

**ENCS3340**

**Machine Learning**

**Tweets Spam Detection**

**Prepared by**

**Abrar Abdul-Samad 1171481**

**Layth Bazzar 1180612**

**Osama Rihami 1190560**

Table of Contents

[Abstract 3](#_Toc96192677)

[Spam Detection 4](#_Toc96192678)

[1) Unsupervised collective-base d framework for dynamic retraining of supervised real-time spam tweets detection model. Mahdi Washha, Aziz Qaroush, Manel Mezghani, Florence Sedes 4](#_Toc96192679)

[2) Kabakus, A. T., & Kara, R. (2017). A survey of spam detection methods on twitter. International Journal of Advanced Computer Science and Applications 5](#_Toc96192680)

[3) Wu, T., Wen, S., Xiang, Y., & Zhou, W. (2017). Twitter spam detection: Survey of new approaches and comparative study. Computers and Security 11](#_Toc96192681)

[Used features for detecting Spam 12](#_Toc96192682)

[Code and Design 13](#_Toc96192683)

[Simulation And Comparison 18](#_Toc96192684)

[References 20](#_Toc96192688)

[Appendix 21](#_Toc96192689)

# Abstract

Unwanted and dangerous tweets, such as some commercial tweets, click baits and other tweets that may threat the mental well-being of the user, are all considered as Spam tweets. These kinds of tweets must be detected and marked as Spam (and sometimes deleted) before it is delivered to the other users, avoiding any kind of damage or other effects.

Such tweets can be detected from some information that derived from the tweet itself or from the user. The tweet body, the number of actions on this tweet and the tweet being a re-tweet or not are all information derived from the tweet that can help determine if this tweet is a Spam or not. The user information that can also help us are the numbers of followers and people how the user follows and the location given by the user, note that this location is not always the real location of the user.

Machine learning algorithms help us detect Spam tweets without human interference. Such algorithms are Naïve Bayes, Neuron networks and decision trees. All these algorithms need some kind of parameters called features that are taken from the previously mentioned information to detect entered by the programmer, then these algorithms will decide how can each of these features will decide if the tweet is Spam.

# Spam Detection

Detection spam using machine learning is dependent on features that are derived from information about the tweet. There are a lot of features that can be taken in mind when designing a spam detector. The features that were included in this Spam detector were mostly based on these papers which states:

## 1) Unsupervised collective-base d framework for dynamic retraining of supervised real-time spam tweets detection model. Mahdi Washha, Aziz Qaroush, Manel Mezghani, Florence Sedes

Because spammers are becoming smarter as a result of complicated spamming methods, the statistical aspects of spam tweets are changing with time, making the current machine learning based detection method ineffective. The authors present a system for dynamic retraining of supervised real-time tweet-level spam detection models to reduce the effect of the spam drift problem. The proposed framework is made up of two primary modules, the first of which operates in batch mode and uses the strength of the unsupervised learning method to deliver up-to-date annotated datasets on a regular basis. The first module derived knowledge from unlabeled tweets through studying and analyzing the collective perspective of streamed tweets and their users' behavior. The second module includes a real-time tweet-level classification model trained on 17 lightweight features and retrained on a regular basis using the first module's up-to-date annotated dataset. On the authors’ gathered dataset, which contains over 2 million tweets annotated by the suspended account-based method, the authors tested their framework and two other related methods. The results reveal that simply increasing the size of the training data will not improve the spam classification model any more. Furthermore, as compared to conventional and asymmetric classification approaches, the authors’’ methodology has a superior and controllable spam recall performance, which has a significant impact on reducing spam drift.

The proposed framework provides an online unsupervised learning method that does not require a human intervention in the way of periodically preparing annotated training datasets, which is a significant difference from other real-time spam detection methods proposed in the literature for dealing with the spam drift problem. Furthermore, this method provides a lightweight tweet-level spam detector that works in real-time and updates itself periodically in batch mode, requiring no pre-information such as a blacklist of spamming sites, an initial annotated dataset, or a pre-trained classifier. Furthermore, because of the high recall values, this method can be used by Twitter-based researchers and industries to stream only high-quality tweets, which are required by a variety of intelligent tweet-based applications like sentiment analysis.

