Faculty of Engineering and Technology
Department of Electrical Engineering

Artificial Intelligence

ENCS3340

Automatic Spam Tweet Detection

Prepared By:
Yasmin Abusharbak – 1182523
Munther Anati – 1182028

Instructor:
Dr. Aziz Qaroush

Section "2"

January 10th, 2022

Abstract

This project aims to build an automatic spam tweet detector, by implementing classifiers based on machine learning techniques. Starting from the step of data preparation by pre-processing, and performing features extraction, to building different classifiers, and applying model fitting and evaluation. **In this project, an automated tool for detecting** spam tweets or spammers is built, with the purpose of eliminating their influences. Spam. Unwanted, malicious, unsolicited content or behavior that affects normal social network users, directly or indirectly. Due to the high rate of data generation by social media users, spam detection is such an important approach in the field of machine learning and artificial intelligence.

Contents

Abstract i
1 Introduction 1
  1.1 Machine Learning 1
  1.2 Forms of Learning 1
  1.3 Spam Tweet Detection 1
  1.4 Scikit Learn Library 2
  1.5 Data Set 2
2 Related Work 3
  2.1 Machine Learning Approach 3
    2.1.1 Account-based Features 3
    2.1.2 Tweet-based Features 3
    2.1.3 Graph-based Features 4
    2.1.4 Hybrid Spam Detection Methods 4
  2.2 Other Approaches 4
    2.2.1 Blacklist Approach 4
    2.2.2 Honeybot Approach 4
  2.3 Applications of Spam-Tweet Detection 4
3 Methodology 5
  3.1 Step 1: Data Pre-processing 5
  3.2 Step 2: Features Extraction 6
  3.3 Step 3: Features Selection 7
4 Experiments and Results 9
  4.1 Low Variance Filter- Models Evaluation 11
  4.2 A Combination of Features 12
  4.3 Models Evaluation Based on Tweet-Content 13
  4.4 All Features- Models Evaluation 14
5 Conclusion 14
6 References 16

1 Introduction

1.1 Machine Learning

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. In other words, it is about making machines able to learn to do new things and adapt to new situations.

1.2 Forms of Learning

Depending on the problem, the input and the desired outcome, machine learning can be classified as follows:

1. Supervised Learning: In this case, an agent tries to find the function that corresponds to the example from the data set, in which the inputs in the data set correspond to specific output classes, this type of algorithm is used for classification targeted problems. An example of classifiers that apply this technique is Decision Tree, Naïve Bayes and the Artificial Neural Network.
2. Unsupervised Learning: Learning from patterns without specifying an output, such as clustering.
3. Reinforcement Learning: The agent does not know the exact output for an input, but the agent receives feedback on whether its behavior is desirable. This feedback can come from the outside environment or internally from the agent itself.

1.3 Spam Tweet Detection

Spam presents online in many different forms, including malware dissemination, sending commercial URLs, posting fake news or abusive content, and following or mentioning random users. Users of social media platforms freely generate and consume information leading to unprecedented amounts of data. Specifically on Twitter, one in every 21 tweets is estimated to be spam. It is likely that spammers are always developing new ways to evade detection systems. As a result, some approaches may become obsolete and ineffective in responding to these new tricks. Twitter states that it considers platform manipulation (including spam) to be activity that is intended to negatively impact the experience of people on Twitter. This includes unsolicited or repeated actions. Spam can include malicious automation and other forms of platform manipulation such as fake accounts. Several researches and machine learning tools have been developed to fight the huge increase of spam behavior across Twitter and other social media platforms. Twitter itself now challenges between 8 and 10 million potentially manipulative accounts per week, to guarantee a good experience for users and enhance the positive effect of technology and development.

Some examples of behaviors which would violate Twitter's Rules against spam include:

1. Consistently Tweeting or DMing links only, without any commentary.
2. Posting duplicate or very similar content across multiple accounts.
3. Posting multiple, duplicate updates on one account.
4. Creating duplicate or very similar accounts; or creating fake accounts, impressions or account interactions (such as followers, Retweets, likes, etc.).
5. Posting multiple updates in an attempt to manipulate or undermine Twitter trends.
6. Sending large numbers of unsolicited replies or mentions.
7. Purchasing or attempting to artificially inflate account interactions (such as followers, Retweets, likes, etc.).
8. Using or promoting third-party services or apps that claim to get you more followers, Retweets, or likes; or that claim to be able to get topics to trend.

Thus, Twitter has limited the number of messages and tweets a user can share per day. The limit is 1,000 messages sent per day, tweets: **2,400 per day**. Still, this has led to a need of developing many researches and models to reduce the rate of spam tweets. This project represents one example of these models that aims to classify tweets as **spam** or **quality**.

1.4 Scikit Learn Library

Scikit-learn is an open-source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection, model evaluation, and many other utilities. Scikit-learn provides dozens of built-in machine learning algorithms and models, called estimators. Each estimator can be fitted to some data using its fit method. Machine learning workflows are often composed of different parts, a typical pipeline consists of a pre-processing step that transforms or imputes the data, and a final predictor that predicts target values. After performing pre-processing, data is loaded into the estimator (classifier) and is split into training and testing data. Next, after training (fitting) the model, it is evaluated using the set of testing (unseen) data. All the mentioned steps and many others, can be performed easily using the tools and algorithms Scikit learn (Sklearn) library provides.

1.5 Data Set

The data set provided in this project includes more than 11900 samples collected by professionals for academic and educational purpose. Each sample involves the following information: tweet text, number of followers and number of followings of the user, number of actions (likes and retweets) on the tweet, an indicator of whether it is a retweet or not, as well as the location provided in the user's account, and the classified type of the tweet (Spam/ Quality).

