

# Spam Detection on Twitter

AcademicianHelp

## Abstract

Twitter is a social media which most people use in their day to day life. Twitter has grown to be very famous and is currently having an active user base of more than 328 million. Twitter allows users to "follow" another user's account that are of interest to them. Compared to various other social media platform, the connection between users is bi-directional rather than unidirectional connection; this means that a particular user may not be following one of his followers. A user can "like" or "retweet (RT)" a tweet which implies sharing that tweet to his "followers" (Atefeh and Khreich, 2015), while retweet is when a tweet made by one user is shared and used by another user.

This project retrieved 41632 twitter account information, using Twitter API to classify spam and real account. A python code was written to run machine learning algorithms like support vector machine (SVM), decision tree, Naïve Bayes, K-NN and Random forest to find out which algorithm is the suitable for the identification of spam or real accounts on. Moreover, the python program identified ten best features that can be used to identify if an account is a spam account or not.

(Word Count 216)

## Table of Contents

Abstract .....	2
1.0. Introduction.....	8
1.1. Aim and Objectives .....	9
1.2. Research Questions .....	9
1.3. Report Structure .....	10
2.0. Literature Review .....	11
2.1. The Twitter Social Network .....	11
2.1.1. Features of Twitter .....	11
2.1.2. How Twitter Deals with Spam .....	12
2.2. Related Work .....	13
2.3. Twitter Spam Detection Methods .....	15
2.3.1. Account-based Spam Detection Methods.....	15
2.3.2. Tweet-based Spam Detection Methods.....	16
2.3.3. Graph-based Spam Detection Methods.....	17
2.3.4. Hybrid Spam Detection Methods.....	18
2.4. Twitter API .....	20
2.5. Conclusion.....	21
3.0. Methodology .....	22
3.1. Introduction .....	22
3.2. Approach used .....	22
3.3. Advantages and limitations of the approach.....	23
3.4. Classifiers used for the processing.....	24
3.4.1. Support Vector Machine (SVM).....	24
3.4.2. Decision Tree .....	24
3.4.3. Naïve Bayes.....	24
3.4.4. K-Nearest Neighbour (K-NN) .....	25
3.4.5. Random Forest .....	25
3.5. Reason for using Python Language.....	25
3.6. Performance Measurements .....	26
3.6.1. Confusion Matrix .....	26
3.6.2. Accuracy .....	26
3.6.3. Precision .....	27
3.6.4. Recall .....	27
3.6.5. Specificity (SP) .....	27

3.6.6.	Error Rate (ERR) .....	27
3.6.7.	Area Under the Curve (AUC) .....	28
3.7.	Cross-validation .....	28
3.8.	Conclusion .....	28
4.0.	Data Collection and Categorisation .....	29
4.1.	Introduction .....	29
4.2.	Dataset .....	29
4.2.1.	User-Based Features .....	34
4.2.2.	Content-Based Features .....	34
4.3.	Data Collection .....	34
4.4.	Data Clean-up .....	37
4.4.1.	Issues identified during Data Clean-up .....	37
4.5.	Data categorisation .....	37
4.6.	Conclusion .....	38
5.0.	Data Processing .....	39
5.1.	Introduction .....	39
5.2.	Data Processing .....	39
5.3.	Data Processing Results .....	42
5.3.1.	No best features selection results .....	42
5.3.2.	Ten best features selection results .....	43
5.4.	Conclusion .....	43
6.0.	Data Result Analysis .....	44
6.1.	Introduction .....	44
6.2.	Confusion Matrix .....	44
6.3.	Performance Measurement – Accuracy .....	45
6.4.	Performance Measurement – Precision .....	46
6.5.	Performance Measurement – Specificity .....	46
6.6.	Performance Measurement – Recall .....	47
6.7.	Performance Measurement – F1 (F-measure) .....	48
6.8.	Performance Measurement – AUC .....	48
6.9.	Performance Measurement – Error Rate .....	49
6.10.	Conclusion .....	50
7.0.	Discussion .....	51
7.1.	How objectives are achieved .....	55
8.0.	Conclusion and Future works .....	56
9.0.	Project Management .....	57
9.1.	Risk Table .....	57

9.2. Gantt Chart .....	57
Glossary .....	58
References.....	60
APPENDIX A Gantt Chart .....	65
APPENDIX B Ethics Certificate .....	66
APPENDIX C Twitter API connection .....	67
APPENDIX D Processing .....	70

AcademicianHelp

## List of Figures

Figure 1: The relationship between users in Twitter (Atefeh and Khreich, 2015).....	11
Figure 2: The relationships between lists and users (Atefeh and Khreich, 2015). .....	12
Figure 3: The user interface of Twitter which is used to report an account by selecting the motive (Atefeh and Khreich, 2015). .....	13
Figure 4: Method adopted for the project.....	22
Figure 5: Spam Account Twitter Data .....	54
Figure 6: Real account Twitter Data.....	54

AcademicianHelp

## List of Tables

Table 1: Relate objectives with research questions.....	9
Table 2: The advantages, limitations and the approach adopted to overcome the limitations .....	23
Table 3: The data fields its datatype, importance and the feature categorisation. ....	29
Table 4: Data Categorisation .....	38
Table 5: Classifier results before best feature selection .....	42
Table 6: Classifier results after ten best features selection .....	43
Table 7: Summary accuracy.....	45
Table 8: Summary precision .....	46
Table 9: Summary Specificity .....	46
Table 10: Summary Recall .....	47
Table 11: Summary F1 .....	48
Table 12: Summary AUC.....	49
Table 13: Summary Error Rate .....	49
Table 14: Summary of Algorithms results .....	50
Table 15: Discussion of best feature selection.....	51
Table 16: Statistical analysis for the best features.....	54
Table 17: object achievement.....	55
Table 18: Risk table .....	57

AcademicianHelp

## 1.0. Introduction

Twitter is amongst the most famous social media platforms which allow a social network of users to publish data of about 140 characters referred to as "tweet". Twitter gives users an opportunity to share their messages concerning everything identified with as genuine including news, occasions, celebrities, political issues, and so on (Bravo-Marquez, 2013). As indicated by Twitter, Twitter has 313 million months to month dynamic users that publish 500 million tweets for each day, which amounts to about 350,000 tweets for each moment (Bai, 2017). Be that as it may, this fame and common sense also draw in the interest of spammers.

Twitter was overwhelmed by a lot of malignant tweets that were sent by a huge number of spammed users account; this occurred around April of 2014 (Chen et al., 2015). In August of 2014, Twitter discovered that 8.5% of its monthly dynamic users, roughly 23 millions of them, have consequently reached their servers for regular updates (Seward, 2014).

Spammers present their contents as valuable or as applicable contents, and they send them to the user. The authentic users mistake the spam information for an important one. Spams are difficult to stop, email administrations such as Gmail, Microsoft and the likes have been effectively distinguishing spam messages; despite this fact, spam messages still exist online. These administrations announced that email spamming has risen to 90 to 95 per cent of the aggregate email trades (Waters 2009). After organisations have successfully identified spams, they cannot stop spammers; the spammers take advantage of this to entice users to click on a spam interface (Perveen et al., 2016). The seriousness of the danger posed by spamming has increased with the rise of online social communities, and Twitter stands out from amongst the most common online social communities that have been exceedingly influenced by spam (Perveen et al., 2016). Twitter spamming is more focused on the trending topic on Twitter and it is not easily infiltrated because of its hash-tag administrator (Perveen et al., 2016).

Twitter users cover all segments of life including teachers, students, superstars, government officials, advertisers or clients. Every age group can use Twitter; however, the age group that uses it the most is between 55 to 64 years (Perveen et al., 2016). There are about 60% of users that get to twitter from their mobile phones. Because of this spamming concern, a user faces numerous issues with query items that result in duplicating and unnecessary data.

Additionally, this can be exceptionally stressful since a user needs to look through all the data to get a general perspective of the subject. It is difficult to detect the location of spam in the Twitter network,



and this is because of the use of URLs, abbreviations, casual languages and present-day language ideas (Stringhini et al., 2010). Old-style strategies for identifying spam data miss the mark here. Till date, there is literature on several methods for identifying spams on Twitter.

In this project, the Twitter API is used to collect raw data of accounts in Twitter. The collected data is used to perform a comparison between binary classification methods such as Support Vector Machine (SMV), Naïve Bayes, K-NN, Decision Tree, and Random Forest. The comparison derived the most suitable binary classifier for Twitter to identify spam accounts.

### 1.1. Aim and Objectives

This project primarily aims at studying machine learning classification methods to decide which method has the highest precision in detecting spammer accounts on Twitter.