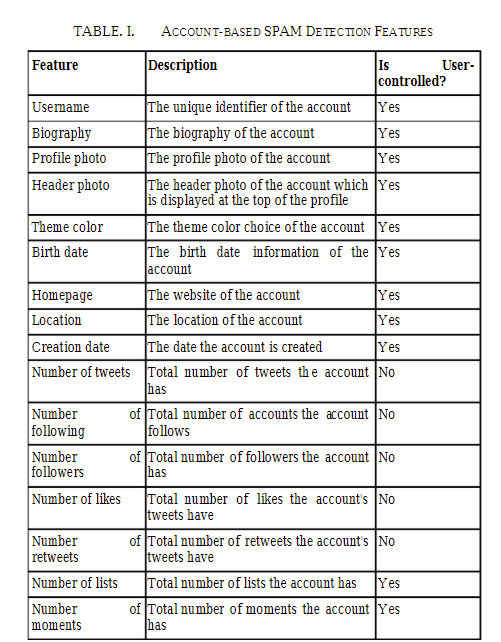
In addition, the proposed framework has a flaw. The system included no provision for the expansion of the obtained training dataset. In fact, by including fresh spamming actions, the goal of expanding the size of the training data is to remove the influence of spam drift. In the long run, however, the contribution of extremely old spam contents will decrease as the correlation between older contents and new spam contents decreases. As a result, without effective growth management, the classifier will not be able to adapt fast to new spamming behaviors. Furthermore, classifiers will demand longer training time. Furthermore, it has the potential to amplify the negative effects of the class imbalance dataset. The authors should focus on lowering the size of the acquired data in the future by removing very old tweets, particularly non-spam tweets, after a specified period of time. Another restriction is the annotation approach employed in the gathered dataset, which may include non-spam tweets from suspended accounts. However, there is no publicly available dataset or common evaluation mechanism suited for testing this framework, and manually classifying a huge number of tweets takes a lot of time and effort.

## 2) Kabakus, A. T., & Kara, R. (2017). A survey of spam detection methods on twitter. International Journal of Advanced Computer Science and Applications

The features of Twitter spam detection are divided into three categories: (1) account-based features, (2) tweet-based features, and (3) the sender-receiver relationship. These features are the mainframes of the characteristics utilized in connected literary works. The following paragraphs go through each feature category.

1. Account-based Features

Spammers can be identified by looking at their Twitter accounts, which have the characteristics listed in Table 1. Some of these features, such as biography, location, homepage, and creation date, are useless in terms of spam detection because they are user-controlled.



When the behaviors of spammers are analyzed within the scope of account-based features, these facts are observed:

Since spammers tend to follow too many legitimate accounts in order to attract attention, the number of following is expected to be high compared to legitimate users.

Since spammers are not followed by legitimate users, the number of followers is expected to be less compared to legitimate users.

Since spammers' tweets are unsolicited, the number of likes and retweets for their tweets are expected to be less compared to legitimate users.

Since spammers tend to post lots of tweets to attract the attention of legitimate users, the number of tweets sent by the account is expected to be high compared to legitimate users.

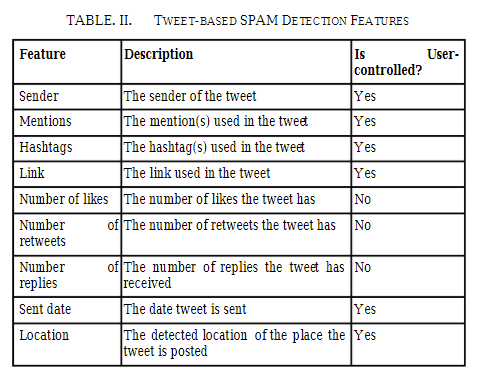
Spammers’ tweets mostly contain links and Hashtags to attract the attention of legitimate users. Since spammers' tweets are ignored by legitimate users, the number of replies and mentions spammers get are expected to be low compared to legitimate users.