2 Related Work

With over 313 million monthly active users and more than 500 million tweets each day, Twitter is one of the most popular social networking networks. Thus, spam on Twitter has long been a serious yet tough problem to solve. Spammers that use Twitter for nefarious purposes such as scamming genuine users or spreading malicious software advertise through URLs posted within tweets, aggressively follow/unfollow legitimate people, and hijack trending topics to grab their attention, promoting pornography.

2.1 Machine Learning Approach

Many detection and defense strategies have been proposed based on machine learning approach to protect Twitter users from spamming activities. Many unique methods have been created in the recent three years, which have substantially increased detection accuracy and efficiency when compared to those offered before. As a result, several new studies have been conducted on Twitter spam detection strategies.

However, these systems have a number of flaws that prevent them from identifying dynamic spammer activities in real time to the desired level of efficiency. Such that, the majority of these methods are based on supervised machine learning techniques that are trained on static, human-annotated datasets, resulting in low human labor costs. Moreover, due to the lack of information in the tweet object itself, the difficulty of information fabrication, and the variance in social spammers' activities, relying on tiny and static annotated datasets to develop a model to follow-up on social spammers' patterns and tactics is not an efficient option.

Thus, identifying the ideal data-set to work on has been the most challenging part in developing a detection mechanism, and researchers have been encouraged to offer more robust approaches to boost data quality for apps that use Twitter as a primary source of information due to flaws in Twitter's anti-spam mechanisms.

Twitter spam detection methods can be classified as: **Account-based**, **Tweet-based**, **Graph-based** or **Hybrid**. These methods vary depending on the set of the selected **features** they are developed based on.

2.1.1 Account-based Features

Spammers can be identified by examining their Twitter accounts and looking for account-level characteristics. Some of these features, such as biography, location, homepage, and creation date, are meaningless in terms of spam detection because they are user-controlled. Other profile or account-based features can be: the number of following and the number of followers.

2.1.2 Tweet-based Features

To attract attention, spammers usually post a large number of unsolicited tweets to legitimate users. Spammers can be identified by considering the links, mentions, hashtags, number of likes and retweets, and other tweet-level features involved in their tweets.

2.1.3 Graph-based Features

Trying to represent the structure of Twitter in a graph, users and tweets can be represented as nodes in the graph model, and relationships can be represented as links between nodes. These connections demonstrate how the sender and mentions of a tweet are related to one another. Furthermore, these connections are clear evidence of genuine conversations. The distance between the tweet's sender and mentions can be estimated for spam analysis by creating a graph model to represent users and their relationships. Graph-based features are: distance (the length of the shortest path between two users), and connectivity (the strength of the connection).

2.1.4 Hybrid Spam Detection Methods

Hybrid spam detection methods use a combination of spam detection methods discussed in the previous subsections in order to provide more robust spam detection which investigates the possibility of spam in a more efficient way.

Although detection methods can vary in the selected features, the following steps of the process of developing detection methods are quite the same. The selected features are extracted from the data-set and used next to train and evaluate classification models.

2.2 Other Approaches

In addition to the machine learning approach represented in the previous part, other detection approaches presented in several studies, including Blacklist Approach and Honeybot Approach.

2.2.1 Blacklist Approach

The majority of spammers use URL links in their spam tweets to market their services/products. Thus s, detecting tweets containing spam links relying on third-party blacklisting techniques is an effective way of spam detection. It detects spam by searching a list and can be implemented for domain level rather than a single URL. It's a lightweight technique with a lower cost than other classifiers. However, because it is a time-consuming procedure, it is unable to deal with the dynamic nature of spamming activities, and many spammers tend to use shortened URLs, which render blacklisting approaches ineffective.

2.2.2 Honeybot Approach

Traditional supervised learning methods necessitate a human-labeled dataset, which is inefficient in spam drift issues. Like the Honeybot detection method, some spam detection methods rely on community reporting mechanisms. Which can monitor the behavior of social spammers by logging their information, such as account details and any available content.

2.3 Applications of Spam-Tweet Detection

According to the literature review, there are many spam detection methods, however the number of Twitter spam detection systems that can apply detection methods to real-world situations is restricted. As a result, developing a Twitter spam detection system with good performance and high speed is essential and significant.

3 Methodology

This part illustrates the several steps completed to achieve the final goal of this project and develop a spam tweet detector.
Basic steps can be summarized as follows:

1. **Data Preparation (Preprocessing)**: this involves dealing with the raw data provided, by performing preprocessing techniques.
2. **Features Extraction:** in addition to the provided features in the data set, and with the help of those features, several other features were extracted to enhance the model performance and provide better accuracy. Preprocessed data was written into a new csv file, along with the extracted features.
3. **Model Fitting:** various models have been chosen and used to load the preprocessed and balanced data into, to complete the model training (fitting) stage.
4. **Model Evaluation:** after fitting the models, all trained models were stored and ready to be evaluated (tested) with the set of unseen testing data.