The objectives of the project are provided below:

**Objective 1:** Performing secondary research to identify similar methods used in identifying the Twitter spam accounts

**Objective 2:** Using Twitter API to collect raw account details and clean up the data to perform the analysis.

**Objective 3:** Performing analysis of the clean data using binary classification methods such as Support Vector Machine (SMV), Naïve Bayes, K-NN, and Decision Tree, and Random Forest

**Objective 4:** Comparing the analysis and suggesting the binary classifier that can be used to detect spam accounts on Twitter

### 1.2. Research Questions

The research questions for the project linked to the objective are provided in table 1.

Table 1: Relate objectives with research questions

Objectives	Research Questions
<b>Objective 1:</b> Performing secondary research to identify the similar methods used in identifying the Twitter Spam accounts	<b>Research question 1:</b> What are the current methods used to identify the spam accounts in Twitter?
<b>Objective 2:</b> Using Twitter API to collect raw account details and clean up the data to perform the analysis.	<b>Research question 2:</b> What details of the account need to be collected from Twitter to identify whether the account is spam or not?

<p><b>Objective 3:</b> Performing analysis of the clean data using binary classification methods such as Support Vector Machine (SMV), Naïve Bayes, K-NN, and Decision Tree, and Random Forest</p>	<p><b>Research question 3:</b> What machine learning algorithm can be used to achieve the appropriate results in detecting spam accounts on twitter?</p> <p><b>Research question 4:</b> What are the properties or factors that distinguish spam accounts from non-real accounts?</p>
<p><b>Objective 4:</b> Comparing the analysis and suggesting the binary classifier that can be used to detect the spam account on Twitter</p>	<p><b>Research question 5:</b> What is the best binary classifier to detect the spam accounts on Twitter.</p>

**Research question 1:** What are the current methods used to identify the spam accounts in Twitter?

**Research question 2:** What details of the account need to be collected from Twitter to identify whether the account is spam or not?

**Research question 3:** What machine learning algorithm can be used to achieve the appropriate results in detecting the spam accounts on twitter?

**Research question 4:** What are the properties or factors that distinguish spam accounts from non-real accounts?

**Research question 5:** What is the best binary classifier to detect the spam accounts in Twitter.

### 1.3. Report Structure

The report structure is provided below

- Section 2 discusses the background study of the project.
- Section 3 provides a detailed information regarding the methodology used for this project.
- Section 4 discusses the data collection and categorisation
- Section 5 focuses on the data processing steps involved in the project
- Section 6 is the complete analysis of the data results.
- Section 7 provides the discussion regarding the results derived in this project.
- Section 8 contains the conclusion and future works

## 2.0. Literature Review

### 2.1. The Twitter Social Network

Twitter is a microblogging service that allows its users to convey compact information (otherwise known as tweets) which can be seen on their friend's pages. It can also be compared with other social networking sites such as Myspace and Facebook (Kwak et al., 2010). A username is the only tool used to identify a Twitter account; sometimes, actual names are also used. A Twitter user may begin to "follow" a second user 'U'. That user, therefore, sees user U's tweets on her page. User U who is "followed" can follow back if she desires. Tweets can be collected utilising hashtags which are prominent words, starting with a "#" character. Hashtags enable users to seek tweets regarding subjects of that are of interest to them. Whenever a user likes somebody's tweet, she can "retweet" that information. Therefore, that information will appear to every one of her followers. A user can choose to secure her profile. In so doing, any user who may like to follow that private user requires her authorisation. Twitter has grown over the years with an active monthly users that are currently more than 328 million (Arun et al., 2017).

#### 2.1.1. Features of Twitter

Twitter allows accounts to "follow" other different accounts that are of interest to them. Compared to various other social media platform, the connection between users is a bi-directional one and not a unidirectional connection since a particular user may not be following one of his followers. The user can "like" or "retweet (RT)" a tweet which implies sharing that tweet to his "followers" (Atefeh and Khreich, 2015). The connection between users in Twitter is shown in Figure 1. Every user has a unique Twitter username, and users can post tweets that talk about others by including their usernames beginning with "@" character which is referred to as "mention" on Twitter (Atefeh and Khreich, 2015). Users are promptly alerted with notifications when a mention, like, or RT transpires on any of his tweets (Atefeh and Khreich, 2015).

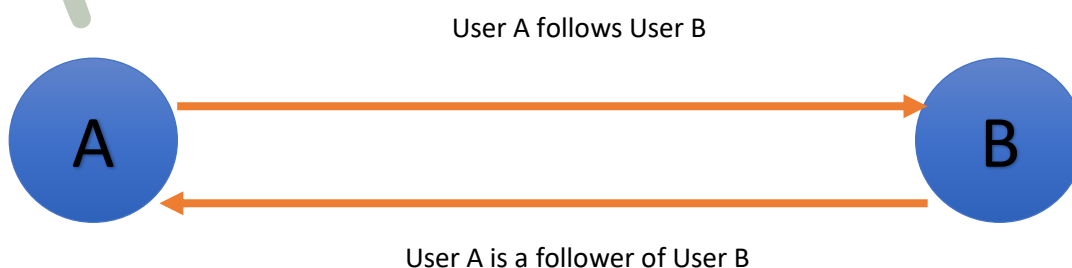


Figure 1: The relationship between users in Twitter (Atefeh and Khreich, 2015)

An additional element of Twitter is that it gives users a chance to make users switch their account type from public to private and vice versa, keeping in mind that the end goal is to arrange their interests by gathering users whose interests are alike or comparable (Kim et al., 2010; Yamaguchi et al., 2011). The list that the user subscribed to are classified as "subscribed to" while the list the user is included by their owners are called "member of" as seen in figure 2.

a

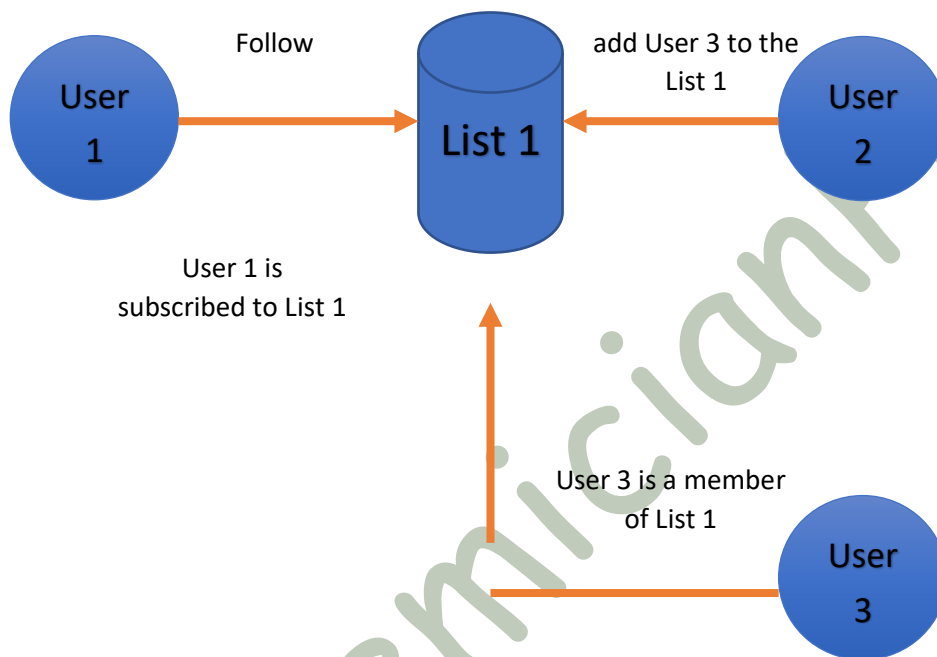


Figure 2: The relationships between lists and users (Atefeh and Khreich, 2015).

### 2.1.2. How Twitter Deals with Spam

Twitter utilises both manual and mechanised administrations to fight spammers. The manual method involves Twitter giving users a chance to report spammers via the spammers' profile pages. Twitter provides a UI as seen in Figure 3 to report the account by choosing the reason.

Report ×

---

Help us understand the issue with @A94730809. What's the problem with this account?

- I'm not interested in this account
- They are posting spam
- Their account may be hacked
- They're being abusive or harmful

[Learn more](#) about reporting violations of our rules.

Next

Figure 3: The user interface of Twitter which is used to report an account by selecting the motive (Atefeh and Khreich, 2015).

A different method as detailed in literature involves reporting spammers to the authority "@spam" account (Song et al., 2011; Wang 2010; Kaur et al., 2016; Chen et al., 2016; Verma, and Sofat, 2014; Gee and The, 2010). Still, as indicated by the current report by Twitter, this strategy for revealing spam is obsolete (Atefeh and Khreich, 2015). Likewise, Wang reports that this technique is mishandled by both inventors and spam (Wang, 2010). These manual methodologies are stressful and may not be sufficient to distinguish between all spammers because of billions of users. Twitter utilises different elements, for example,

- 1) The publishing of a copy of the messages over numerous accounts or various copy of the messages in a single account,
- 2) The following/unfollowing of a huge number of accounts within a brief span,
- 3) The possessing of a huge number of spam protests documented against the account,
- 4) Forcefully liking, following, and retweeting,
- 5) The publishing of malignant connections,
- 6) The publishing of tweets which for the most part, comprise of connections rather than also publishing individual updates, and
- 7) The presentation of inconsequential tweets to a trending subject to figure out what lead is thought to be spamming (Atefeh and Khreich, 2015).

