intertweet_times (avg_intertweet_times_seconds)
- max_intertweet_times (max_intertweet_times_seconds)

- min_intertweet_times (min_intertweet_times_seconds)
- std_intertweet_times (std_intertweet_times_seconds)
- followers_count (followers_count_minus_2002)
- friends_count (friends_count_minus_2002)

We keep only one instance of each feature mentioned above, to maintain the dimensionality of the dataset. After removing these features, we are left with 138 features.

Next, we encode the categorical features (*lang* and *time_zone*), and transform the date-time features to timestamp (*date_newest_tweet* and *date_oldest_tweet*) to be able to feed to the model for training.

We also observe that the feature *utc_offset* has a missing value in 168 samples. We fill up the missing values in the feature with 0.

3.2 Feature Selection

In order to reduce the dimensionality of the dataset, we need to remove the unnecessary features from the dataset. For this, we use statistical function to calculate the importance of the features with respect to the target variable. We choose correlation ratio and chi-square score as our primary criteria for selecting relevant features.

We calculate the correlation of each of the 138 features with the target variable. With chi-square score, we filter the features by selecting top 40 features based on the highest chi-square score. Then, we generate feature groups of different sizes to train the machine learning model and compare the results. The feature groups are as follows:

(1) all the features into consideration
(2) features with a correlation score with the labels of more than 0.2
(3) features with a correlation score with the labels of more than 0.3
(4) top 30 features according to the highest chi-square scores
(5) top 50 features according to the highest chi-square scores
(6) top 80 features according to the highest chi-square scores

3.3 Learning models used

We try with different machine learning models and compare the results to find the best model, which will be used to predict the labels for the test set.

We use the following machine learning models for our experiment:

- K Nearest Neighbors Classifier with n_neighbors=5
- Decision Tree Classifier
- Random Forest Classifier with n_estimators=100
- Random Forest Classifier with n_estimators=200
- Gradient Boosting Classifier with n_estimators=100
- Gradient Boosting Classifier with n_estimators=200