The above steps were performed with the help of Python libraries, along with the Sklearn library, which made it quite easier to deal with the data and complete the final goal. The following represents a detailed description for each of the mentioned steps:

3.1 Step 1: Data Pre-processing

Enhancing data quality results in enhancing the performance of the model. Therefore, and as the data is already collected and provided, the first step was implementing preprocessing on the raw data. Data preprocessing is a procedure of several steps of cleaning, organizing and mining of the raw data. The following represents the procedure followed in this project:

1. **Handling Missing Data Fields:** This step was performed to fill in the gaps and the missing data fields in the data set. Missing data can reduce the statistical power of a model performance and can produce biased estimates, leading to invalid conclusions. The data set includes both numeric and text (string) fields, thus, missing fields in numeric features were filled using the mean value, where missing text fields were filled using mode value. Such that:
   Mean: The average value of a column of numeric data.
   Mode: The most common value in a column of text (string) data.
2. **Removing Stop Words in Tweet Text:** Removal of stop words reduces the dataset size and thus reduces the training time due to the fewer number of tokens involved in the training. Therefore, in this step, stopwords such as (the, are, is, am, …etc.) were removed from each tweet text.
3. **Removing Special Characters:** In this step, characters that doesn't carry any information (or non-alphabetical characters) were removed from the tweet text. This is useful when considering the stemming process next, and it helps the model considering key features.

4. **Stemming the Tweet Content**: Stemming is a process where words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix. Stemming is used in information retrieval systems like search engines. It is used to determine domain vocabularies in domain analysis. Many researchers demonstrate that stemming improves the performance of information retrieval systems. This step was performed to prepare the Tweet text to be encoded.
5. **Encoding Tweet Text**: Text encoding is a process to convert meaningful text into number/ vector representation so as to preserve the context and relationship between words and sentences, such that a machine can understand the pattern associated in any text and can make out the context of sentences.
6. **Normalizing Numeric Features:** Since numeric features have different ranges, numeric columns in the dataset (including the extracted features) were changed to a common scale. The great difference in the scale of the numbers could cause problems when you attempt to combine the values as features during modeling. **Normalization** avoids these problems by creating new values that maintain the general distribution and ratios in the source data, while keeping values within a scale applied across all numeric columns used in the model.

3.2 Step 2: Features Extraction

In feature extraction, raw data is transformed into numerical features that can be processed while preserving the information contained in the original data set. Feature extraction helps to reduce the amount of redundant data from the data set; thus, it yields better results than applying machine learning directly to the raw data. It's also worth mentioning that this step is covered by the model in deep learning, but when performing classic machine learning, feature extraction must be done by the user. In this step, several numerical features were extracted using the information contained in the data set. The following represents the extracted features, which were written into a csv file later:

1. **The Number of URLs in the Tweet Text:** this feature represents the count of links (including shorten links) included in each tweet. Spam tweets often consist of uniform resource locators (URLs) having links which are either adult contemporary or out of the context content. With the help of those URLs, spammers redirect the users to those mischievous sites which contain viruses.
2. **The Number of Hashtags in the Tweet Text:** this feature counts the number of hashtags shared in the tweet text. Spammers tend to use hashtags more often, especially the hashtags related to a trending topic (trending hashtags), in order to make their content more popular and easier to reach.
3. **The Number of Mentions in the Tweet Text:** this feature counts the number of accounts mentioned in the tweet through looking for words starting by "@". Spammers tend to mention a large number of users to try to propagate information.

4. **The Following to Followers Ratio:** this feature represents a ratio of the number of the number of accounts a user follows to the number of accounts a user is followed by. Spammers tend to exhibit unusually high following to followers ratios.
5. **The Encoded Tweet Content and Location:** For the text data, after applying stemming operation in preprocessing stage, texts were encoded into numeric representation.
6. **The Number of Characters in the Tweet Text:** this feature counts the number of characters used in creating the tweet, including numbers and symbols.
7. **The Number of Words in the Tweet Text:** this feature counts the number of words written in the tweet, where white-space is used as s separator among words. The count of words is not really an informative enough feature, as spammers can share shorter or longer tweets. However, this feature was combined into the mentioned below features, in order to get more informative tweet- level features, such as the following (7-10) features.
8. **Number of URLs per Words:** the ratio of the number of URLs to the number of words in the tweet.
9. **Number of Hashtags per Words:** the ratio of the number of hashtags to the number of words in the tweet.
10. **Number of Mentions per Words:** the ratio of the number of mentions to the number of words in the tweet.
11. **Number of Capitalized Words per Words:** the ratio of the number of capitalized words to the number of words in the tweet.

3.3 Step 3: Features Selection

After extracting features, a method was developed to select the most informative set to be fed into the classifiers later. The implemented method is **Low Variance Filter**, as variance indicates the spread of the data and how far the points are from the mean. It is known that features with low variance have less impact on the process of classification. Following are the steps of the implemented procedure:

1. Checking for missing fields in the data was the first step and was previously covered in preprocessing.
2. Variance is range dependent; thus, all numeric features need to be normalized. This step was also already covered in the preprocessing stage.
3. The variance was calculated for each feature using the **var()** built-in function.
4. Features with a variance greater than a threshold value were kept, where other features with less variance were omitted.

Low-variance filter method was tested for various threshold values, finally, threshold value was set to be 0.009. Next, another random-based method was tested, where random features were dropped keeping other features included, performing several trials. Another evaluation was done base on tweet content only. Results obtained are recorded in the testing section.

3.4 Step 4: Models Training

Model training (or model fitting) is an iterative process in which the input data is run through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model. Therefore, training is the final step before testing and evaluating.
This part illustrates the set of classifiers trained and evaluated in this project. Five different classifiers were implemented including the following:

1. **Decision Tree Classifier**: a supervised machine learning that is one of the most extensively used classification techniques, due to its simplicity and high efficiency. It continuously splits the data according to a specific parameter, creating a structure of a tree where leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision Tree Classifier was implemented using Sickit-learn library, where data was split into 80% training data and 20% testing data. Next, an object of the implemented classifier was created and provided with the training data represented as feature vectors with the class output.
2. **Naïve Bayes Classifier:** a simple and effective classification algorithm that aids in the development of fast machine learning models capable of making quick predictions. It's a probabilistic classifier, which means it makes predictions based on an object's probability, as defined by Bayes Theorem.
3. **Neural Network Classifier:** also know as Artificial Neural Networks (ANNs), a supervised machine learning method that joins a number of a very simple processors known as neurons, creating a structure that reflects the behavior of the biological neurons in the human brain. Neural networks rely on training data to learn and improve their accuracy over time.
4. **Random Forest Classifier:** a meta estimator that employs averaging to increase predicted accuracy and control over-fitting by fitting a number of decision tree classifiers on various sub-samples of the dataset. The sub-sample size can be controlled by a parameter, otherwise the entire dataset is used to create each tree. It can be used for classification and regression.
5. **XGBoost Classifier:** a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. It can be used to solve regression, classification, ranking, and user-defined prediction problems.