Spammers tend to post same or similar tweets which are posted by one or more controlled accounts.

Legitimate users tend to be added to the lists unlike spammers unless bots under the command and control (C&C) architecture add them to the lists they intentionally created in order to manipulate spam detection approaches.

1. Tweet-based Features

Spammers tend to post lots of unsolicited tweets to legitimate users to attract attention. Spammers can be detected by analyzing their tweets. This is necessary to filter spam tweets from legitimate ones and provide users a spam-free environment which is the aim of Twitter [60]. Each tweet contains the information listed in Table 2



When the behaviors of spammers are analyzed within the scope of tweet-based features, these facts are observed:

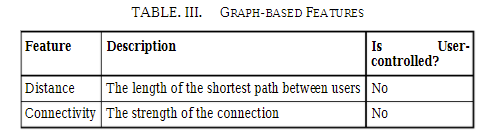
Spammers tend to use links to direct legitimate users to their malicious purposes.

Spammers tend to use lots of mentions to attract the attention of more legitimate users. Spammers tend to use lots of Hashtags (especially the trending ones) to reach more users.

Since spammers' tweets are unsolicited, the number of likes and retweets their tweets have received are much lower compared to legitimate users.

1. Graph-based Features

Twitter is a network of users with relationships between them and tweets. This structure can be represented as a graph. For the graph model, users and tweet can be represented as nodes and relationships can be represented links between nodes. These relationships show how the tweet's sender and mentions are connected to each other. Also, these relationships are clear indicators of legitimate conversations. By constructing a graph model to represent users and their relationships, the distance between the tweet's sender and mentions can be calculated for spam analysis. Graph-based features are listed in Table 3.



When the behaviors of spammers are analyzed within the scope of graph-based features, these facts are observed:

The distance between a spammer and a legitimate user is further than the distance between two legitimate users.

The connectivity between a spammer and a legitimate user is more robust than the connectivity between two legitimate users.

Graph-based features provide the most robust performance to detect spam and spammers since they are hard to manipulate and not user-controlled.

TWITTER SPAM DETECTION METHODS

The approaches for detecting Twitter spam in the literature are described and explored in this section. The proposed methods are divided into the following categories: Methods include (1) account-based spam detection, (2) tweet-based spam detection, (3) graph-based spam detection, and (4) hybrid spam detection.

1. Account-based Spam Detection Methods

Account-based spam detection methods are based on the features (or a combination of them) of Twitter account which are listed in Table 1.The average number of tweets per day, the ratio of following to followers, the percentage of bi-directional friends, the ratio of the number of URLs in the 20 most recently posted tweets, the ratio of the number of unique URLs in the 20 most recently posted tweets, the ratio of the number of usernames in the 20 most recently posted tweets, and the ratio of the number of Hashtags in the 20 most recently posted tweets are all features they consider when detecting spam. Lin and Huang present a method for detecting spam in Twitter based on two characteristics: (1) the URL rate, which is the number of tweets having a URL divided by the total number of tweets, and (2) the interaction rate, which is the number of tweets engaging divided by the total number of tweets. Gee and Hakson suggest a strategy based on account-based characteristics including the followers-to-following ratio, the number of tweets to account lifetime ratio, the average time between postings, posting time fluctuation, maximum inactive hours, and link fraction. The work's weakness is that they use the manual method of reporting spam on Twitter, which is antiquated as previously noted. Many Twitter spam detection approaches combine account-based features with other spam detection features to provide more powerful spam detection methods, which are referred to in this paper as "hybrid" spam detection methods.