CHAPTER 4

Results

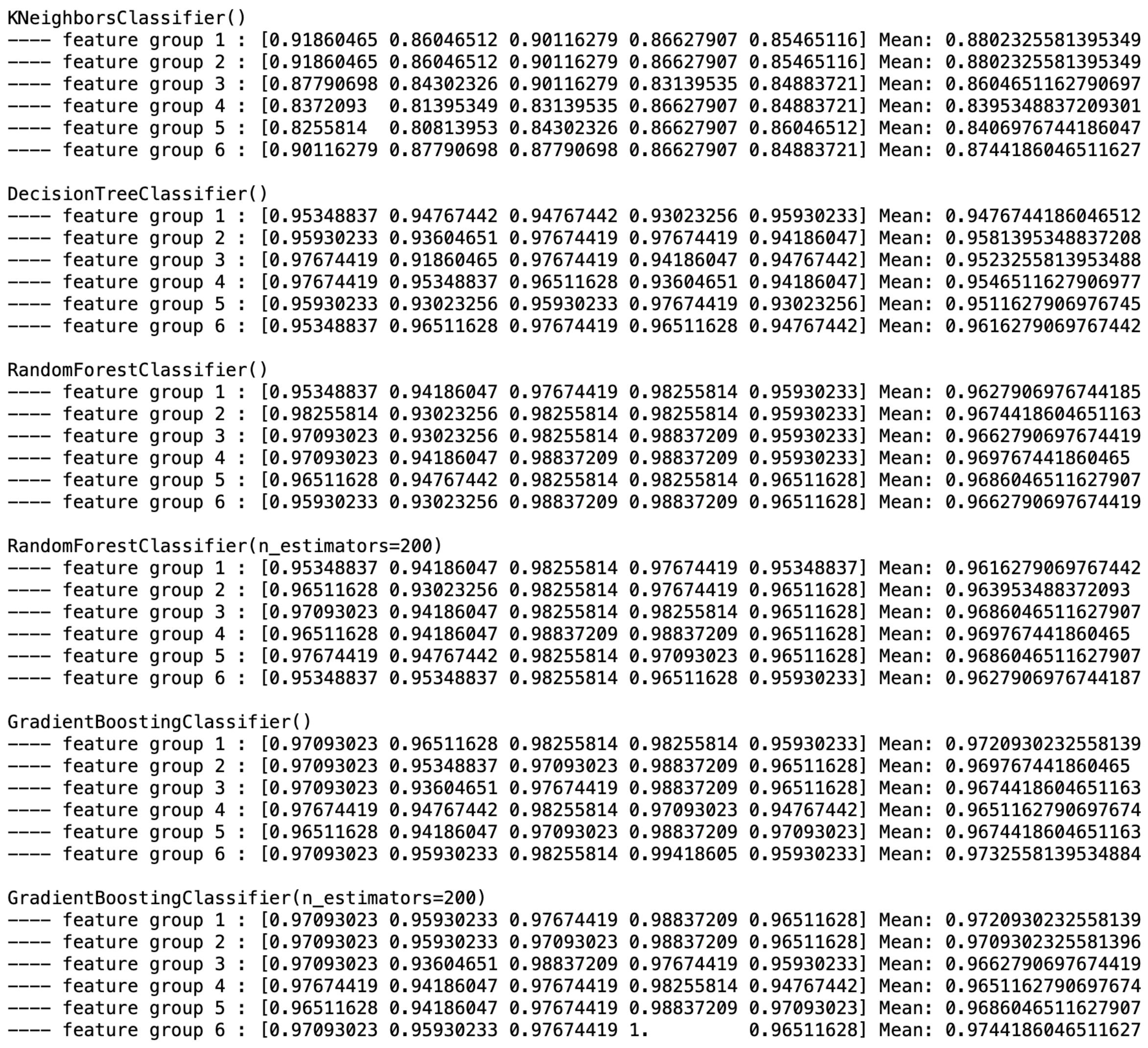
We experiment with 6 different feature groups, 4 types of machine learning with different hyper-parameters (number of estimators). In this section, we display the results of all the experiments, make a comparison, and select the highest scoring pair of model and feature group to make prediction for the test set.

For each feature group, we perform cross validation with 5 folds, and we take an average of the results obtained from cross validation. With this, we have 6 accuracy scores for each feature group (from different models).

Figure 4.1 shows the results for cross validation results for all feature groups with different machine learning models. The best results for each feature group are shown in Table 4.1. Based on these results, we can observe that the best score obtained is **0.97441**, with Gradient Boosting Classifier (n_estimators=200) and the feature group that contains the top 80 features with highest chi square score. We use this pair of model and feature group to predict the labels for the test set.

Group No.	Classifier	Hyperparameters	Score
1	Gradient Boosting Classifier	n_estimator=200	0.97209
2	Gradient Boosting Classifier	n_estimator=200	0.97093
3	Random Forest Classifier	n_estimator=200	0.96860
4	Random Forest Classifier	n_estimator=200	0.96976
5	Random Forest Classifier	n_estimator=200	0.96860
6	Gradient Boosting Classifier	n_estimator=200	0.97441