4 Experiments and Results

This part illustrates the last step of a classifier implementation, which is testing or evaluating. At this stage, the quality of a system's prediction is quantified. The performance of a trained model is measured based on several parameters (metrics) including the following:

1. **Precision:** is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.
2. **Recall:** is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.
3. **Accuracy:** is the number of correct classifications divided by the total number of test cases.
4. **F1 score:** is the harmonic mean of precision and recall, and it tends to be closer to the smaller of the two.

The above metrics were observed for each one of the five classifiers mentioned in the previous section, Naïve Bayes, Decision Tree, Neural Network, Random Forest and XGBoost classifiers. Each classifier was tested several times upon several features-combinations.

First, trained models were tested for the entire set of extracted features, including both numeric and encoded text features. Next, the features were selected using the Low Variance Filter method, in which the variance threshold was set to be 0.009. The set of features with variance greater than or equal to the threshold were written onto a file and applied to the models to be evaluated.

Another combination of features was created randomly, by dropping the features that theoretically seem to be less informative, such as the number of words, the number of characters and the Tweet content, as spammers can share shorter or longer tweets, and can control the content of the Tweet. Especially in the provided data-set, Tweet content was noticed to be an unreliable feature.

Moreover, another evaluation was performed based on the tweet content, which was pre-processed previously by applying stemming operation and encoded into a numeric representation.

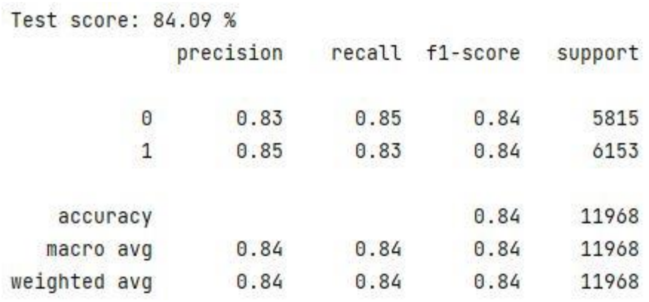
Observed metrics were recorded at *Table 1*, followed by the set of figures of classification reports obtained by applying each feature selection method.

Table 1: Models Evaluation- Testing Metrics

Features Selection Method	Class	Classification Method											
		Decision Tree				Naïve Bayes				Neural Network			
		P	R	A	F1	P	R	A	F1	P	R	A	F1
Low Variance Filter	0	0.83	0.85	0.84	0.84	0.77	0.53	0.70	0.63	0.74	0.59	0.70	0.66
	1	0.85	0.83	0.84	0.84	0.66	0.85	0.70	0.74	0.68	0.81	0.70	0.74
Random Combination of Features	0	0.96	0.97	0.97	0.97	0.53	0.97	0.57	0.69	0.65	0.81	0.69	0.72
	1	0.97	0.97	0.97	0.97	0.87	0.20	0.57	0.32	0.77	0.58	0.69	0.66
All Features	0	1.00	0.99	0.99	0.99	0.83	0.47	0.70	0.60	0.49	1.00	0.49	0.65
	1	0.99	1.00	0.99	0.99	0.65	0.91	0.70	0.75	0.54	0.00	0.49	0.00
Tweet Content Only	0	0.90	0.90	0.90	0.90	0.52	0.56	0.54	0.54	0.50	0.00	0.51	0.00
	1	0.91	0.90	0.90	0.91	0.55	0.51	0.54	0.53	0.51	1.00	0.51	0.68

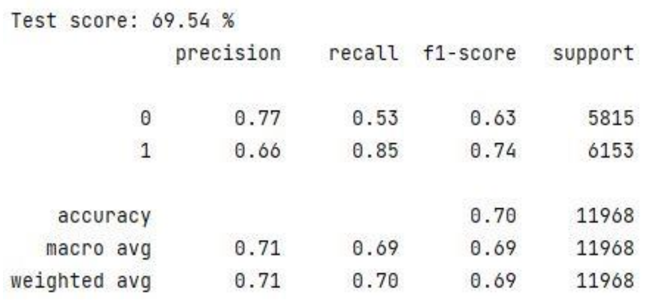
		Random Forest				XGBoost			
		P	R	A	F1	P	R	A	F1
Low Variance Filter	0	0.84	0.84	0.84	0.84	0.77	0.73	0.76	0.75
	1	0.85	0.85	0.84	0.85	0.76	0.79	0.76	0.77
Random Combination of Features	0	0.97	0.97	0.97	0.97	0.92	0.95	0.93	0.93
	1	0.98	0.97	0.97	0.97	0.95	0.92	0.93	0.94
All Features	0	1.00	0.99	1.00	1.00	0.99	0.99	0.99	0.99
	1	0.99	1.00	1.00	1.00	0.99	0.99	0.99	0.99
Tweet Content Only	0	0.89	0.90	0.89	0.89	0.65	0.71	0.67	0.68
	1	0.90	0.89	0.89	0.90	0.70	0.64	0.67	0.67

