DATA MINING FOR BIG DATA
Spam detection on Twitter

Camil Brahmi,Théotime Darmet, Anthony Lecerf, Duy Anh Philippe Pham*

January 27, 2021

Contents

We used python 3.7.3 with the following last dependencies:
- numpy
- matplotlib
- pandas
- networkx[3]
- imblearn
- sklearn[7]

You can also find our Notebook here.

*camil.brahmi@etu.univ-st-etienne.fr, theotime.darmet@etu.univ-st-etienne.fr, anthony.lecerf@etu.univ-st-etienne.fr, dap.pham@cpe.fr

1 Introduction

We use a dataset composed of 767 tweeter users classified in two sets, the training set composed of 118 spammer and 568 legitimate. And a test set composed of 81 non-labelled users. We have an unbalanced training set composed of 16% positive spammer label. On this data sheet we have different information distributed on different features. The most important one is an aggregate of different sources with 145 features. It expresses information about the user's profile on Tweeter. The second aggregate represents the platforms used by these users to tweet. It is composed of 215 features representing the different platforms.
It is important to note that there is a difference between coded_id and user_id. We have a table of equivalence allowing to make the conversion between these two identifiers. The coded_id corresponds to anonymous identifiers while the user_id corresponds to the user identifier on the tweeter platform. The same applies to applications.
We also have other information provided to us to help us make our model graphs. In particular, we are given the edges and nodes with the associated weights for both the users and the applications they use. It is important to note that we can regenerate similar information using the raw data.
First we do data analysis on our dataset in order to understand its main characteristics, we use graphs. After that, we will try to isolate the features of interest on our dataset both by human expertise and in a more systematic way. We will mainly work on the feature covariance matrix On another dataset related to applications used by individuals, we seek to enrich our original dataset by discriminating features, for this we make a statistical study.

2 Data Analysis

The purpose of this part is to make data visualization in order to provide an input to the relevant learning machine algorithm so that it can discriminate spammers from legitimates. There is also some attempt to provide more precise characteristics for different user groups.

2.1 Graph of the users based on these similarities and relationships

In this part our goal is to display the graph associated to the different weights that we have been provided in the dataset graph. This allows us to have an overview of the relationships between the different individuals and to try to delimit communities within this social network. Our first goal is to display the graph associated with the weights 1. This allows us to test our displays especially at the label level. We make the choice to put the green color to the legitimate, red to the spammer and blue for the unknown. The network was then displayed with the weight 2 which provided more information.

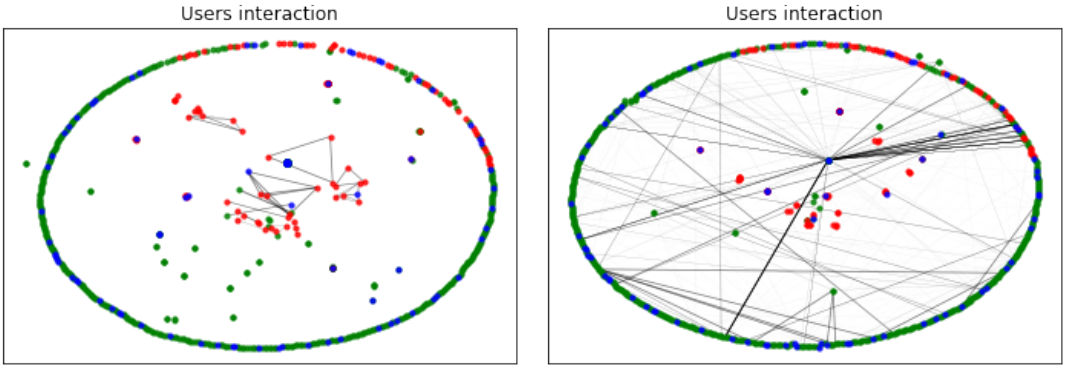


Figure 1: Graph depending on the weight

We can now see there are strong connections between the different users. We are trying to find out if this connectivity is a function of the label or not. If we find a correlation between the connectivity between individuals and their label it could help us to build our predictive model later on. This is why we will display the labels individually as well as those of the test set.