## 2.2. Related Work

In as much as social networks are unequivocally given the idea of a system of trust, the abuse of this trust may prompt huge repercussions. Research carried out in 2008, demonstrated that 41% of the Facebook users who were reached, recognised a friend request from a random individual [10]. L. Bilge et al. (2009) demonstrate that after a hacker has gained entry into the system of a victim, the victim will probably click on any connection which constitutes the messages published, regardless of

if he/she is familiar with the hacker or not. Another interesting conclusion by experts (Jagatic et al., 2007) is that phishing attempts will probably do well if the hacker utilises stolen data from victims' friends in social networks to create their phishing messages. For instance, phishing messages from the sippy bag were frequently sent from a user's friend list and afterwards, a user is regularly deceived into trusting that messages originate from trusted friends and subsequently, readily gives login data of his/her email account. According to Yardi et al. (2009), the creators made a prominent hashtag on Twitter and watched how spammers began to utilise it in their messages. They talk about a few elements that may be used to differentiate spammers from genuine users; for example, a duplication of messages and hub degree. The use of basic components such as the duplication of messages and hub degree, in any case, may not be sufficient, as long as there are some young Twitter users or TV anchors that publish many messages. In Stringhini et al., (2010), bigger spam research was explained. The creators of Stringhini et al. (2010) produced nectar profiles as bait for spammers to connect with them. They made 300 profiles each, on popular social networking sites such as Twitter, Myspace and Facebook. Their 900 profiles pulled in 4250 friend's requests (mostly from Facebook). However, 361 out of 397 friend requests on Twitter were from spammers. They later proposed using highlights such as the rate of tweets with URLs, message similarities, cumulative messages sent, number of friends for spam identification. Their discovery using the Random Forest classifier can create an incorrect positive ratio of 2.5% and an incorrect negative ratio of 3% on their Twitter database (Mccord, and Chuah, 2011). In Wang (2010), the creators suggested utilising chart established and content established components to distinguish the spammers. The chart established elements they utilised include the number of followers, the number of friends (the number of individuals a user is following) and a notoriety score which is characterised as the proportion between the number of followers over the aggregate total of the number of followers and the number of individuals a user is following. The guess is that if the number of followers is little compared to the number of individuals that a user is following, the notoriety is small and consequently, the likelihood is increased that the related account is spam. The substance based elements they used include

- (a) similarity of content,
- (b) number of tweets that contain HTTP interfaces in the last 20 tweets,
- (c) the number of tweets that contain the "@" signs in a user's 20 last tweets,
- (d) the number of tweets that comprise the "#" hashtag sign. By using a Bayesian classifier, the creator discovered that of the 392 users that are named spammers, 348 are genuinely spam accounts and 44 users are false positives, so the precision of his spam recognition scheme is 89% (Mccord, and Chuah, 2011).

Debatin et al. (2009) portrayed that beforehand, all spam discovery techniques checked just single messages or account for the presence of spam. They concentrated on the identification of spam attacks that manage different accounts to spread spam on the twitter network. Bilge et al. (2009), suggested a diagram display referred to as guided chart prototype to find the friend and follower connection on twitter network. The use of Nave Bayesian classifier chart-based and content-based components are recommended for the identification of spam tweets. In chart-based elements, three elements are utilised; specific friends, followers, and the notoriety of a user is ascertained for finding spam. In content-based components, copy tweets, HTTP connections, replies and mention and trending topics processed for spam identification. In Yardi et al. (2009), Nikita Spirin researches about URLs shared by users on Twitter and the calculation of spam for those users that share these connections in the network and use the data for web spam location calculations, by suggesting another arrangement of URL determined components to portray a twitter user. Additionally, she suggested an answer for the development of programmed dataset by dissecting URLs shared by non-spam users in social media for the issue of web spam discovery (Mccord, and Chuah, 2011).

Stringhini et al. (2010) discussed a different method for spam discovery in the Twitter network. The authors researched the spread of spam in the network. Besides, they tried to find out if there is a format that spammers use to multiply spam in the system and to decide if the accounts are either bargained or overwhelmed by spammers or if a set of accounts are made solely for spam purposes in the network. They look at the qualities of the diagram of spam tweets and perform Trust Rank strategy on the gathered information. Wang (2010), presented highlights for spam tweets discovery without prior insights of the user and utilise measurable introduction for the research to resolve a dialect that will be used to distinguish spam in twitter stories.

### 2.3. Twitter Spam Detection Methods

#### 2.3.1. Account-based Spam Detection Methods

Account-based spam recognition techniques depend on the components (or their combination) of the Twitter account which are recorded in Table 1. Lee et al. (2010) suggested a honeypot-based way to deal with distinguishing spam in social media platforms. The elements they considered when identifying spam are the lifespan of the account on Twitter, the regular daily tweets, the proportion of the number following and number of followers, the rate of bi-directional friends, the proportion of the number of URLs in the 20 most recently published tweets, the proportion of the number of a unique URLs in the 20 most recently published tweets, the proportion of the number of usernames in the 20 most recently published tweets, and the proportion of the number of individual usernames in the 20 most recently published tweets. As suggested by Lin and Huang (2013), a strategy to identify spam in Twitter is done on the premise of two elements:

(1) URL ratio which characterises the proportion of the number of tweets with URL in the aggregate quantity of tweets, and

(2) connection ratio which characterises the proportion of the number of tweets collaborating over the aggregate number of tweets.

As suggested by Gee and Teh (2010), a strategy regarding the account based components, for example, followers-to-following proportion, the number of tweets to account lifetime proportion, the average time between posts, publishing time variety, maximum idle hours, and connection division. This report's scope is the use of the manual method for identifying spam in Twitter which is obsolete as it was recently discussed.

### 2.3.2. Tweet-based Spam Detection Methods

Tweet-based spam recognition techniques depend on the components (or their combination) in a tweet. URL separating approaches utilise static or dynamic crawlers to research recently watched URLs. Likewise, they utilise URL or domain boycotting to distinguish suspicious URL redirections into account, and the landing site's source code (HTML). A phishing discovery strategy regarding lexical components of a URL was introduced by McGrath and Gupta (2008). The elements they consider for the recognition of phishing are the length of URL and the domain name, the original piece of the domain name, the nearness of brands in URLs, and misuse of URL-connection and free web hosting administrations. Ma et al., (2009) suggested a strategy of investigating URLs to recognise the malignant sites. The elements they use distinguishes malicious sites containing WHOIS resources. Examples of such include, who is the enlistment centre of the site, who is the registrant of the site, when the site is enrolled, domain name resources, for example, a time-to-live (TTL) worth for DNS documents, and geographic resources. It includes, in which nation does the IP address have a place, the speed of the uplink connection nearby lexical elements of URL. Canali et al. (2011) is a channel that utilises static examination methods to distinguish the harmful elements of a site. The source of the site's components originate from:

(1) the HTML component of the site, for example, the number of components with little range, the number of components comprising of suspicious components, the number of URLs included, and the number of known malicious examples,

(2) the related JavaScript code, for example, catchphrases to-words proportion, the number of the long strings nearness of translating schedules, the likelihood of shellcode nearness, and the number of DOM-altering capacity, and



(3) the comparison of URL, for example, the number of suspicious URL designs, the nearness of subdomains or IP addresses in URLs, and the TTL esteem for DNS A and NS record.

Since Prophiler utilises static investigation strategies, it is not ready to recognise malicious URLs inserted into the active component. Its example includes, Java applets, flash and some portion of JavaScript which is currently the most commonly used programming language (Parveen et al., 2016) Strategies concerning the dynamic investigation methods (Whittaker et al., 2010; Wang et al., 2006; Thomas 2011; Cova et al, 2010) utilise virtual machines and mechanized web programs including Selenium for thorough examination of components. Chhabra et al. (2011) show a phishing location technique concerning URL investigation. Their strategy is uniquely intended to have the capacity to analyse abbreviated URLs which are generally utilised by Twitter to control spam tweets as discussed recently. The elements of the suggested technique that utilise identifying phishing via URL are the number of clicks, land spread, fleeting spread, and web fame. Lee and Kim (2013) is a suspicious URL recognition structure for Twitter which researches connections of URL divert chains. Lee and Kim (2013) uses 14 components to distinguish the suspicious URL, for example, the length of URL diverts, the number of various landing URLs, the relative numbers of various Twitter accounts, the similarities in the account creation dates, the closeness in the number of followers and following, the similitude in the follower following proportion, and the comparability of tweets. Martinez-Romo and Ajauro (2013) suggest a tweet-based spam discovery strategy which concentrates on the investigation of the language utilised as a part of tweets. In particular, the language prototype they utilise are

- (1) the language prototype of the tweets identified with an inclining point,
- (2) the language prototype of the tweet, and
- (3) the language prototype of the page linked to the tweet. Like the account based spam recognition techniques, many Twitter spam discovery strategies utilise tweet-based elements related to other spam identification to give a stronger spam location.