1. Tweet-based Spam Detection Methods

Tweet-based spam detection algorithms are based on the properties (or combinations of them) given in Table 2 of a tweet. Static or dynamic crawlers are used in URL filtering algorithms to explore newly discovered URLs. They also use URL or domain blacklisting to identify questionable URLs in a knowledge base. These methods make use of URL and DNS information, URL redirections, and the source code of the destination website (HTML). McGrath and Gupta describe a phishing detection method based on a URL's lexical properties. The length of the URL and the domain name, the character composition of the domain name, the inclusion of brands in URLs, and the overuse of URL-aliasing and free web hosting services are among the criteria they check when identifying phishing. Ma et al. offer a method for detecting fraudulent websites based on URL analysis. WHOIS properties such as who is the registrar of the website, who is the registrant of the website, when the website is registered, domain name properties such as the time-to-live (TTL) value for DNS records, and geographic properties such as which country the IP address belongs to, the speed of the uplink connection, and lexical features of URL are among the features they use to detect malicious websites. Prophiler is a filter that detects harmful information on a website using static analysis techniques. The characteristics Prophiler considers are derived from (1) the website's HTML content, such as the number of elements with small areas, the number of elements with suspicious content, the number of included URLs, and the number of known malicious patterns, and (2) the associated JavaScript code, such as the keywords-to-words ratio, the number of long strings containing decoding routines, the probability of shellcode presence, and the number of known malicious patterns. Since Prophiler relies on static analysis techniques, it is unable to detect harmful URLs included in dynamic content such as JavaScript, the most extensively used programming language, Flash, and Java applets. For in-depth content analysis, methods based on dynamic analysis techniques use virtual machines and automated web browsers like Selenium. A phishing detection approach based on URL analysis was published by Chhabra et al. As previously said, their technology is specifically built to evaluate abbreviated URLs, which are widely used in Twitter to create spam messages. The features the proposed method use detecting phishing through an URL are the number of clicks, geographical spread, temporal spread, and web popularity. WarningBird is a suspicious URL detection system for Twitter which investigates correlations of URL redirect chains. WarningBird uses 14 features to detect suspicious URL such as the length of URL redirect, the number of different landing URLs, the relative number of different Twitter accounts, the similarity in the account creation dates, the similarity in the number of followers and following, the similarity in the follower-following ratio, and the similarity of tweets. Martinez Moro Ajauro offers a tweet-based spam detection system that focuses on linguistic analysis in tweets. They employ (1) the language model of tweets relevant to a trending issue, (2) the language model of the tweet, and (3) the language model of the page linked by the tweet as language models. Many Twitter spam detection approaches, similar to account-based spam detection methods, combine tweet-based features with other spam detection features to provide more comprehensive spam detection.

1. Graph-based Spam Detection Methods

Graph-based spam detection methods are based on the features (or combinations of them) of a tweet which are listed in Table 2. Song et al. extract the distance and connectivity between the tweet's sender and mentions. While the length of the shortest path between the tweet's sender and mentions is defined by distance, the strength of the link between users is defined by connection. Graph-based spam detection approaches mimic Twitter properties such as nodes and edges using graph data structures. Graph data models are ideal for representing data in which information about data interconnection or topology is just as important as the data itself. As a result, social networks like Facebook and Twitter, which are based on people, topics, and bi-directional interactions, frequently use graphs. Despite the fact that graph-based characteristics have the best accuracy and sensitivity for distinguishing spammers from genuine users, additional graph-based spam detection approaches are included in hybrid spam detection methods because they are paired with other spam detection methods.

1. Hybrid Spam Detection Methods

Hybrid spam detection methods combine the spam detection methods discussed in the preceding subsections to give more powerful spam detection that explores the likelihood of spam in a more comprehensive way. Stringing et al. proposed an approach based on both account-based and tweet-based features, including the ratio of the number of friend requests sent by the user to the number of friends she has, the ratio of the number of tweets containing URLs to the total number of tweets the user has, the similarity of tweets sent by the user, the number of tweets sent by the user, the number of friends the user has, and the possibility of whether an account likely used a list of friends. Gao et al. proposed a tweet-based spam detection approach based on the tweet's sender's social degree, history of interaction, cluster size, average time interval, average number of URLs in tweets, and unique number of URLs in tweets. Chen et al. presented a Twitter real-time spam detection approach based on 12 lightweight features collected from a dataset of 6.5 million spam tweets. The age of the account, the number of followers, the number of following, the number of likes the account received, the number of the account's lists, the number of tweets, the number of retweets of the tweet, the number of Hashtags used in the tweet, the number of mentioned users in the tweet, the number of URLs used in the tweet, the number of characters used in the tweet, and the number of characters used in the tweet are the features they consider when detecting spam on Twitter. Wang proposes a hybrid Twitter spam detection approach that combines graph and tweet-based features. The number of followers, the number of following, a reputation score computed as the ratio of the number of followers over the total sum of the number of followers and following, and the number of following are the graph-based features evaluated in the proposed method.