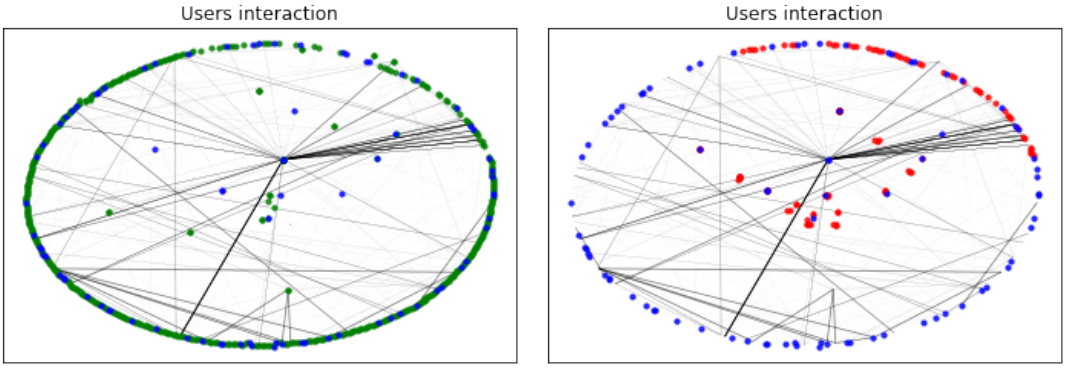


Figure 2: Relationship between unknown users and other users

We can see that there is connectivity within the labels, and that there are very similar behaviors. We will then try to determine the common characteristics of each subgroup according to both their label and their behaviour. Indeed, some users are the link between the two labels.
We tried to develop a tool to convert a covariance matrix into a model graph. This will facilitate interpretations but also allow us to confirm or invalidate hypotheses as to which features should be included or excluded.

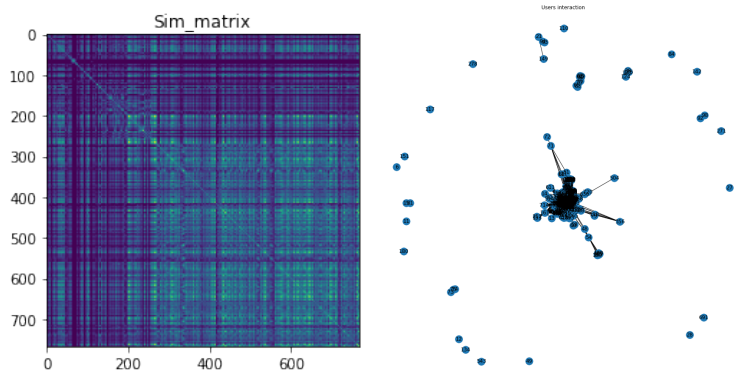


Figure 3: Graph depending on the weight

We can see that the conversion between the sim_matrix and the graph is well done, and that there are isolated users who do not interact with any other. These isolated users are more

difficult to label because one cannot compare the behavior of their neighborhood in order to label them. We can rightly assume this matrix has removed too many features that could have connected these users to the rest of the network.

2.2 Compare the predictive power of the features

In this part our goal is to retain only the features that are the most relevant in the sense that allow us to best discriminate users according to their label.
At first we do a first manual sorting within features. We remove all the features that are equivalent, especially those in terms of time: we have features in hour, minute, second, and their equivalent in seconds. This already allows us to remove 4 columns. Features with the same value for all users are also removed without distinction, especially for the average, median or minimum values of certain features. These are not relevant because they are non-discriminatory for our spammer detection. In addition, we also sort the features that are not present in our test set.
We made the arbitrary choice to remove other features because in our opinion they introduced a bias in particular for the date of the oldest and most recent tweet because this date depends on the dataset, for a more recent deployment this can be problematic, the same goes for the language or geographical characteristics. It is assumed that there is an invariance in time and space. This is due to the fact that the amount of spammer in our set is too small and there is a very high risk of bias. To Only then can we build the covariance matrix allowing us to select the least correlated and most relevant features.