### 2.3.3. Graph-based Spam Detection Methods

Diagram-based spam discovery techniques depend on the components (or their combination) in a tweet. Song et al. (2011) remove the separation and availability between the tweet's sender and mentions. While separate characterises the length of the shortest path between the tweet's sender and mentions, association characterises the quality of the connection between users. Diagram-based spam location techniques use the chart information patterns to show elements of Twitter as hubs and edges. In this manner, charts are regularly utilised by social networks such as Facebook and Twitter (Ugander et al., 2011; Weaver and Tarjan, 2013; Myers et al., 2014; Gabielkov and Legout, 2012) which are for the most part based on users, subjects, and bi-directional relationships. Notwithstanding,

the chart-based components provide the best execution regarding precision and ability to separate spammers from real users. Other diagram-based spam recognition techniques are introduced in hybrid spam identification strategies since they are combined alongside other spam location strategies.

#### 2.3.4. Hybrid Spam Detection Methods

Hybrid spam identification techniques utilise a blend of spam location strategies depicted in preceding subsections to give more dynamic spam identification which researches the possibility of spam more exhaustively. Stringing et al. (2010) suggests a method regarding both accounts based and tweet-based elements. They are both proportional to the number of friend request that the user sent to the number of friends she has. They are additionally proportional to the number of tweets which make up the URLs to the aggregate number of tweets the user has, the similarity of tweets sent by the user, the number of tweets sent he sends, the quantity of friends the user has, and the likelihood of whether an account used a record of names to select its friends or not. A tweet-based spam recognition approach suggested by Gao et al. (2012) is based on the social level of the tweet's sender, the historical background of communication, the size of the group, the regular time interval, the regular number of URL in tweets, and the unique number of URL in tweets.

Chen et al. (2015) introduce a continuous spam identification strategy for Twitter regarding the twelve inconsequential elements which are separated from a dataset made-up of 6.5 million spam tweets. The elements they in which they tried to use to identify spam on Twitter are age of the account, the number of followers, the number of those following, the number of likes the account got, the number of records that the account has, the number of tweets in the account, the number of retweets, the number of hashtags used as a part of the tweet, the number of mentioned users in the tweet, the number of URLs used as a part of the tweet, the number of characters used as a part of the tweet, and the amount of digits used as a part of the tweet. A hybrid twitter spam location technique suggested by Wang (2010) has to do with the diagram-based and tweet-based components. The diagram-based elements considered in the suggested technique are the number of followers, the number of those following, a notoriety score which is figured as the proportion between the number of followers over the total number of followers and following. The tweet-based elements considered in the suggested strategy are tweet comparability, the number of tweets which make up the URLs in the last 20 tweets, the number of tweets that make up the mentions in the last 20 tweets, and the number of tweets that contain hashtags. A twitter spam recognition technique suggested by Yang et al. (2013) refers to a blend of the diagram-based, tweet-based, and account-based elements.

The proposed strategy uses stronger components including the number of bidirectional connections, the proportion of bi-directional connections, the connection between significance, grouping number related to tweet-based and account-based elements. The example includes the number of followers,

the number of following, the number of tweets sent by the account, the length of existence of the account, the proportion of the number of tweets makeup of the URL, the proportion of the quantity of tweets that make up the hashtags, the number of copy tweets, the proportion of spam word, the proportion of the number of tweets utilised to answer to others, and the proportion of the quantity of retweets. Benevenuto et al. (2010) suggest a crossbreed spam location strategy concerning the account-based elements. The example includes the number of followers, the number of following, the proportion between followers compared to following, the number of tweets sent by the account, the number of mentions the account got, the number of answers, and the proportion of tweets gotten from the account's followers. The tweet-based elements of the suggested strategy are the amount of words contained in each tweet, the number of URLs per word, the number of expressions of each tweet, the number of characters of each tweet, the number of hashtags on each tweet, the number of mentioned on each tweet, the number of URLs of each tweet, and the number of times the tweet is retweeted.

Chu et al. (2010) demonstrate a technique to order Twitter accounts as human, bot, and cyborg which depends on both account-based and tweet-based elements. The components they consider to sort the Twitter account into human, bot or cyborg are the quantity of the proportion of tweets that contain URLs, gadget makeup, the number of the proportion of followers to friends, the connection of interests, and if the account is approved. Amleshwaram et al. (2013) suggest a crossover Twitter spam location technique given both account-based and tweet-based components. They classify spammers into two:

- (1) user-driven, and
- (2) URL-driven.

The elements they consider for spam investigation are the number of remarkable mentions, spontaneous mentions, seizing trends, crossing point with approved patterns, variation in tweet intervals (VaTi), variations in tweet width (VaTw), proportion of VaTi and VaTw, tweet sources, copy of URLs, copy of domain names, IP/domain fluxing, tweet's language uniqueness, relationship between tweets, URL and tweet similarities, followers-to-following proportion, and profile description's language difference of opinion.

Chakraborty et al. (2012) suggest a hybrid strategy, given account-based and tweet-based elements which utilise some modern elements. The example includes spam score of profile description, name, and screen name, appearance or nonappearance of profile picture and normal compatibility of the same hashtag. McCord and Chuah (2011) introduce a mixture technique regarding account-based and tweet-based components to aid spam identification. The components they employ in the suggested technique are the conveyance of tweets over a 24-hour time frame, the number of URLs, the aggregate

quantity of answers/responses in the most 100 late tweets, the number of retweets in the 20-100 last tweets, the aggregate number of hashtags in the 100 last tweets. Wang et al. (2015) propose a spam identification strategy regarding account-based, tweet-based, natural language dialect processing (NLP), and supposition highlights. Some special components which they utilise to distinguish spam are length of the profile name, naturally or physically made assessment vocabularies, the number of exclamation marks, the number of question marks, most extreme word length, mean word length, the number of upper casing words, the number of void areas, and part of speech (POS) labels per tweet.

#### 2.4. Twitter API

Before starting, there should be a twitter account, and the credential should be obtained. The credentials include API key, API secret, Access token and Access token secret. In order to access the Twitter API, the following changes have to be made;

**Step 1:** There should be an existing twitter account or a new account has to be created

**Step 2:** We have to log in to the Twitter account by pressing on this link <https://apps.twitter.com/>. This gives a dev Account with the same name of the user's account.

**Step 3:** Click on “Create New App”

**Step 4:** The form has to be filled, and the term and conditions should be accepted, then finally the created twitter application has to be clicked.

**Step 5:** Then “Keys and Access Tokens” tab has to be clicked from where the “API key” and “API secret” is copied. After that, “Create my access token” has to be clicked from where “Access token”, and “Access token secret” is copied.

A python wrapper is used for performing the API requests such as searching users and downloading tweets. With the help of this library, OAuth and API queries are handles which give a simple python interface. To get the OAuth key, a Twitter App has to be created; this will give access to Twitter's API.

Python Twitter Tools is a library which has been used to connect the Twitter API and to download data from Twitter. There are also other different libraries used for different programming languages, in this case, we have used python since it is easy to use and also supports Twitter API.

- The python tool can be downloaded from <https://pypi.python.org/pypi/twitter>.
- Then the Python Twitter Tools package has to be installed by typing the commands given below:

```
$ python setup.py --help
```

```
$ python setup.py build
```

```
$ python setup.py install
```

## 2.5. Conclusion

From the secondary research conducted, it is clear that there is a threat of spam account in Twitter and many research has been done to differentiate spam accounts from the real accounts. Some of the methods of identifying a spam account from the real account suggested in the literature are an account-based, tweet-based, graph-based and hybrid method. However, there are loopholes in the approach provided which includes the non-provision of a specific algorithm or adequate features in categorising an account. However, this project attempts to fill this void by introducing an appropriate machine learning algorithm to identify spam accounts. Furthermore, ten features of the accounts were selected to categorise an account as either a spam account or a real account.

AcademicianHelp

## 3.0. Methodology

### 3.1. Introduction

This Chapter mainly focuses on discussing the method adopted in collecting data, cleaning-up data, and processing data. Moreover, the processed spam and real (not spam) account files will be used to analyse the different machine learning algorithms which are support vector machine, decision tree, Naïve Bayes, Random Forest and K-NN.

### 3.2. Approach used

This project involves four major steps; however, the first three steps will be repeatedly performed until the account details from twitter is 40,000. The approach adopted is provided in Figure 4.