## 3) Wu, T., Wen, S., Xiang, Y., & Zhou, W. (2017). Twitter spam detection: Survey of new approaches and comparative study. Computers and Security

To address the feature selection issue, this research developed a unique methodology for recognizing twitter spam data using machine learning methods. Pre-processing, feature selection, and classification are the three aspects of the suggested approaches. The results of the authors’ experiments on a variety of multilevel datasets showed that their unique framework could effectively find the appropriate feature subset and reduce greater dataset size errors.

Furthermore, the author evaluated the value of feature selection results to known methods of twitter spam detection, such as Nave Bayes, SVM, and KNN. In terms of classification, the results of this research reveal that real-time running classification jobs can distinguish between spam and non-spam. The greatest of all performance accuracy and running time appears to be a revolutionary framework method. Existing methods revealed a long time for training data, overlapping, and inferior performance measures, as well as the removal of some features during the selection period.

This research's contributions include: reducing dimension from tweet datasets, identifying tweet spam in real-time, increasing detection speed, selecting the optimal features for learning models, and improving efficiency accuracy jobs.

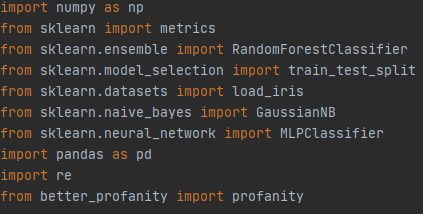
## Used features for detecting Spam

In this Spam detector, several features were taken in account when designing. Most of these features were based on the papers mentioned above, and some of them were based on our groups’ observation and opinions. Here is a summary of these features:

1. Number of URLs: the number of links and short links found in the tweet, which will be compared to the number of words.
2. Number of Hashtags : the number of words that start with the character ‘#’ that are found in the tweet, which will be compared to the number of words.
3. Number of mentions: the number of words that start with the character ‘@’ that are found in the tweet, which will be compared to the number of words.
4. Number of words: the number of words in the tweet separated by a space character. This will be used in comparing the number of URLs, Hashtags and mentions.
5. Number of characters: it is the number of every character, spaces, numbers and special characters included
6. Rate between followers and people followed by the user: if the rate between these two is too big, it means this is a Spam account.
7. Missing attributes of the followings, the followers the “is a re-tweet” and actions in the excel file. This feature was based on our observations which helped raising the accuracy significantly. Each attribute is a feature by itself. The value of the features of each attribute is zero if missing, one otherwise.
8. Number of profanity words: this feature too was based on our observation, which will count the number of words that is a profanity word. Adding this observation resulted to a slight improvement in the Random Forest Decision Tree and Neuron Networks algorithms, but a considerable decline in the Naïve Bayes algorithm. It also resulted to slowing down the detector considerably. So it was removed after.

It is right to mention that too much features might cause an over fitting like what happened with the number of profanity words feature in the Naïve Bayes classifier.