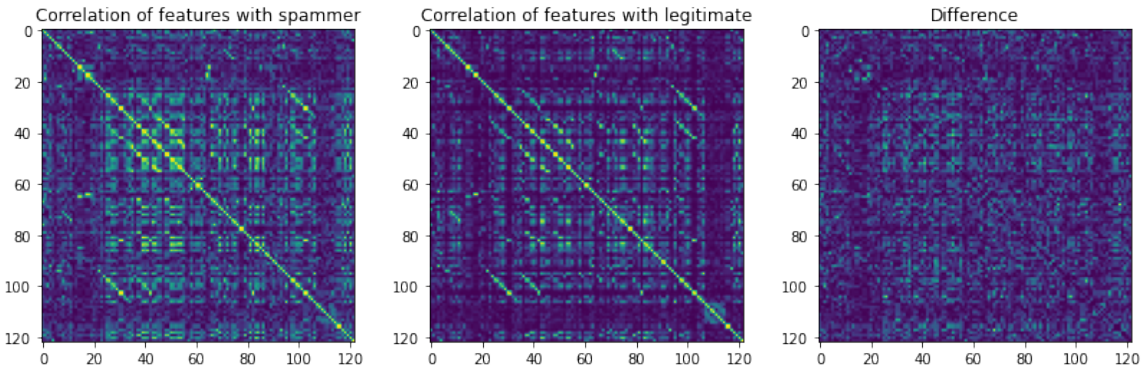


Figure 4: Correlation of features according to subspaces

Looking at the sub-space of the labels, we can see that the covariance matrices are not identical. We can then assume that there are unique characteristics for each label. Our goal is then to decrease the size of this covariance matrix in order to select the relevant features.
By seeking to automate the feature selection process so as not to do it manually, we seek to remove the most correlated features. A threshold value is introduced to retrieve only the least correlated features. We then have for a fixed threshold the following sub-space:

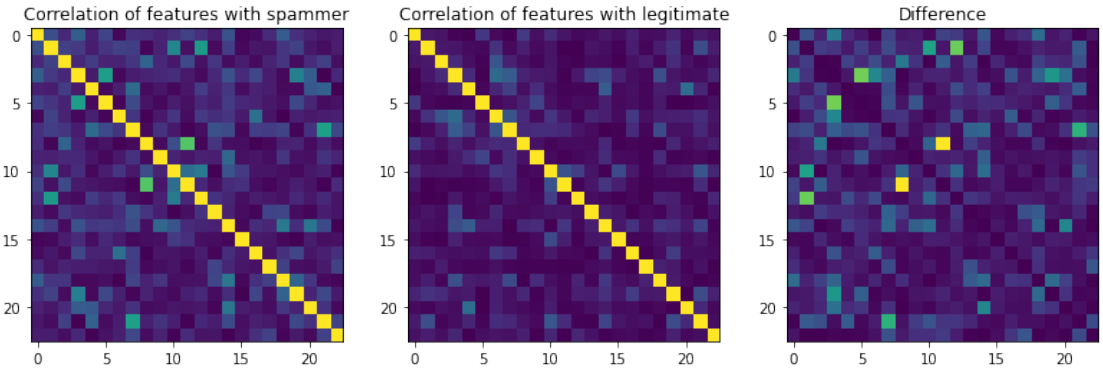


Figure 5: Correlation of features according to subspaces

For a threshold of 0.3 in our processing function, we have a difference between the two subspaces that is more important. We can reduce the number of features by lowering the threshold. This will be particularly useful in the pre-processing phase for the machine learning algorithm. In the present case, only 23 features are retained, i.e. less than 15% of the initial features.

Remark: our method of feature reduction is similar to PCA in that we go through the covariance matrix again. However, we have not chosen PCA and we do not want to project the initial feature space into another workspace. This would complicate the addition of new features in particular. This is like selecting the largest singular values without going through the more expensive matrix calculation.

2.3 Applications mainly used by abusive users

In this part we make an in-depth analysis of the relationship between the different types of users and the applications they use. At first we wanted to have a graphical representation of the data we have at our disposal. In order to do so we use the data from the applications used by each normalized user. We get the following figure on the left.

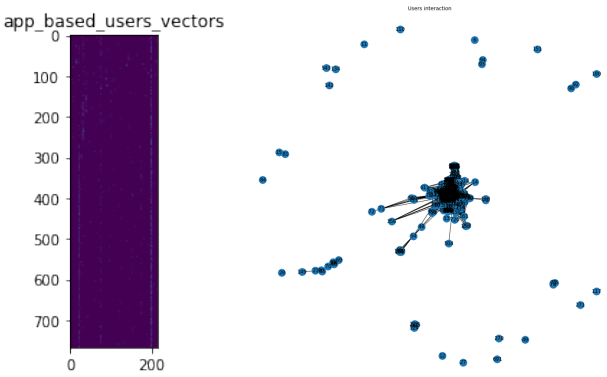


Figure 6: Distribution of the applications used by whole the user

The previous figure on the right is the translation of this one in the form of a graph representing the connection between users according to the applications they use. This information is difficult to interpret as it is, so we will subdivide the space according to the labels we have been given. We obtain by a similar treatment the following figures.