4.1 Low Variance Filter- Models Evaluation



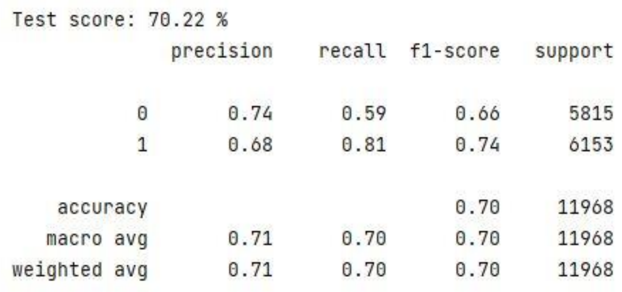
```
Test score: 84.09 %
              precision    recall  f1-score   support

           0       0.83      0.85      0.84      5815
           1       0.85      0.83      0.84      6153

    accuracy                           0.84     11968
   macro avg       0.84      0.84      0.84     11968
weighted avg       0.84      0.84      0.84     11968
```

Figure2 : Decision Tree Classifier



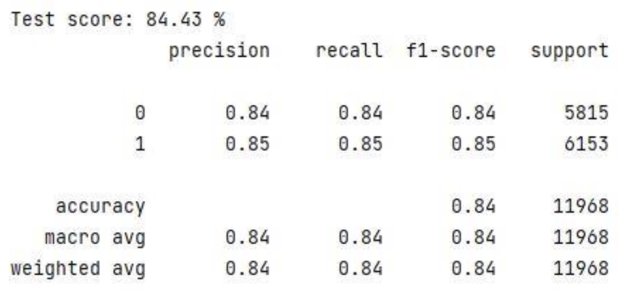
```
Test score: 69.54 %
              precision    recall  f1-score   support

           0       0.77      0.53      0.63      5815
           1       0.66      0.85      0.74      6153

    accuracy                           0.70     11968
   macro avg       0.71      0.69      0.69     11968
weighted avg       0.71      0.70      0.69     11968
```

Figure1 : Naïve Bayes Classifier



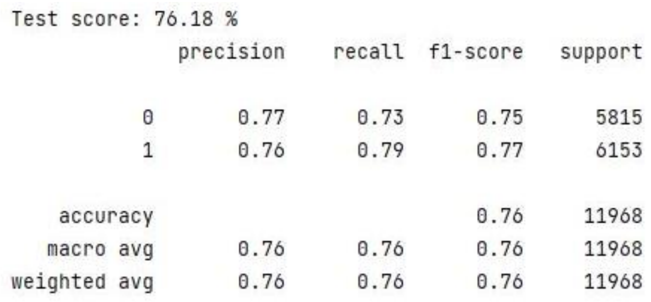
```
Test score: 70.22 %
              precision    recall  f1-score   support

           0       0.74      0.59      0.66      5815
           1       0.68      0.81      0.74      6153

    accuracy                           0.70     11968
   macro avg       0.71      0.70      0.70     11968
weighted avg       0.71      0.70      0.70     11968
```

Figure 3: Neural Network Classifier



```
Test score: 84.43 %
              precision    recall  f1-score   support

           0       0.84      0.84      0.84      5815
           1       0.85      0.85      0.85      6153

    accuracy                           0.84     11968
   macro avg       0.84      0.84      0.84     11968
weighted avg       0.84      0.84      0.84     11968
```

Figure 4: Random Forest Classifier



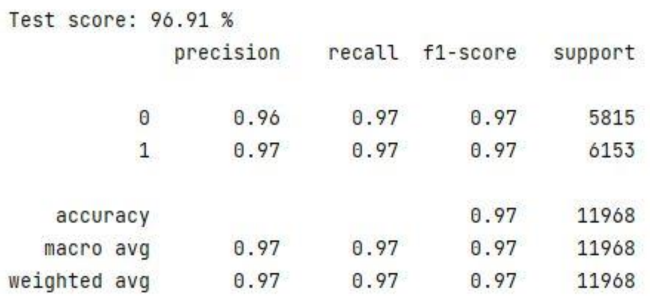
```
Test score: 76.18 %
              precision    recall  f1-score   support

           0       0.77      0.73      0.75      5815
           1       0.76      0.79      0.77      6153

    accuracy                           0.76     11968
   macro avg       0.76      0.76      0.76     11968
weighted avg       0.76      0.76      0.76     11968
```

Figure 5: XGBoost Classifier

Low variance filter method was applied on the set of processed data and extracted features, the threshold value was determined by experiment and set to be 0.009. Final features concluded by this method were written onto the "LVF.csv" file, including: is_retweet, location, URLs, Words, Links_to_Words_Ratio, Caps_to_Words_Ratio and Type. Results specified by the metrics are quite accepted.

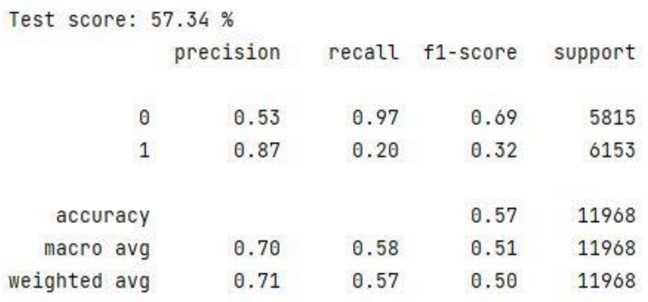
4.2 A Combination of Features

At this stage, a combination of features was created by observing the informativity of features based on the data-set provided. Some features were noticed to be redundantly volatile and uninformative, such as the number of words, number of characters, tweet content, number of followings (because spam and quality accounts can both have high numbers of followings), and other features. Final features were stored in "MyFeatures.csv" file, including the following features: followers, location, URLs, Hashtags, Mentions, Hashtags_to_Words_Ratio, Links_to_Words_Ratio, Caps_to_Words_Ratio and Type. This combination of features provided better performance and better metrics values than the Low Variance Filter method used previously.