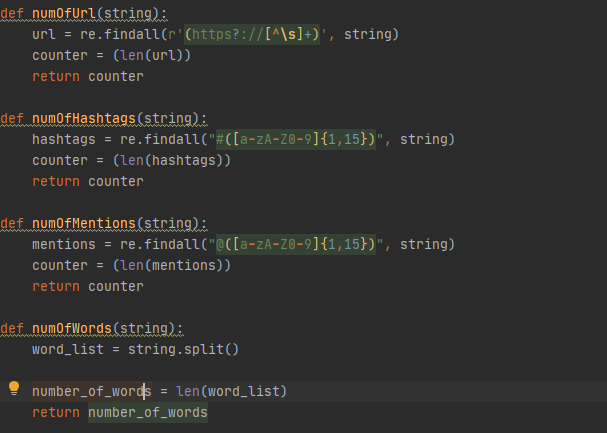
# Code and Design



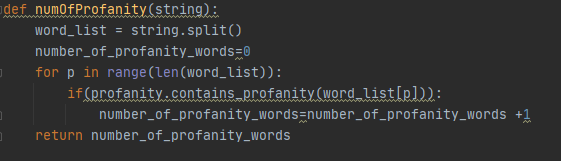
This part of the code is the imported APIs that were used in designing the detector. Numpy was used in array management, Sklearn has the classifiers used in the detector in it, Pandas was used for reading from an excel file, Re for regex functions and Better\_profanity was used in detecting profanity words.



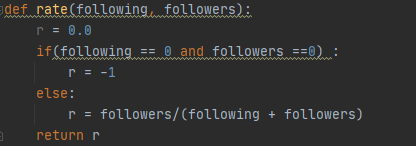
This part is responsible for reading from an excel file and storing the data in X array and Y array. X array has data about the tweets and Y array determines whether the tweet is a Spam or not.



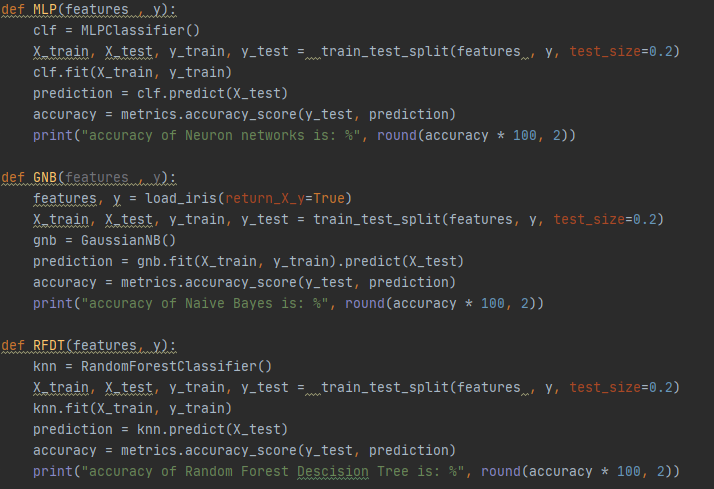
This part calculates some of the features, which are the number of UCLs, Hashtags, Mentions and words in the tweet. These calculations were mostly done with the help of the Re API.



This part of the code was done using the Better\_profanity API. It is used to calculate the number of profanity words n the tweet. Due to the API used, this part of the code takes a lot of time, even when this API is one of the fastest APIs to detect profanity on the internet.

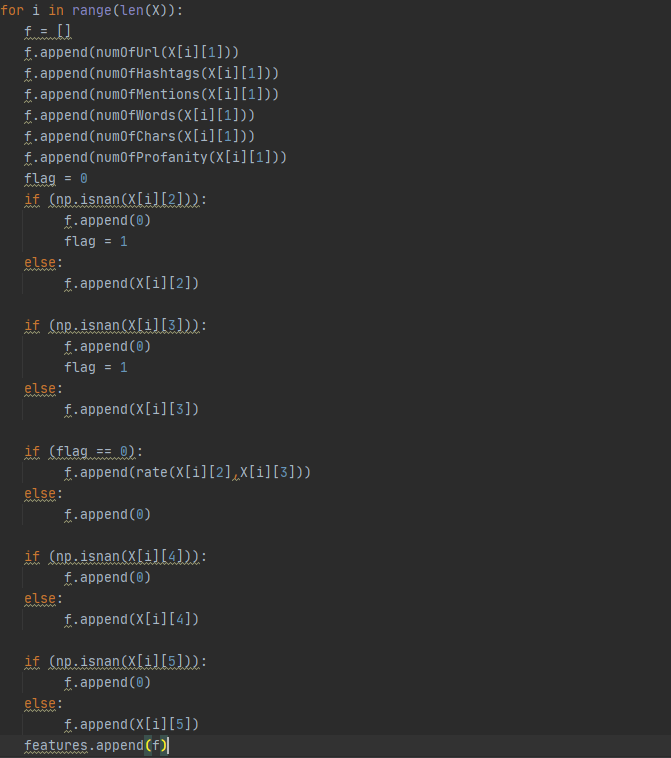


Here the rate of the followers to the sum of the people who the user follows is calculated.