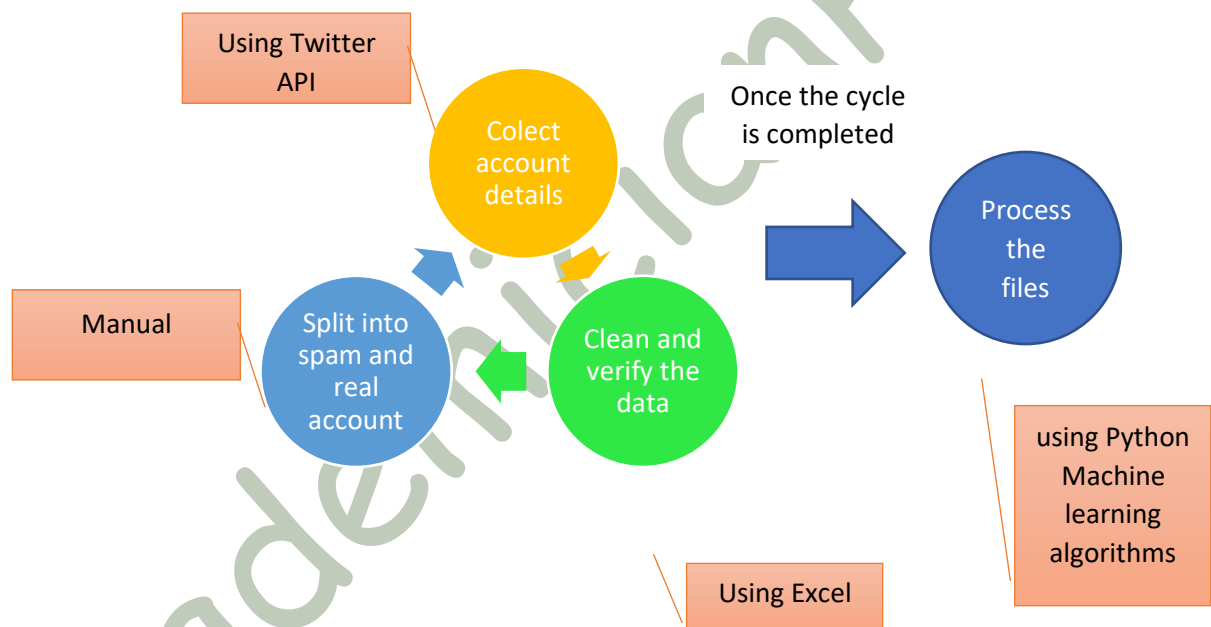


Figure 4: Method adopted for the project

The detail description of the steps is provided below:

**Step 1:** Collect account details using twitter API. The program was developed in python language in order to collect information from the twitter. The data was collected between June 2017 and mid of July 2017.

**Step 2:** Clean and verify the details collected from the step 1. The data cleaning and verification were done manually in the Excel sheet.

**Step 3:** After the step 2 completion, the cleaned data will be split into spam account (negative) and real account (positive) using some specific constraint.

NOTE: Step 1 to step 3 will be repeated. Once the complete 40,000 twitter account details are collected, cleaned and sorted, then the following step 4 will be done.

**Step 4:** After completing step 3 repeatedly, the two-separate file will be processed to identify which machine learning algorithm is appropriate for the identification of the spam account in twitter. The Python language is used to implement different machine learning algorithm to process the data files. The cross-validation was performed for data training.

### 3.3. Advantages and limitations of the approach

The advantages and limitations of each stage used in this project are listed in table 2. Moreover, table 1 includes the approach adopted in order to overcome the identified limitations for each stage.

Table 2: The advantages, limitations and the approach adopted to overcome the limitations

Stages	Advantages	Limitations	The approach adopted to overcome the limitations
Stage 1	The python program will automatically use the Twitter API to access the information and download it. Moreover, the program will store the information into the excel sheet for the user to easily access the downloaded data.	Twitter has a daily limitation of downloading 1000 account information. The account that is used to download the information will be blocked for several hours.	Many Twitter accounts were created to download the twitter account information. Therefore, there are some duplicate account information retrieved.
Stage 2	The data will be verified and cleaned manually to ensure the accuracy of the data.	The manual data clean-up and verification will require more time.	The excel data filter option is used to filter the information to verify and clean-up the data.
Stage 3	The specified constraints will be checked in order to split the account into real and spam. The manual categorisation ensures accuracy.	The manual categorisation will require more time	The excel formula and data filter options are used to categorise a specific account into real and spam.

Stage 4	The python program will automatically take the categorised accounts and run the machine learning algorithms to produce the results.	The program is slow in execution because the data input is 40,000 records.	The 16GB RAM computer was used, and no other program was running during the execution of this processing.
------------	---	--	---

### 3.4. Classifiers used for the processing

The machine learning algorithms implemented in Python language for processing are Support Vector Machine, Decision Tree, Naïve Bayes, K-NN, Random Forest.

#### 3.4.1. Support Vector Machine (SVM)

The support vector machine was proposed to interpret the pattern recognition issues (Vapnik, 1995). The data is mapped in an exorbitant dimensional input space using this approach and then designs an ideal disjointed hyperplane within this expanse. A quadratic software issue is mostly used, whereas for neural network architectures, an inclined tutoring approach, on the one hand, deteriorates from the genuineness of most of the native minima (Huang et al., 2014).

A hyperplane or an array of hyperplanes betwixt classes is construed by a support vector machine if the classes are properly disjointed from one another; a decent disjointing created by SVM, since the overall greater the separation betwixt data points and the hyperplane, the less the fallacy can be gotten by the classifier.

#### 3.4.2. Decision Tree

An easy and broadly utilised classification method is the Decision Tree Classifier. The uncomplicated concept is employed to resolve the classification issue. Every time it accepts a response, a follow-up query is requested until a resolution concerning the class tag of the data is attained. Decision Tree Classifier acts as an array of meticulously drafted enquiries concerning the features of the test record (Bui et al., 2014).

A decision tree is constructed, even though they were subsequently derived from lesser subsets of training dataset of a tree-based algorithm. The tree nodes constitute the characters while the leaf nodes constitute a classification or resolution. This tree is utilised to predict the investigation of the dataset.

#### 3.4.3. Naïve Bayes

Naïve Bayes classifiers in machine learning can be regarded as a group of easy probabilistic classifiers founded on the usage of Bayes' hypothesis along with strong (naive) freedom



presumptions betwixt the characters. Naive Bayes classifiers are greatly measurable since they require some factors that correspond with the number of inconsistencies (characters/ determinants) in an educative issue. A closed form impression using maximum-likelihood tutoring can be made through investigation. This allows linear duration compared to costly repetitive estimation as used for various kinds of classifiers (Patil, and Sherekar, 2013).

Established on Bayes' theorem along with the sound presumption that the characteristics are dependent on self-governing provided by the class label, Naive Bayes classifiers are easy probabilistic classifiers.

#### 3.4.4. K-Nearest Neighbour (K-NN)

K-Nearest Neighbor can likewise be referred to as a lazy learning classifier. Decision tree and rule-based classifiers are created to understand a prototype that charts the data characteristics to the class label immediately the tutoring information is ready. Therefore they are regarded as eager learning classifiers. Compared to the eager learning classifier, K-Nearest Neighbor does not build a classification prototype from information by paring the investigation example with K training instances; it carries out groupings also. It determines its class, rooted in the resemblance to K nearest neighbours (Bidder et al., 2014).

KNN is an easy approach which is used to group information by a large poll of its k neighbours. By utilising a parallel appraisal, it mainly restores all accessible learning information and classifies the recent information (Bidder et al., 2014).

#### 3.4.5. Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression including various operations, which is used by setting up a large number of decision trees at learning duration and outputting the class which is the approach of the mean prediction (regression) or classes (classification) of the separate trees. Random decision forests improve for the decision trees' manner of over-fitting to their learning plan (Ellis et al., 2014).

The grouping of training estimation can also be referred to as Random forests or random decision forests. It is managed by creating a large number of decision trees on the learning information array, after which result can be grouped because of the method of the classes (Ellis et al., 2014).

### 3.5. Reason for using Python Language

The primary reason for using Python to implement the data collection and processing is provided below:

- **Improved Programmer's Productivity:** The language uses a vast aid library and polished article aligned pattern to raise the programmer's output.
- **Integration Feature:** Python combines the Enterprise Application Integration which allows improvement of web services by combining COBRA or COM features. Therefore, it possesses a strong authority competency as it knows instantly via C, C++ or Java via Python. Python likewise means XML and various markup languages as it can be used on every new OS using similar byte set of symbols.
- **Extensive Support Libraries:** This comprises of the vastness. For example, internet service implementation, OS commands and interfaces and a series of behaviours that uses large quality libraries. A larger part of the extremely utilised programming functions is earlier set up into it, which holds back the extent of cyphers to be composed in Python. By identifying built-in function in machine learning SKlearn package in Python, the classifiers are summoned.