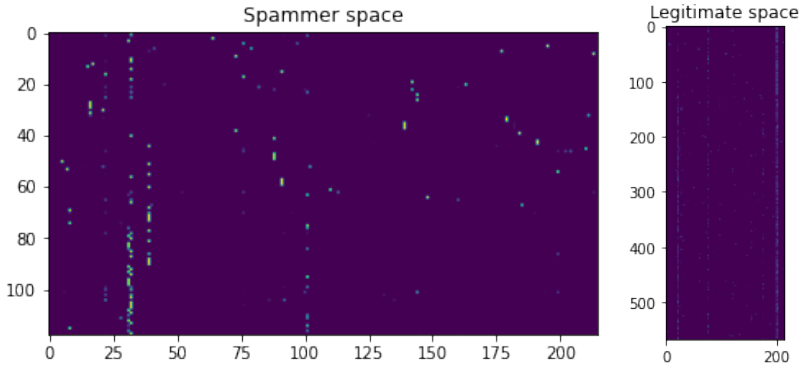


Figure 7: Subdivision of the space according to the two labels

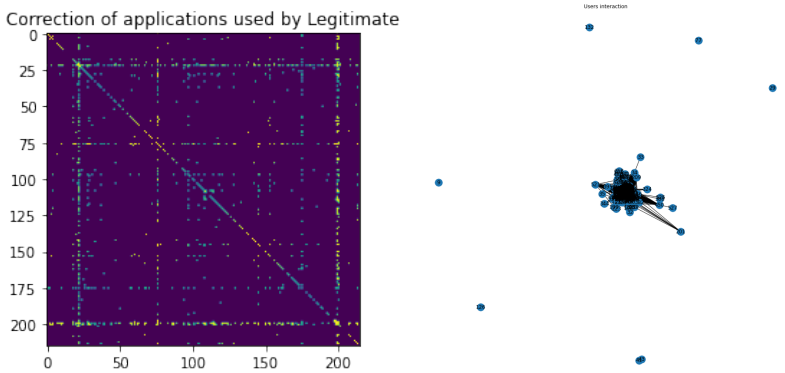


Figure 8: Graph of applications used by legitimate

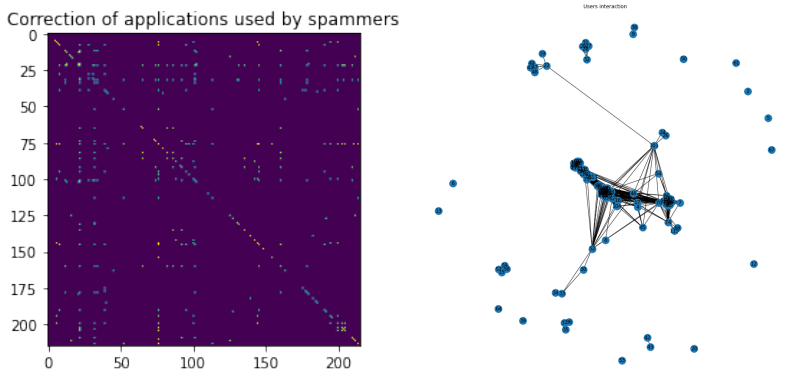


Figure 9: Graph of applications used by spammer

At first glance, the correlation between the applications used jointly by the two groups seems to be distinct. This is visible by the fact that there are no obvious superposition of the two correlation figures. Furthermore, it is important to note that there seem to be different spammer profiles with key users connecting the different groups. One should also pay special attention to users with zero connectivity in this case. They are more prevalent in the class of spammers than in the class of legitimates.

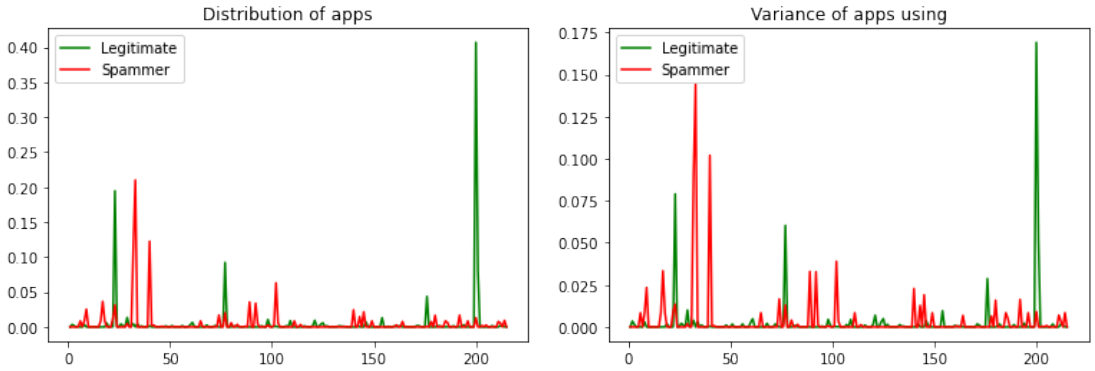


Figure 10: Distribution of the applications used according to the label

The in-depth study through the establishment of a histogram confirms our intuition. As a precautionary measure, we also studied the variance in order to know if the diversity of applications used is important. This allows us to determine confidence thresholds and determine the bias in the event that we rely solely on a statistical study of this criterion to determine whether a person is in the spammer category or not. The following is a list of the most commonly used applications within the two groups.
For legitimate group we have:

- 199: Twitter for iPhone with 40.67%
- 22: Twitter Web Client with 19.46%
- 76: Twitter for Android with 9.22%
- 200: TweetDeck with 8.04%

For spammer groupe we have:

- 32: Done For You Traffic with 20.99%
- 39: IFTTT with 12.24%
- 31: dlvr.it with 11.37%
- 101: Google with 6.29%

We classify by decreasing frequencies the applications used in the group of spammers.

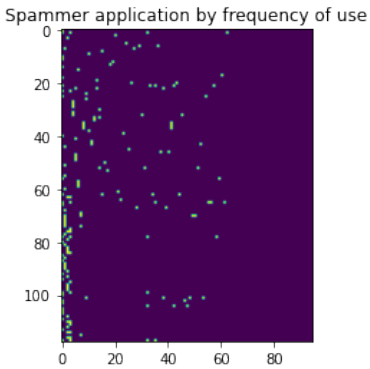


Figure 11: Reduced space of applications used by spammers

We decided to investigate further in order to determine the most relevant applications to be retained in the case of spam detection by this means. So we are doing a correlation study between the applications and we will select the most relevant ones.

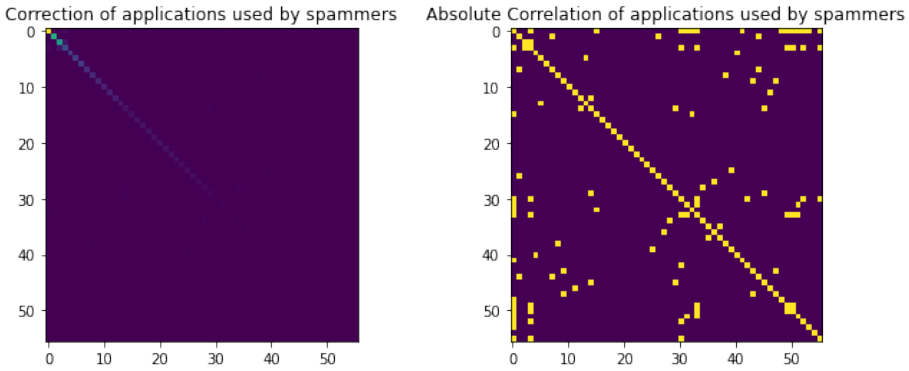


Figure 12: Correlation of applications used by spammers

It can be seen that there is little correlation between the applications. This can be explained by the fact that the spammers present in our databases are from click providers. Thus, by targeting these in priority, we have been able to find a large majority of them. This explains that: we can see that by using a smaller list, here at least 40%, of applications associated with spammers we obtain a more readable graph without significant loss of data.

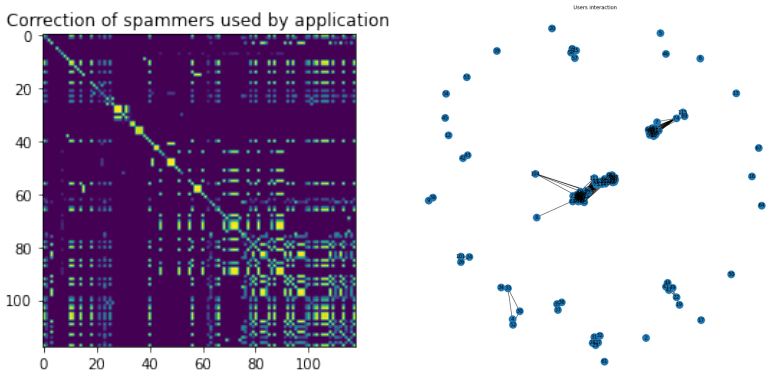


Figure 13: Reduced graph of applications used by spammer

Note: In this study, we have chosen to recalculate the equivalent of the similarity matrix provided in the dataset. We used only the raw data present in the files of the app_based_similarity path.