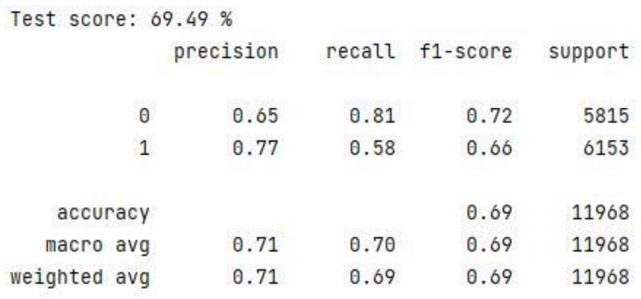
```
Test score: 96.91 %
              precision    recall  f1-score   support

           0       0.96      0.97      0.97      5815
           1       0.97      0.97      0.97      6153

    accuracy                           0.97     11968
   macro avg       0.97      0.97      0.97     11968
weighted avg       0.97      0.97      0.97     11968
```

Figure 6: Decision Tree Classifier



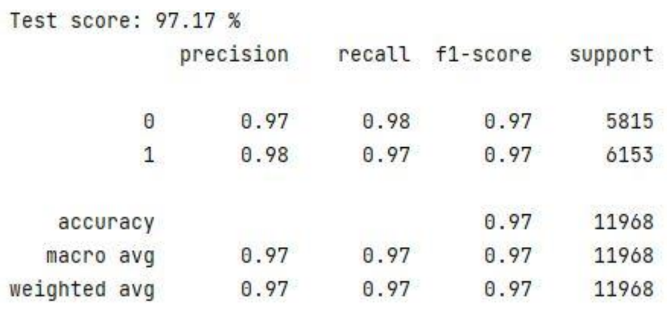
```
Test score: 57.34 %
              precision    recall  f1-score   support

           0       0.53      0.97      0.69      5815
           1       0.87      0.20      0.32      6153

    accuracy                           0.57     11968
   macro avg       0.70      0.58      0.51     11968
weighted avg       0.71      0.57      0.50     11968
```

Figure 7: Naïve Bayes Classifier



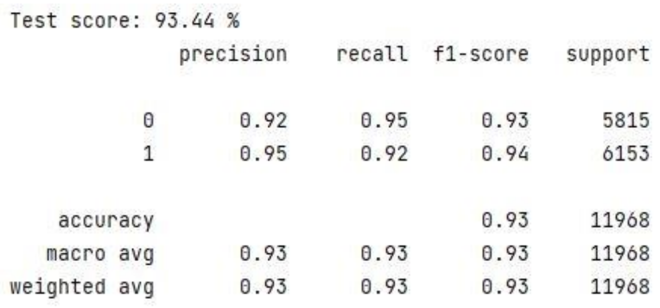
```
Test score: 69.49 %
              precision    recall  f1-score   support

           0       0.65      0.81      0.72      5815
           1       0.77      0.58      0.66      6153

    accuracy                           0.69     11968
   macro avg       0.71      0.70      0.69     11968
weighted avg       0.71      0.69      0.69     11968
```

Figure 8: Neural Network Classifier



```
Test score: 97.17 %
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      5815
           1       0.98      0.97      0.97      6153

    accuracy                           0.97     11968
   macro avg       0.97      0.97      0.97     11968
weighted avg       0.97      0.97      0.97     11968
```

Figure 9: Random Forest Classifier

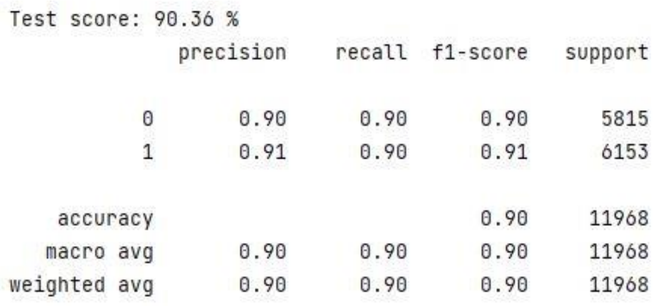


```
Test score: 93.44 %
              precision    recall  f1-score   support

           0       0.92      0.95      0.93      5815
           1       0.95      0.92      0.94      6153

    accuracy                           0.93     11968
   macro avg       0.93      0.93      0.93     11968
weighted avg       0.93      0.93      0.93     11968
```

Figure 10: XGBoost Classifier

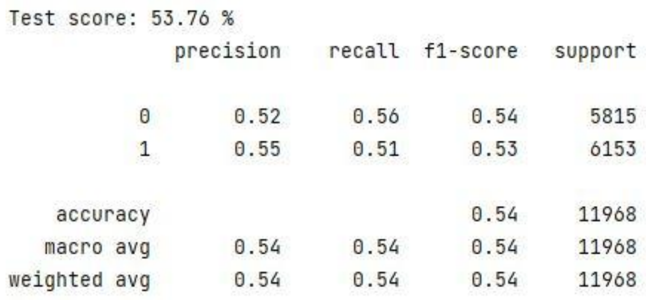
4.3 Models Evaluation Based on Tweet-Content

After applying stemming operation and encoding the tweet text into numeric representation, it was stored into “” file along with the Type. Models were provided with this file and evaluated. Results, as expected, are lower than the metrics values obtained previously, however, for the decision tree and the random forest classifiers, results remained roughly unchanged and high.