TABLE 4.1: Best performing models and their results for each feature group



```
KNeighborsClassifier()
---- feature group 1 : [0.91860465 0.86046512 0.90116279 0.86627907 0.85465116] Mean: 0.8802325581395349
---- feature group 2 : [0.91860465 0.86046512 0.90116279 0.86627907 0.85465116] Mean: 0.8802325581395349
---- feature group 3 : [0.87790698 0.84302326 0.90116279 0.83139535 0.84883721] Mean: 0.8604651162790697
---- feature group 4 : [0.8372093  0.81395349 0.83139535 0.86627907 0.84883721] Mean: 0.8395348837209301
---- feature group 5 : [0.8255814  0.80813953 0.84302326 0.86627907 0.86046512] Mean: 0.8406976744186047
---- feature group 6 : [0.90116279 0.87790698 0.87790698 0.86627907 0.84883721] Mean: 0.8744186046511627

DecisionTreeClassifier()
---- feature group 1 : [0.95348837 0.94767442 0.94767442 0.93023256 0.95930233] Mean: 0.9476744186046512
---- feature group 2 : [0.95930233 0.93604651 0.97674419 0.97674419 0.94186047] Mean: 0.9581395348837208
---- feature group 3 : [0.97674419 0.91860465 0.97674419 0.94186047 0.94767442] Mean: 0.9523255813953488
---- feature group 4 : [0.97674419 0.95348837 0.96511628 0.93604651 0.94186047] Mean: 0.9546511627906977
---- feature group 5 : [0.95930233 0.93023256 0.95930233 0.97674419 0.93023256] Mean: 0.9511627906976745
---- feature group 6 : [0.95348837 0.96511628 0.97674419 0.96511628 0.94767442] Mean: 0.9616279069767442

RandomForestClassifier()
---- feature group 1 : [0.95348837 0.94186047 0.97674419 0.98255814 0.95930233] Mean: 0.9627906976744185
---- feature group 2 : [0.98255814 0.93023256 0.98255814 0.98255814 0.95930233] Mean: 0.9674418604651163
---- feature group 3 : [0.97093023 0.93023256 0.98255814 0.98837209 0.95930233] Mean: 0.9662790697674419
---- feature group 4 : [0.97093023 0.94186047 0.98837209 0.98837209 0.95930233] Mean: 0.969767441860465
---- feature group 5 : [0.96511628 0.94767442 0.98255814 0.98255814 0.96511628] Mean: 0.9686046511627907
---- feature group 6 : [0.95930233 0.93023256 0.98837209 0.98837209 0.96511628] Mean: 0.9662790697674419

RandomForestClassifier(n_estimators=200)
---- feature group 1 : [0.95348837 0.94186047 0.98255814 0.97674419 0.95348837] Mean: 0.9616279069767442
---- feature group 2 : [0.96511628 0.93023256 0.98255814 0.97674419 0.96511628] Mean: 0.963953488372093
---- feature group 3 : [0.97093023 0.94186047 0.98255814 0.98255814 0.96511628] Mean: 0.9686046511627907
---- feature group 4 : [0.96511628 0.94186047 0.98837209 0.98837209 0.96511628] Mean: 0.969767441860465
---- feature group 5 : [0.97674419 0.94767442 0.98255814 0.97093023 0.96511628] Mean: 0.9686046511627907
---- feature group 6 : [0.95348837 0.95348837 0.98255814 0.96511628 0.95930233] Mean: 0.9627906976744187

GradientBoostingClassifier()
---- feature group 1 : [0.97093023 0.96511628 0.98255814 0.98255814 0.95930233] Mean: 0.9720930232558139
---- feature group 2 : [0.97093023 0.95348837 0.97093023 0.98837209 0.96511628] Mean: 0.969767441860465
---- feature group 3 : [0.97093023 0.93604651 0.97674419 0.98837209 0.96511628] Mean: 0.9674418604651163
---- feature group 4 : [0.97674419 0.94767442 0.98255814 0.97093023 0.94767442] Mean: 0.9651162790697674
---- feature group 5 : [0.96511628 0.94186047 0.97093023 0.98837209 0.97093023] Mean: 0.9674418604651163
---- feature group 6 : [0.97093023 0.95930233 0.98255814 0.99418605 0.95930233] Mean: 0.9732558139534884

GradientBoostingClassifier(n_estimators=200)
---- feature group 1 : [0.97093023 0.95930233 0.97674419 0.98837209 0.96511628] Mean: 0.9720930232558139
---- feature group 2 : [0.97093023 0.95930233 0.97093023 0.98837209 0.96511628] Mean: 0.9709302325581396
---- feature group 3 : [0.97093023 0.93604651 0.98837209 0.97674419 0.95930233] Mean: 0.9662790697674419
---- feature group 4 : [0.97674419 0.94186047 0.97674419 0.98255814 0.94767442] Mean: 0.9651162790697674
---- feature group 5 : [0.96511628 0.94186047 0.97674419 0.98837209 0.97093023] Mean: 0.9686046511627907
---- feature group 6 : [0.97093023 0.95930233 0.97674419 1.         0.96511628] Mean: 0.9744186046511627
```