2.4 Identify groups of users or communities and characterize them

Our goal here is to identify the different users groups and their communities as a result of the treatments we have done including feature reduction. To do this, we have to take a particular care to check that we can discriminate according to labels, or at least to a large extent, and that we no longer use the data we were provided to build the previous graphs. We calculate covariance matrices between users according to their sub-space in order to identify different groups.

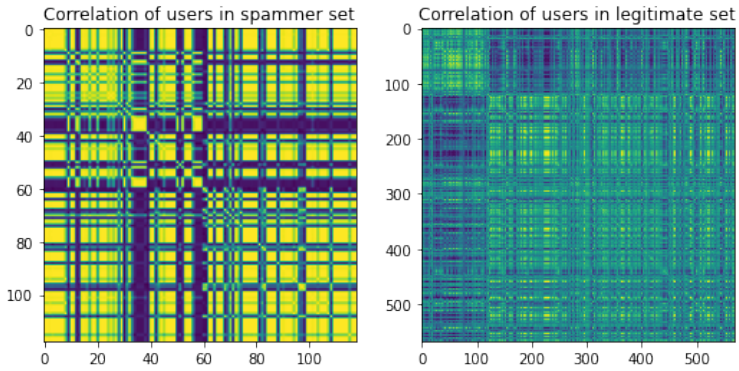
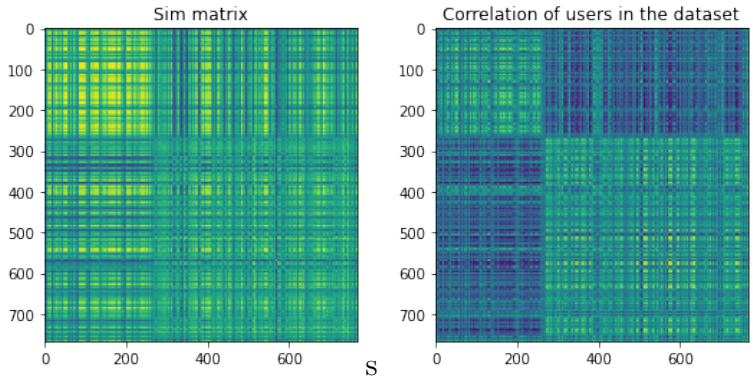


Figure 14: Subspace sim matrix

We can see that there is a high degree of homogeneity between spammers. This is visible by very high correlations between users. We can see that there is a majority group with minority subgroups, at least 3 (left figure). As for the legitimates, we have a diversity of profile which is greater we can count two majority groups, and multiple minority sub-groups (right figure). Finally to determine their characteristic it is interesting to isolate in sub-space the individuals of the groups to identify and to extract the common characteristics they possess.



S

Figure 15: Sim matrix

We want to display the similarity of all users of the dataset. It is qualitatively observed that the spammer are mainly absorbed by the secondary subgroup of legitimates. We can then assume that this second largest group of legitimate labels may be a source of error when detecting spammers because they have similar behaviors.

3 Prediction Task

In this part our goal is to test different machine learning tools in order to have the best detection between spammers and legitimates. To do this we will experiment with several learning machine algorithms including KNN and SVM. We must first do a pre-processing part to clean up the data, which are essentially based on the tools we developed in the previous part.

3.1 Approach

In a first step we have to take care of the pre-processing. We can see from the study carried out in the previous section that there is a great imbalance between the two labels of interest. One solution to counter this phenomenon is to use a Synthetic Minority Oversampling Technique (SMOTE). This is data increase and allows to artificially rebalance the two labels.

We decided to work on two machine learning algorithms to predict labels with KNN and SVM. By trying several algorithm settings we will then make a soft majority vote in order to have a result from the prediction of our different algorithms. We use the F1 score to evaluate the performance of the algorithm. We subdivide the training dataset into two sets, one for validation and one for testing (10% of the global set). It is important to note that we will recalculate the F1 score on the original unbalanced set for each label in order to evaluate if we have a strong deviation of the score related to the SMOTE or the projection.
At the end of its different trials, a soft majority vote is set up to predict the results. The soft way is used to weight the contribution of each model according to its performance; the better a model is performing and the more important its contribution is, but also according to the type of algorithm chosen. In addition, the choice is made to set the points manually. Moreover, since spammers are the minority label but of interest, we also compare the score to a stupid prediction which is to consider that no matter what the entry is, we always have a legitimate prediction that gives us an idea of the minimum performance threshold that our algorithm must have.