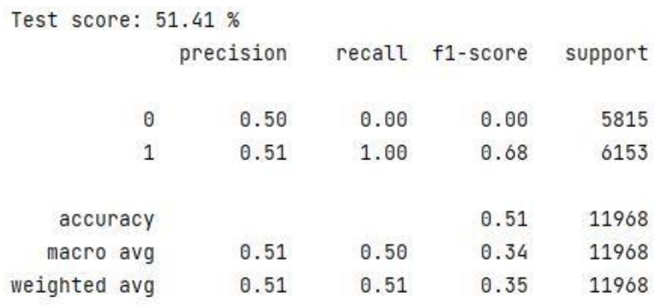
```
Test score: 90.36 %
              precision    recall  f1-score   support

           0       0.90      0.90      0.90      5815
           1       0.91      0.90      0.91      6153

    accuracy                           0.90     11968
   macro avg       0.90      0.90      0.90     11968
weighted avg       0.90      0.90      0.90     11968
```

Figure 11: Decision Tree Classifier



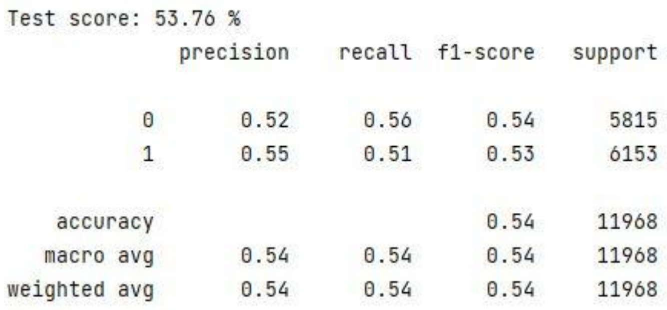
```
Test score: 53.76 %
              precision    recall  f1-score   support

           0       0.52      0.56      0.54      5815
           1       0.55      0.51      0.53      6153

    accuracy                           0.54     11968
   macro avg       0.54      0.54      0.54     11968
weighted avg       0.54      0.54      0.54     11968
```

Figure 12: Naïve Bayes Classifier



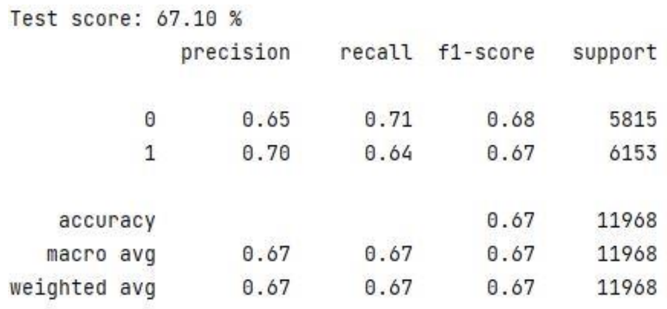
```
Test score: 51.41 %
              precision    recall  f1-score   support

           0       0.50      0.00      0.00      5815
           1       0.51      1.00      0.68      6153

    accuracy                           0.51     11968
   macro avg       0.51      0.50      0.34     11968
weighted avg       0.51      0.51      0.35     11968
```

Figure 13: Neural Network Classifier



```
Test score: 53.76 %
              precision    recall  f1-score   support

           0       0.52      0.56      0.54      5815
           1       0.55      0.51      0.53      6153

    accuracy                           0.54     11968
   macro avg       0.54      0.54      0.54     11968
weighted avg       0.54      0.54      0.54     11968
```

Figure 14: Random Forest Classifier

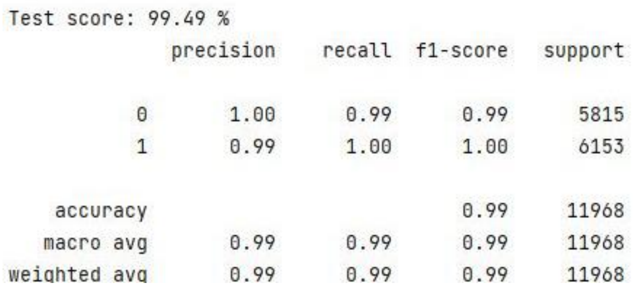


```
Test score: 67.10 %
              precision    recall  f1-score   support

           0       0.65      0.71      0.68      5815
           1       0.70      0.64      0.67      6153

    accuracy                           0.67     11968
   macro avg       0.67      0.67      0.67     11968
weighted avg       0.67      0.67      0.67     11968
```

Figure 15: XGBoost Classifier

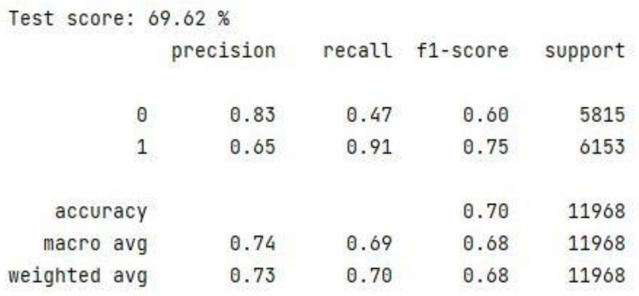
4.4 All Features- Models Evaluation

At this stage, models were evaluated by providing the “ProcessedData.csv” file, in which the pre-processed data and the extracted features are all involved. It is known that involving a large number of features can lead to overfitting, as it increases the model complexity. However, metrics values observed at this stage are the highest, compared to values obtained previously using other features selection methods. Except for the Naïve Bias and Neural Network classifiers, which provided the least metrics values.