### 3.6. Performance Measurements

Several performance measurements are suggested to estimate the classification model, and they include recall, error rates, accuracy, precision, and so on. It is not advisable to estimate the classifier utilising a single performance measure. Although, it may become a demanding and strenuous responsibility for scientists to comprehend and examine the outcomes when several metrics are utilised. It is common for a single prototype to perform better on a particular metrics while it might not work on the next. Although there is no single agreement as to which operation to select over the rest (Japkowicz, 2008). Where the extracted class can be found most of the time, there is for a fact sheet cluster classification dataset which has a variation in class allocation. Estimate the performance of the classification approach to determine the impact of the forecast outcomes in the direction of txt class. Six metrics have been considered, and they rely on the outcomes of designing chaotic matrix. Accuracy, precision, recall, specificity, and f1 are therefore utilised as a performance matrix for this piece of research.

#### 3.6.1. Confusion Matrix

The total number of correct and incorrect predictions which are made by classifying the model compared to the actual outcome of the data is shown in a confusion matrix. The actual outcome is the targeted value. Matrix is  $N \times N$  where  $N$  stands for a total number of target values. The data in the matrix is usually used to evaluate the performance of models (Powers, 2011).

#### 3.6.2. Accuracy

The most unlearned performance measure is the Accuracy. It is the rate of accurately classified tags on every forecast which can also be estimated by employing the formula below (Powers, 2011)

$$\text{Accuracy} = \frac{T\text{Positives} + T\text{Negative}}{T\text{Positives} + T\text{Negative} + F\text{Positives} + F\text{Negative}}$$

For symmetric dataset and if the value of incorrect affirmatives and the incorrect contrary is about the same using this measure; therefore, the only accuracy is insufficient as long as the class allocation of this outcomes is compared (Powers, 2011).

### 3.6.3. Precision

The accuracy that is part of the most popularly utilised performance measure is the Precision (Powers, 2011). The rate of the accurate affirmative tag over all the accurate forecasts, encompassing accurate inspections which are inaccurate, can be termed as Precision. It can be calculated using the formula below (Powers, 2011).

$$\text{Precision} = \frac{T\text{Positives}}{T\text{Positives} + F\text{Positives}}$$

### 3.6.4. Recall

The recall is the entirety that is likewise defined as responsiveness or verifiable affirmative ratio (Powers, 2011). It is the rate of accurately forecasted affirmative tags. The denominator of recall formula calculates all affirmative activity, notwithstanding where they were forecasted accurately by the prototype (Powers, 2011).

$$\text{Recall} = \frac{T\text{Positives}}{T\text{Positives} + F\text{Negatives}}$$

### 3.6.5. Specificity (SP)

The number of true inaccurate forecasts, divided by the aggregate amount of inaccurate, is used to estimate Specificity. It can be estimated using the formula below (Powers, 2011):

$$\text{Recall} = \frac{T\text{Negatives}}{T\text{Negatives} + F\text{Positives}}$$

### 3.6.6. Error Rate (ERR)

The error rate is calculated as the number of all incorrect predictions divided by the total number of the dataset. The error rate is the complement of accuracy, which can be computed by following formula:

$$ERR = \frac{FPositive + FNegative}{TPositive + TNegative + FPositive + FNegative}$$

### 3.6.7. Area Under the Curve (AUC)

Area Under the Curve (AUC), Area under the ROC curve is often used as a measure of the quality of the classification models. A ROC curve is plotting True Positive Rate (TPR) against False Positive Rate (FPR) which depicts relative trade-off between benefits (true positives) and costs (false positives).

### 3.7. Cross-validation

An evaluation method which is better compared to residuals is cross-validations. One drawback of residual evaluation is that they do not indicate how well the learners will make changes for the given new prediction data which they have not seen before. This can be avoided by not providing all the data sets for beginners. Therefore, some of the information is removed before providing it to beginners. Once the training is completed for the beginners, the removed data can be used to test the learners' performance when they provide new information. This is the basic idea for a whole class of model evaluation method known as the cross-validation.

Cross-validation is a powerful way to deal with overfitting. One of the most common used cross-validations is K-fold method. K-fold was performed by partitioning the dataset into k folds (10 folds is used). K-1 folds are used for training purposes, and one-fold is used for testing. This process has been repeated ten times with different testing part for each execution. The cross-validation method is used to validate with and without the best features in this project.

### 3.8. Conclusion

The complete method used in this project is the background study of the classifiers which are Support Vector Machine, Decision Tree, Naïve Bayes, K-NN, Random Forest that will be used for the validation as well as the performance measurement units which are accuracy, precision, recall, specificity, and f-measure (f1).

## 4.0. Data Collection and Categorisation

### 4.1. Introduction

This chapter discusses the dataset that will be downloaded from the Twitter using its API, and it also provides a clear explanation of how the data is collected.

### 4.2. Dataset

The Twitter API is used to retrieve much information regarding the twitter user accounts. This information will be used to identify whether the specific account is spam or real. The data fields that will be retrieved regarding the twitter account and its datatype is provided in table 2. The features or attributes of the account is mainly divided into two types which are User-Based Features and Content-Based Features. The table 3, also includes the classification of the feature to its relevant types.

Table 3: The data fields its datatype, importance and the feature categorisation.

<b>Dataset attributes</b>	<b>Datatype</b>	<b>Importance of the attributes</b>	<b>Categorisation of Features</b>
user_id	Long Integer	This unique ID which is used to identify a profile. Each profile has its unique user ID.	User-based features. This feature is used to remove a duplicate data of a user
screen_name	Text	This is the name that displays in a user's profile. A particular user can be searched and viewed using this screen name.	User-based features. This feature is important because most of the spam accounts have random a name, for example, a number or a few characters are mixed in some account because the machine which gives a random name generates them
verified	Boolean	This will check if a user is verified or not.	User-based features. This feature is important to identify the real accounts

Dataset attributes	Datatype	Importance of the attributes	Categorisation of Features
			because there are many accounts which are spam
default_profile	Boolean	These are the profile details of a user. The account holder usually fills these details. The details include the name, date of birth, and so on.	User-based features. This feature is essential to identify the difference between spam and non-spam account; mostly spam account is generated by a machine which has an empty profile
default_profile_image	Boolean	This is a picture which can be uploaded by a user. This picture can also be viewed by other users along with the screen name when they are searched.	User-based features This feature helps to identify spam from non-spam accounts; mostly spam account is generated by a machine and the machine does not assign profile image to the accounts.
favourites_count	Long Integer	When a user tweets something other users can like and comment on it, which will tell us how reliable that user is.	User-based features. This feature indicates how the other users interact with this accounts since spam counts do not have any interaction.
followers_count	Long Integer	A user can follow other people on twitter. When they follow the people, they can view their profile, like and	User-based features. This feature tells us if the account has followers or not because usually, spam accounts do not have many followers in them

Dataset attributes	Datatype	Importance of the attributes	Categorisation of Features
		comment on their posts.	
friends_count	Long Integer	One user can become friend with another when they follow each other and share details between them.	User-based features. This feature identifies if an account has friends or not because spam accounts do not have a significant number of friends
geo_enabled	Boolean	The users can click on their location when needed on Twitter. This is when they tweet the location of the tweet to make it appear.	User-based features. This feature helps us to identify spam accounts because spam accounts tweet by machine, using a tweets dataset which does not contain a location
notifications	Boolean	Users get a notification on their device when another user likes or comments on their post.	User-based features. This feature helps in the categorisation because spam accounts do not enable notification as they are not human. Therefore, notifications are not important to them
time_zone	Text	Twitter is being used in all the countries around the world. When a user tweets something, the time of the post will appear at the bottom of the tweet made.	User-based features. This feature is important to identify spam account with the others features because this feature is mostly missing from spam accounts. The reason is, the machine does not assign the time

Dataset attributes	Datatype	Importance of the attributes	Categorisation of Features
			zone when generating the accounts
location	Text	The location feature can be used where when a user tweets something the location of the tweet will also appear.	User-based features. This feature is important to identify spam account with the others features because this feature is mostly missing on spam accounts because the machine does not assign the location in account profile when generating the accounts
listed_count	Long Integer	A user can rate their profile with different emotions such as sad, happy, and so on. Based on these emotions, they can also add new users.	User-based features. This feature tells us if the accounts have any interaction with other accounts when they put the account on the list
created_at	DateTime	When a new account is created, the date and time of creation are saved.	User-based features. This feature helps us to identify the spam or non-spam account because twitter company always deletes spam accounts. The last use of a spam account will be months.
profile_use_background_image	Boolean	This image appears at the top of a user profile which can be viewed by other users.	User-based features. This feature helps to identify spam accounts from non-spam. Most cases, spam