3.1.1 KNN

The choice of the k-nearest neighbors algorithm at first glance is the most relevant for this problem because, as seen in the previous section, we have within the different groups similar behaviors. This means that a user behaves very similarly to his neighbors. So, selecting the right features, we just have to look for the closest neighbors to predict his label. It remains to determine the best possible k parameter, for this we can use cross-validation.

3.1.2 SVM

We currently have a non-linear problem, however by using kernel trick we hope to be able to project the space of our features in a dimension where we have a linear separation according to labels. This would then allow us to use a SVM algorithm to characterize individuals. The difficulties here are more important, we have to find the best kernel to project the data in a linearly separable space to try the T-SNE algorithm. In addition, we do not know the behavior of the data generated by the SMOTE during a projection. This is why we have made two ML algorithms, one with the smote before the projection, and the other after.

3.2 Results

We have empirically determined that keeping the 23 least correlated features was the best setting, we can go down to 6, but beyond 23 we have an underfitting problem. An in-depth study should be done on a robust model. For 23 features kept on the initial set at the end of the pre-processing step we have:

3.2.1 KNN

We selected k=2 arbitrarily, a cross validation would allow us to select the best hyperparameter. We potentially had to take the average over several trainings in order to minimize the impact of the set composition. For k=2 we have:

Without SMOTE

KNN	0.95
Stupid model always predict	0.91
Correct prediction about original spammer	0.76
Correct prediction about original legitimate	0.86

With SMOTE

KNN	0.93
Stupid model always predict 1	0.65
Correct prediction about original spammer	0.95
Correct prediction about original legitimate	0.91

Result with features increases

KNN	0.92
Stupid model always predict 1	0.69
Correct prediction about original spammer	0.96
Correct prediction about original legitimate	0.89

3.2.2 SVM

For a gaussian kernel we have: **Without prepossessing**

SVM	0.66
Stupid model always predict 1	0.63
Correct prediction about original spammer	0.34
Correct prediction about original legitimate	0.94

With prepossessing

SVM	0.64
Stupid model always predict 1	0.65
Correct prediction about original spammer	0.49
Correct prediction about original legitimate	0.96

T-SNE
For T-SNE with 2 components and SMOTE after the projection we have:

SVM	0.84
Stupid model always predict 1	0.60
Correct prediction about original spammer	0.00
Correct prediction about original legitimate	0.79

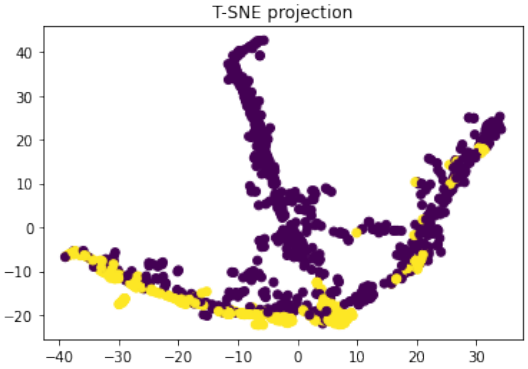


Figure 16: Projection space

For T-SNE with 2 components and SMOTE before the projection we have:

SVM	0.84
Stupid model always predict 1	0.59
Correct prediction about original spammer	0.00
Correct prediction about original legitimate	0.77

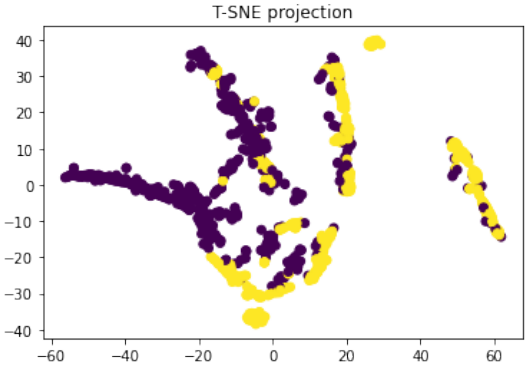


Figure 17: Projection space

Majority vote
A majority vote is taken in order to have the best possible prediction because we consider that different models can capture different information and complement each other. However, we choose to make a soft majority vote because the accuracy of the models is very disparate. The following contributions are selected manually:

Model	Coef	Accuracy on spammer	Accuracy on legitimate
KNN	1/3	0.95	0.91
KNN + features augmentation	1/3	0.96	0.89
SVM	1/6	0.95	0.49
SVM + T-SNE + SMOTE	1/12	0	0.79
SVM + SMOTE + T-SNE	1/12	0	0.77

3.3 Discuss

It is interesting to note that in our KNN it is more efficient when you make a SMOTE that allows us to compensate the imbalance of the data. We also can observe with an increase in

features a better detection of spammers, but less detection of legitimates. We could counter this by adding features characteristic of legitimates but this is not in the spirit of what we want to do: detect spammers. We assume that it is better to have false positives than to forget them. Moreover, as we can be seen the KNN provides the best result. We have correct results with SVM, but very mixed results for projection with T-SNE. On the other hand, there is a difficulty in predicting spammers compared to legitimate. This is due to the fact that we have an unbalanced dataset. Despite the use of SMOTE, this does not allow to generate authentic samples and explains the discrepancies. We can improve the SVM[6] if we improve the projection kernel, we should deepen the investigations.