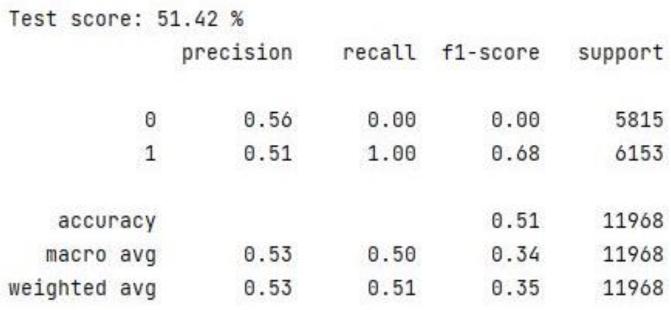
```
Test score: 99.49 %
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      5815
           1       0.99      1.00      1.00      6153

    accuracy                           0.99     11968
   macro avg       0.99      0.99      0.99     11968
weighted avg       0.99      0.99      0.99     11968
```

Figure 16: Decision Tree Classifier



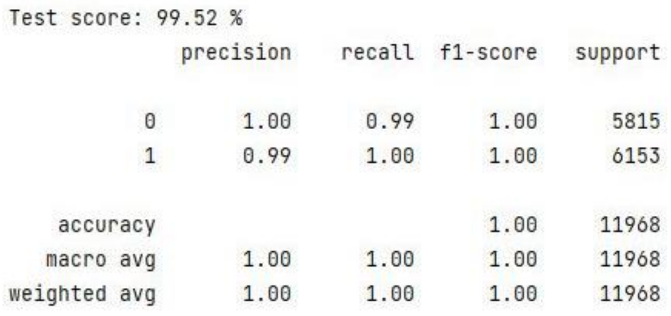
```
Test score: 69.62 %
              precision    recall  f1-score   support

           0       0.83      0.47      0.60      5815
           1       0.65      0.91      0.75      6153

    accuracy                           0.70     11968
   macro avg       0.74      0.69      0.68     11968
weighted avg       0.73      0.70      0.68     11968
```

Figure 17: Naïve Bayes Classifier



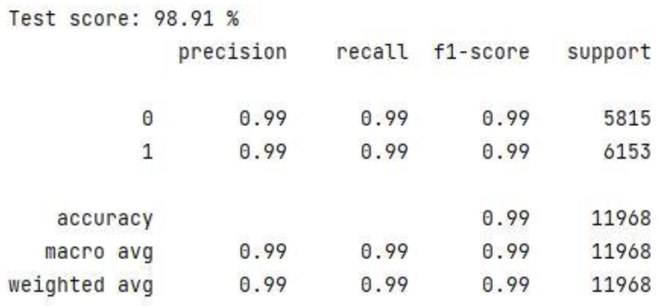
```
Test score: 51.42 %
              precision    recall  f1-score   support

           0       0.56      0.00      0.00      5815
           1       0.51      1.00      0.68      6153

    accuracy                           0.51     11968
   macro avg       0.53      0.50      0.34     11968
weighted avg       0.53      0.51      0.35     11968
```

Figure 18: Neural Network Classifier



```
Test score: 99.52 %
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      5815
           1       0.99      1.00      1.00      6153

    accuracy                           1.00     11968
   macro avg       1.00      1.00      1.00     11968
weighted avg       1.00      1.00      1.00     11968
```

Figure 19: Random Forest Classifier



```
Test score: 98.91 %
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      5815
           1       0.99      0.99      0.99      6153

    accuracy                           0.99     11968
   macro avg       0.99      0.99      0.99     11968
weighted avg       0.99      0.99      0.99     11968
```

Figure 20: XGBoost Classifier

***Note: the program can be tested using any of the implemented classifiers, based on any of the previously mentioned feature selection methods. Once the program is run, it accepts an input from the user, performs classification and displays classification reports.**

5 Conclusion

This project was a great practice to become more familiar with machine learning techniques and concepts, through getting closer to real time applications and developing a solution for one common problem nowadays. In this project, a spam Tweet detector was developed going through all the steps including preprocessing, features extraction, features selection, model fitting and evaluation. Furthermore, dealing with the Sickit-learn and the NLP libraries such as the NLTK library have made the process quite easier and helped achieving the final goal. Final results were measured based on precise metrics, including the accuracy, precision, recall and f1 score, and results were found to be acceptable.

Such projects bring the worker closer to real life problems and required solutions, and encourages them to develop their knowledge regard machine learning field, trying to produce more creative, helpful and efficient solutions in the future.

6 References

[1]: https://www.sas.com/en_us/insights/analytics/machine-learning.html

[2]: https://www.sciencedirect.com/science/article/pii/S0925231218308798

[3]: https://blog.twitter.com/en_au/topics/company/2019/Spamm-ybehaviour-and-platform-manipulation-on-Twitter

[4]: https://www.geeksforgeeks.org/genetic-algorithms/

[5]: https://shortest.link/34Pc

[6]: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[5]: Unsupervised collective-base d framework for dynamic retraining of supervised real-time spam tweets detection model. Mahdi Washha, Aziz Qaroush, Manel Mezghani, Florence Sedes.

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

[7]: Wu, T., Wen, S., Xiang, Y., & Zhou, W. (2017). Twitter spam detection: Survey of new approaches and comparative study. Computers and Security, 76.

[8]: Chowdhury, R., Gourav Das, K., Saha, B., & Kumar, S. (2020). A Method Based on NLP for Twitter Spam detection.

[9]: Sun, N., Lin, G., Qiu, J., & Rimba, P. (2020). Near real-time twitter spam detection with machine learning techniques.