The part above contains the code that has uses the Sklearn API to detect spam tweets. MLP, GNB and RFDT stands for Multiple layer Percepton (Neuron Network) classifier, Gaussian Naïve Bayes classifier and Random Forest Decision Tree classifier respectively.

Each classifier was defined as a function which takes the features and the result if it is spam or not then splits the data to training data and testing data 4:1. It then proceeds to learn from the training data and tests it on the testing data then print the accuracy of the predicted outcome.



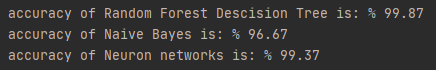
Here the code combines the calculated features for each tweet into an array, and then combines these arrays that describes the features of each tweet into an array of arrays called features.



Lastly, the detector calls the previously defined functions that calls each classifier passing them features array and y array that has the real classification of each tweet, whether it is a spam or not. It will print the accuracy of each classifier on the same set of training data and testing data.

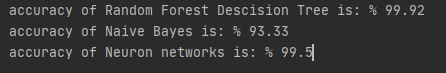
# Simulation And Comparison

The accuracy can slightly differ for each simulation due to the fact that each time, the training set and testing set will be taken randomly from the data, in a ratio of 4 to 1. Three simulations were noted down, one simulation for each case of: having the number of profanity words feature, one without the number of profanity words and one simulation without the features of the missing attributes.



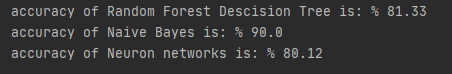
### Figure 1

Figure 1 shows the results of simulation for the case were number of profanity words feature was not included. The result were satisfying for the three classifiers , with the Random Forest decision tree having the most accurate predictions with a %99.87 chance of getting the right predictions. The Neuron Networks classifier having %99.37 accuracy which does not differ by much than the Random Forest Decision Tree classifier. And the Naïve Bayes classifier lags behind by little which is still pretty accurate with an accuracy of %96.67. it is worth to note that as mentioned above, that accuracy may differ for each simulation, with the Random Forest Decision Tree and Neuron Networks classifiers having narrower range in the accuracy, while the Gaussian Naïve Bayes having a wider range, and sometimes even reaching %100 accuracy rate.



### Figure 2

The Number of profanity words feature was included in the simulation results shown in figure 2. Which from can be noticed that the Random Forest Decision Tree and the Neuron Networks classifiers having a slight increase in the accuracy rate, while the Naïve Bayes classifier having a major decrease in the accuracy rate, and sometimes even reaching an accuracy rate of %83.33. Due to that and the fact that the Number of profanity words takes too much time to be calculated, this feature was removed shortly after the simulation.



### Figure 3

The missing attributes features were not included in Figure 3, resulting in a huge decrease in the Random Forest Decision Tree and Neuron Networks classifiers accuracy rate, and not as much decrease in the Naïve Bayes classifier.

# References

1. Unsupervised collective-base d framework for dynamic retraining of supervised real-time spam tweets detection model. Mahdi Washha, Aziz Qaroush, Manel Mezghani, Florence Sedes.
2. Kabakus, A. T., & Kara, R. (2017). A survey of spam detection methods on twitter. International Journal of Advanced Computer Science and Applications.
3. Wu, T., Wen, S., Xiang, Y., & Zhou, W. (2017). Twitter spam detection: Survey of new approaches and comparative study. Computers and Security, 76. doi: 10.1016/ j.cose.2017.11.013.