4 Conclusion

Our main goal is to detect spammer on Twitter. It is assumed that it is preferable to have 100% of the spammers with false positives in any given proportion. Based on this premise, our goal is to visualize the data in order to make them understandable and to be able to better characterize the different categories of users and the relationships between them. We deduce that there are two majority groups of legitimate users, and that the second majority group in terms of size has a very similar behavior with that of spammers. This can be very problematic concerning the detection of these because it introduces a behavioral bias. One solution to counter this is to add features about the applications used by spammers because they are more discriminating than just the user profile. The group of spammers is relatively homogeneous but we can't be sure about that because of the low number of users of this label.
In addition, it is usual that one is a less important set of data on spammers than legitimates because it is an undesirable class and therefore under-represented. To counter this we used the SMOTE. This significantly improved the prediction performance of our best algorithm. And before training this one, it is common practice to cleanse the data of irrelevant information in order to improve the performance of the algorithm both in terms of execution time and accuracy. To do so, we have essentially studied the feature covariance matrices in order to keep the least correlated and therefore the most divergent features.
By training different models of machine learning we determine that the best model is the KNN. This one allows to have the best detection and clustering score. However, we decide to make a soft majority vote in order to increase the performance of global predictions as we will weight the contribution of each algorithm according to its average performance.

Some aspects of our work can be improved in particular the data visualization where we could derive more information on the characterization of the different profiles by making a study of the characteristic features of each cluster of individuals. This is similar to what we have been able to do for application data. We could also have improved the representation of our graphs by making a more specific study of the centrality nodes or bridging centrality nodes. This would have allowed us to provide us with other features increases. In the case of a longer project we should make more precise studies of the hyperparameters of our algorithms by cross-validation. We would also focus on finding the best kernel and projection space to improve SVM results. However, it seems that it is more difficult to determine if we have a large number of features. It is not easy to reduce them without loss of diversity to differentiate the two labels.

References

[1] Shobeir Fakhraei et al. “Collective Spammer Detection in Evolving Multi-Relational Social Networks”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’15. Sydney, NSW, Australia: Association for Computing Machinery, 2015, pp. 1769–1778. ISBN: 9781450336642. DOI: 10.1145/2783258.2788606. URL: https://doi.org/10.1145/2783258.2788606.

[2] Maksym Gabielkov, Ashwin Rao, and Arnaud Legout. “Studying Social Networks at Scale: Macroscopic Anatomy of the Twitter Social Graph”. In: *CoRR* abs/1404.1355 (2014). arXiv: 1404.1355. URL: http://arxiv.org/abs/1404.1355.

[3] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[4] Ninghao Liu and Xia Hu. “Spam Detection on Social Networks”. In: *Encyclopedia of Social Network Analysis and Mining*. Ed. by Reda Alhajj and Jon Rokne. New York, NY: Springer New York, 2018, pp. 2851–2859. ISBN: 978-1-4939-7131-2. DOI: 10.1007/978-1-4939-7131-2_110199. URL: https://doi.org/10.1007/978-1-4939-7131-2_110199.

[5] Wes McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[6] Sumaiya Pathan and R. H. Goudar. “Detection of Spam Messages in Social Networks based on SVM”. In: *International Journal of Computer Applications* 145.10 (July 2016), pp. 34–38. ISSN: 0975-8887. DOI: 10.5120/ijca2016910793. URL: http://www.ijcaonline.org/archives/volume145/number10/25317-2016910793.

[7] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[8] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[9] Junting Ye and Leman Akoglu. “Discovering Opinion Spammer Groups by Network Footprints”. In: *Proceedings of the 2015 ACM on Conference on Online Social Networks*. COSN ’15. Palo Alto, California, USA: Association for Computing Machinery, 2015, p. 97. ISBN: 9781450339513. DOI: 10.1145/2817946.2820606. URL: https://doi.org/10.1145/2817946.2820606.