Dataset attributes	Datatype	Importance of the attributes	Categorisation of Features
		A user can upload an image as per their wish.	machine do not assign background image in accounts when generated
statuses_count	Long Integer	The total number of status shared by each user is noted.	User-based features. This feature tells us how the accounts interact with each other
sample_tweet	Long Integer	Twitter offers a set of sample tweets (such as sample templates) which can be used by users to tweet.	User-based features. This feature lets us know how many simple tweets an account has, which indicates the account activity
total_hashtags	Long Integer	Users can tag many things using the hashtag sign. The total hashtags made will also be noted.	Content-based features. This feature indicates how many hashtags had been tweeted by an account to know accounts activity
total_links	Long Integer	Users can copy paste a URL from another website and post it on twitter. The total number of links posted by a particular user will be noted.	Content-based features this feature indicates how many links had been tweeted by an account to know accounts activity
total_mentions	Long Integer	A user can be mentioned by other users in a tweet. The total number of mentions will be saved.	Content-based features. This feature indicates how many times an account has been mentioned by other accounts. It is important

Dataset attributes	Datatype	Importance of the attributes	Categorisation of Features
			to know the account's activity since spam accounts do not have any interaction.
favorite_count	Long Integer	This is used to find out how reliable the content of a particular tweet is.	Content-based features. This feature tells us how many favourite_count the accounts has to indicate to accounts interaction
retweet_count	Long Integer	A user can retweet a tweet which has already been made by another user.	Content-based features. This feature tells us how many retweets count the accounts have indicated in accounts interaction

#### 4.2.1. User-Based Features

The twitter permits its users to establish a network of users by following them; also, users allow others to follow them. From the observation, it is clear that the spammers follow a large number of users in order to get their attention. It is also said that the spammers are found in a high ratio. Twitter has another main feature where users can be added and removed from a user's account list, but a spammer is usually not added as a part of this list.

#### 4.2.2. Content-Based Features

Twitter users can add and mention hashtags in the tweets they make; the same tweet can appear in many hashtags. This feature is being misused by spammers as their goal is to get maximum followers as possible. The spammers usually tweet only using links which makes it easy for Twitter to catch them and clear their methods. These spammers use more characters along with the link to avoid getting a spam flag. The spammers also remove the HTTP and HTTPS from the URL they tweet.

#### 4.3. Data Collection

The Python code was written to connect to the Twitter using Twitter API and download the required information and save it into an excel file.

The code has four major libraries imported which are time, tweepy, pandas and csv. The reason for importing the modules are provided below:

```
import time ## To use time-related functions

import tweepy ## This enables the python to communicate with Twitter

import pandas as pd ##loading tabular data from excel

import CSV ##read and write tabular data in CSV format.
```

The tokens required for the user authentication in twitter is written in a separate file. The tokens are consumer\_key, consumer\_secret, access\_token and access\_token\_secret. The separate file name is config.py. The config.py file will be loaded into a list in the program using the following code:

```
config = {} ## Declare empty list

exec(open("config.py").read(), config) ##load the values from the file to the list
```

The initial twitter authentication was done with the information loaded from the config.py file. The code used for that is provided below:

```
## Intial twitter authentication #

auth = tweepy.OAuthHandler(config["khWZKMC74kvOe8qnZGVXuRZ5g"],
config["7zXwNQYpMxgAXQFL7DBa2sthFiVN2UdjC2QDj0iIK6z3MXa0A0"])

auth.set_access_token(config["836672321871998977-pJO6A5UCwKrtynzK9zolgGg90jGW2LC"],
config["cbbW267cHuOKxqIojKYtFEYWMe5q4qnpypRrDq6fHwRm"])

api = tweepy.API(auth)
```

The twitter ids of the account will be retrieved and stored into the list using the following code:

```
twitter_ids = fields.user_id
```

Once the account ids were retrieved and stored in a list, then the other account information were retrieved. The sample code that used to retrieve the user information is provided below:

```
for id in twitter_ids:

    try:

        ## getting user information

        user = api.get_user(id)
```

```

screen_name = user.screen_name

print("Processing: ", screen_name)

verified = user.verified

default_profile = user.default_profile

default_profile_image = user.default_profile_image

```

The following code is used to get the tweets information of the account:

```

timeline = api.user_timeline(screen_name = user.screen_name, count = 100, include_rts = True)

```

Using the timeline information, the number of hashtags, links, mentions, fav\_count and retweet\_counts are retrieved. The code used for that is provided below:

*for tweet in timeline:*

```

    i += 1

    number_of_hashtags += len(tweet.entities['hashtags'])

    number_of_links += len(tweet.entities['urls'])

    number_of_mentions += len(tweet.entities['user_mentions'])

    number_of_fav_count += tweet.favorite_count

    number_of_retweet_count += tweet.retweet_count

```

Some twitter exceptions were handled in the code, which are code 88 that pauses the download for 15 minutes, code 63 is a protected account so it is removed from the list, and the bad user ids. The exception handling code is provided below:

```

if str(e).find("'code': 88") != -1:
    print("Twitter Limitation. Pausing for 15 minutes....")
    fields = fields[fields.user_id != id]
    time.sleep(60*15) #Sleep for 15 minutes

else:

    if str(e).find("'code': 63") != -1:

        print(id, "has been suspended. Removing it from the list...")

```

```
elif str(e).find("Not authorized") != -1:

    print(screen_name, "is protected. Removing it from the list...")

else:

    print(e)

    print(id, "is a bad user ID. Removing it from the list...")
```

The retrieved files will be stored into the file using the following code:

```
fields.to_csv(fileName, encoding='utf_8', index=False)
```

#### 4.4. Data Clean-up

During the clean-up session, the data downloaded has been standardised and normalised. The standardisation was done for attributes such as time zone and location because this attribute information is provided as a text from the Twitter API.

Moreover, the normalisation was done for attributes such as following and list count, because the scale is not similar among the attributes. Apart from the standardisation and normalisation, the duplicate data were removed manually by checking the values of the attributes. If the values of the entire attributes of the account are the same, then it is considered as duplicate and will be removed. Additionally, a new column is created and the year is extracted from the data and time provided from the Twitter API. Furthermore, the Arabic text in the location and time zone are converted into the English language.

##### 4.4.1. Issues identified during Data Clean-up

The major issue identified during the data clean-up is that the dataset has some missing values, such as location and time zone, which are not set by the user. The missing values are considered important because the reasonable percentage of the negative accounts have the attribute not set by the spammers.

#### 4.5. Data categorisation

There are several steps involved in categorising the downloaded data for the twitter account into two categories which are spam account and real account. The detail description of the steps involved in categorising the account into real and spam is provided in table 4.

Table 4: Data Categorisation

Step Number	Step Description	Account Category
1	The data filter is used to extract the accounts that have verified attribute as TRUE.	Real/Positive
2	The data filter is used to extract account details which have all the attributes such as default_profile, default_profile_image, default_profile_use_bckground_image, followers_count, friends_count to zero.	Spam/Negative
3	The data filter is used to extract account details which have friends that are more than 100, followes_count more than 100, simple tweets more than 100.	Real/Positive
4	The data filter is used to filter accounts which have some hashtags less than simple tweets and a small number of retweet and a few mentions.	Spam/Negative
5	Manual checking of the twitter account to find out whether it is spam or a real account	Spam/Negative or real/Positive

#### 4.6. Conclusion

This chapter mentioned the dataset as well as how it will be retrieved and how it will be cleaned. However, the data retrieval is automated by developing a python code. On the other hand, the data clean-up was performed manually using Excel.

## 5.0. Data Processing

### 5.1. Introduction

This chapter mainly focuses on discussing the data processing which includes running different classifiers in the python code to get the percentage of different performance measurement. Moreover, the system identifies automatically, the best feature by selecting the feature that has a high score. This chapter provides a thorough description of the python program that performs the processing, as well as the different running responses.

### 5.2. Data Processing

Several new functionalities were used during coding. Therefore, there are some additional imports. These imports have been attached below, and there are comments added next to each of these of the imports mentioning their use.

```
Import pandas as pd ##loading tabular data from excel  
import math ##Module that use math functions  
from random import shuffle ## The module is used for random shuffle  
from sklearn import preprocessing ## module used classifier processing  
from sklearn import SVM ## importing Support Vector machine classifier  
from sklearn. Tree import DecisionTreeClassifier ## importing decision tree classifier  
from sklearn.neighbours import * ## import K-NN classifier  
from sklearn.naive_bayes Import GaussianNB # importing naive Bayes classifier  
from sklearn. Ensemble import RandomForestClassifier, AdaBoost classifier ## import random forest classifier
```

There are two functions implemented in this system; they are the feature selection concerning kbest and the print result feature. Below is the explanation of these two functions.

#### **Feature selection concerning kbest**

The set of kbest does not accept values which are positive; this is done to normalise the features. There are three variables which have been declared at the beginning of the function; they are, features, classes, feature\_amount. The features of this function are then transformed into different ranges by scaling each of the function separately. Then the data has to be fixed for the transformation to happen.