# Appendix

The python code of this project:

import numpy as np  
from sklearn import metrics  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.datasets import load\_iris  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.neural\_network import MLPClassifier  
import pandas as pd  
import re  
from better\_profanity import profanity  
  
data = pd.read\_csv('train.csv')  
X = data[['Id', 'Tweet', 'following', 'followers', 'actions', 'is\_retweet', 'location']].values  
Y = data[['Type']]  
#tweets = data[['Tweet']].values  
X = np.array(X)  
Y = np.array(Y)  
  
y= []  
for i in range(len(Y)):  
 if (Y[i][0] == 'Quality'):  
 y.append(0)  
 elif (Y[i][0] == 'Spam'):  
 y.append(1)  
  
def numOfUrl(string):  
 url = re.findall(r'(https?://[^\s]+)', string)  
 counter = (len(url))  
 return counter  
  
def numOfHashtags(string):  
 hashtags = re.findall("#([a**-**zA**-**Z0**-**9]{1,15})", string)  
 counter = (len(hashtags))  
 return counter  
  
def numOfMentions(string):  
 mentions = re.findall("@([a**-**zA**-**Z0**-**9]{1,15})", string)  
 counter = (len(mentions))  
 return counter  
  
def numOfWords(string):  
 word\_list = string.split()  
  
 number\_of\_words = len(word\_list)  
 return number\_of\_words  
  
def numOfChars(string):  
 num = len(string)  
 return num  
  
def numOfProfanity(string):  
 word\_list = string.split()  
 number\_of\_profanity\_words=0  
 for p in range(len(word\_list)):  
 if(profanity.contains\_profanity(word\_list[p])):  
 number\_of\_profanity\_words=number\_of\_profanity\_words +1  
 return number\_of\_profanity\_words  
  
def rate(following, followers):  
 r = 0.0  
 if(following == 0 and followers ==0) :  
 r = -1  
 else:  
 r = followers/(following + followers)  
 return r  
def MLP(features , y):  
 clf = MLPClassifier()  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(features , y, test\_size=0.2)  
 clf.fit(X\_train, y\_train)  
 prediction = clf.predict(X\_test)  
 accuracy = metrics.accuracy\_score(y\_test, prediction)  
 print("accuracy of Neuron networks is: %", round(accuracy \* 100, 2))  
  
def GNB(features , y):  
 features, y = load\_iris(return\_X\_y=True)  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, y, test\_size=0.2)  
 gnb = GaussianNB()  
 prediction = gnb.fit(X\_train, y\_train).predict(X\_test)  
 accuracy = metrics.accuracy\_score(y\_test, prediction)  
 print("accuracy of Naive Bayes is: %", round(accuracy \* 100, 2))  
  
def RFDT(features, y):  
 knn = RandomForestClassifier()  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(features , y, test\_size=0.2)  
 knn.fit(X\_train, y\_train)  
 prediction = knn.predict(X\_test)  
 accuracy = metrics.accuracy\_score(y\_test, prediction)  
 print("accuracy of Random Forest Descision Tree is: %", round(accuracy \* 100, 2))  
  
features = []  
for i in range(len(X)):  
 f = []  
 f.append(numOfUrl(X[i][1]))  
 f.append(numOfHashtags(X[i][1]))  
 f.append(numOfMentions(X[i][1]))  
 f.append(numOfWords(X[i][1]))  
 f.append(numOfChars(X[i][1]))  
 f.append(numOfProfanity(X[i][1]))  
 flag = 0  
 if (np.isnan(X[i][2])):  
 f.append(0)  
 flag = 1  
 else:  
 f.append(X[i][2])  
  
 if (np.isnan(X[i][3])):  
 f.append(0)  
 flag = 1  
 else:  
 f.append(X[i][3])  
  
 if (flag == 0):  
 f.append(rate(X[i][2],X[i][3]))  
 else:  
 f.append(0)  
  
 if (np.isnan(X[i][4])):  
 f.append(0)  
 else:  
 f.append(X[i][4])  
  
 if (np.isnan(X[i][5])):  
 f.append(0)  
 else:  
 f.append(X[i][5])  
 features.append(f)  
  
RFDT(features,y)  
GNB(features,y)  
MLP(features,y)