FIGURE 4.1: Cross validation results for all feature groups with different models and hyperparameters

CHAPTER 5

Conclusion

In this section, we summarize what we have done, in Data Analysis and Machine Learning parts.

5.1 Data Analysis

Identifying important features is very important because it helps to enhance the model by considering only important predictive features. By removing irrelevant features, model performance improves. Traditional method of identifying unimportant features is correlation while modern methods are efficient and also complex. There are few other methods to measure feature importance; Permutation and XGBooster classifier.

By analysing the dataset, we could generate new features from the given data. They are number of application used by each user and cluster number.

Data visualization is a graphical representation of data or information. It assists to different visualization methods to see the data in different point of views to make decisions. In this section we used different graph visualization techniques like Network analysis and clustering. Other than normal graphs, here we used bipartite graphs which divided the nodes in to disjoint and independent sets. It helped to identified spam users of twitter application.

We studied connections of user to the other users, grouping of users based on graph topology so that it will help us to have to identify spammers from the real users. By using network analysis we were able to see how each users were connected to others, studied strength of the relationship based on the weight and similarity between them, visualize legitimate users and spammers.

For clustering we used Hierarchical clustering technique,It's sometimes referred to as community detection based on its commonality in social network analysis. So we tried to have clusters based on number of messages with the same content sent by the users and similarity according to the applications used to post their tweets by the users.

5.2 Machine Learning for predictions

For making predictions to the test set, we experimented with different techniques for feature selection and generated 6 feature groups with different number of features. Then, we experimented with several machine learning algorithms with different hyperparameters to analyze which model performed best with which feature group. Finally, we used the best pair of model and feature group to fit the model and make predictions on the test set.

For our experiments, we observed that the best results were achieved with the Gradient Boosting Classifier with n_estimators=200 and the feature group 6, which has the top 80 features based on the highest chi square score. We used this model to make predictions and saved the predictions in the file `coded_ids_labels_test.csv`.

The code for the project is available on the GitHub repository `https://github.com/thejatinbabbar/spam-detection`.

Bibliography

[1] NetworkX Developers. *Converting to and from other data formats*. Accessed: 2021-01-31. URL: https://networkx.org/documentation/stable//reference/convert.html.

[2] EDUCBA. *Data Mining vs Data Visualization*. Accessed: 2021-01-31. URL: https://www.educba.com/data-mining-vs-data-visualization/.

[3] GeeksForGeeks. *Hierarchical Clustering in Data Mining*. Accessed: 2021-01-25. URL: https://www.geeksforgeeks.org/hierarchical-clustering-in-data-mining/.

[4] Firat Gonen. *Predictive Power vs Correlation*. Accessed: 2021-01-31. URL: https://www.kaggle.com/frtgnn/predictive-power-score-vs-correlation.

[5] Vishal Kesti. *Ranking features based on predictive power/importance of the class labels*. Accessed: 2021-01-31. URL: https://medium.com/analytics-vidhya/ranking-features-based-on-importance-predictive-power-with-respect-to-the-class-labels-of-the-25afaed71e90.

[6] Paul van der Laken. *Predictive Power Score: Finding predictive patterns in your dataset*. Accessed: 2021-01-31. URL: https://paulvanderlaken.com/2020/05/04/predictive-power-score-finding-patterns-dataset/.

[7] Sam Schwarzkopf. *Does correlation imply prediction?* Accessed: 2021-01-31. URL: https://neuroneurotic.net/2015/11/30/does-correlation-imply-prediction/.