Then the same step is done for SelectKBest. Based on the k highest score, the feature selection is processed. Then the data has to be fixed for the transformation to happen. Once that is done, the features have to be identified based on its score. These selected features are zipped and returned.

The code used for these have been attached below with comments next to them.

```
def feature_selection_with_kbest(features, classes, feature_amount):  
  
    mm = MinMaxScaler() ##Transforms features by scaling each feature to a given range.  
  
    normalized_features = mm.fit_transform(features) ##Fit to data, then transform it.  
  
    sk = SelectKBest(chi2, k=feature_amount) #Select features according to the k highest scores.  
  
    transformed_features = sk.fit_transform(normalized_features, classes) ##Fit to data, then  
transform it.  
  
    selected_features_and_scores = sorted (enumerate(sk.scores_), key=lambda x:x[1],  
reverse=True)[:feature_amount] ##Identify the features and its score  
  
    return list(zip(*selected_features_and_scores))[0] #zip the selected features and return it
```

This feature\_selection\_with\_kbest function has been called in another function, which is printResult.

### **Print Result**

The print result function has two variables declared; they are labels and results. Once this is done a generation matrix is created as shown below.

```
cm = confusion_matrix(labels, result)
```

This generation matrix has four function calculations; they are accuracy, precision, specificity and recall. Each of these functions has a calculation as shown below.

$$p = tp + fn$$

$$n = tn + fp$$

All the functions have been calculated using float as a data type. The code (mathematical calculation) for each of these calculations has been attached below.

$$accuracy = float(tp + tn) / (p + n)$$

$$precision = float(tp)/(tp+fp)$$

$$specificity = float(tn) / (tn + fp)$$



$$recall = float(tp)/(tp+fn)$$

Along with these calculations, f-measure is also calculated.

$$f1 = float(2*tp)/(2*tp+fp+fn)$$

Once all the calculations are completed, the results are printed in a particular format as shown.

```
print('\tCV Accuracy: {:.2f}%'.format(accuracy*100))
```

then the existing real accounts are connected with their relevant excel file. The name of the excel file has to be mentioned in the code.

```
file_name_1 = "positive.csv" #Link the real account Excel file
```

```
file_name_2 = "negative.csv" #Link the spam account Excel file
```

When the excel file is found, they have to be read, that is, the columns and rows in the excel file has to be read.

```
positive = pd.read_csv(file_name_1, header=0, delimiter=',')
```

```
negative = pd.read_csv(file_name_2, header=0, delimiter=',')
```

Then labels are added; that is, they are mentioned if they are positive or negative. This is made possible using the code given below.

```
positive.insert(23, 'label', 1)
```

```
negative.insert(23, 'label', 0)
```

```
data = pd.concat([positive, negative], ignore_index=True)
```

```
labels = data.label
```

Once the above step is completed, the unnecessary features have to be removed. Unnecessary features include verified, user\_id, screen\_name, label and protected (these features have been explained in user based features).

Then the feature required by the users to run the performance measurements has been done. A sample of this code has been attached below.

```
print('\n\n\t\tResults after performing feature selection\n')
```

```
k = 5 ## mention the number of features the user want to use to run the performance measurements.
```

```
selected_features = feature_selection_with_kbest(data_norm, labels, k)
```

```

print('Best {} Features based on KBest'.format(k))

count = 1

for i in selected_features:

    print '\t', count, '-', data.columns.values[i]

    count += 1

selected_features = sorted(selected_features)

data_selected_features = data_norm[:, selected_features]

clf = svm.SVC(kernel= 'rbf')

result_1 = cross_val_predict(clf, data_selected_features, labels, cv=10)

print('\nSupport Vector Machine')

printResult(labels, result_1)

```

### 5.3. Data Processing Results

The python program was run three times with and without selecting the features to get an accurate value. Moreover, the several numbers of features were selected in order to see which machine learning algorithm perform best to identify the spam accounts in Twitter.

#### 5.3.1. No best features selection results

The observation results received for different performance measurements for the classifiers with no best features selection are provided in table 5.

Table 5: Classifier results before best feature selection

Result Before Best Features					
	Support Vector Machine	Decision Tree	Naive Bayes	K-NN	Random Forest
CV Accuracy	94.84	96.08	82.39	94.05	96.97
CV Precision	95.3	91.69	59.07	92.44	94.31
CV Specificity	98.62	97.19	77.56	97.73	98.11

<b>CV f1</b>	89.04	92.23	73.37	87.49	93.93
<b>CV Recall</b>	83.55	92.77	96.82	83.05	93.56
<b>CV ACU</b>	91.08	94.98	87.19	90.39	95.84
<b>CV Error</b>	5.16	3.92	17.61	5.95	3.03

### 5.3.2. Ten best features selection results

The observation results received for different performance measurements for the classifiers with ten best classifiers selection are provided in table 6.

Table 6: Classifier results after ten best features selection

<b>The result after ten best Features</b>					
	<b>Support Vector Machine</b>	<b>Decision Tree</b>	<b>Naive Bayes</b>	<b>K-NN</b>	<b>Random Forest</b>
<b>CV Accuracy</b>	94.82	95.99	89.33	95.36	96.8
<b>CV Precision</b>	95.19	90.96	71.89	92.73	94.18
<b>CV Specificity</b>	98.59	96.9	87.67	97.68	98.08
<b>CV f1</b>	89	92.09	81.59	90.53	93.57
<b>CV Recall</b>	83.56	93.25	94.3	88.42	92.97
<b>CV auc</b>	91.08	95.08	90.98	93.05	95.52
<b>CV Error</b>	5.18	4.01	10.67	4.64	3.2

### 5.4. Conclusion

This chapter provides the processing code explanation, and the results received for processing the spam and real account.

## 6.0. Data Result Analysis

### 6.1. Introduction

This chapter mainly focuses on providing a detail description regarding the results received for each performance measurement with different feature selection for the classifiers.

### 6.2. Confusion Matrix

The confusion matrix is a table that is mostly used to represent the performance of a specific classification model which can be any algorithms. The confusion matrix for all the classifiers used in this project are provided below:

#### **Support Vector Machine**

[[8718 1717]

[430 30767]]

The machine predicted 8718 accounts as spam accounts and the other accounts about 1717 accounts are spam and they are the wrong prediction. On the other hand, SVM predicted 30767 accounts were not fake accounts, and about 430 accounts had a wrong prediction.

#### **Decision Tree**

[[ 9681 754]

[877 30320]]

The machine predicted 9681 as a spam accounts and the others 754 spam accounts as accounts with the wrong prediction from the Decision Tree algorithm. On the other hand, Decision Tree predicted 30320 accounts were real accounts, and 877 accounts had a wrong prediction from Decision Tree. This prediction explains why the Decision Tree did not give us a high percentage of accuracy

#### **Naive Bayes**

[[10103 332]

[ 7000 24197]]

The machine predicted 10103 as a spam accounts and the others 332 spam accounts have a wrong prediction by using the Naive Bayes algorithm. On the other hand, Naive Bayes predicted 24197 accounts as real accounts, and 7000 accounts had the wrong prediction, this Naive Bayes prediction explains why the Naive Bayes gave us the lowest percentage of accuracy.

#### **K-NN**

[[8666 1769]

[709 30488]]

The machine predicted 8666 as a spam accounts and the other 1769 spam accounts had the wrong prediction by using K-NN algorithm. On the other hand, K-NN predicted a large number of real account which is 30488 accounts and 709 accounts with wrong prediction as explained in K-NN, the K-NN gave the high percentage of accuracy.

**Random Forest**

[[ 9757 678]

[ 583 30614]]

The machine predicted 9757 accounts were spam and just 678 accounts spam accounts with the wrong prediction by using Random Forest algorithm. On the other hand, Random Forest predicted a large number of real accounts which is about 30614. It also identified 583 accounts had a wrong prediction from Random Forest. This prediction gave a high percentage of accuracy.

6.3. Performance Measurement – Accuracy

The accuracy measurement is a ratio between the correct observation over the total observations of the specific dataset. The summary of the accuracy of different algorithms are provided in table 7.

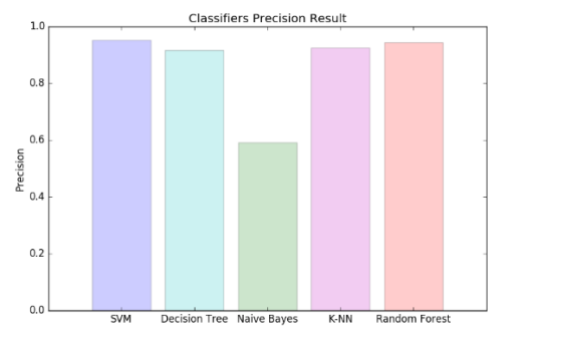
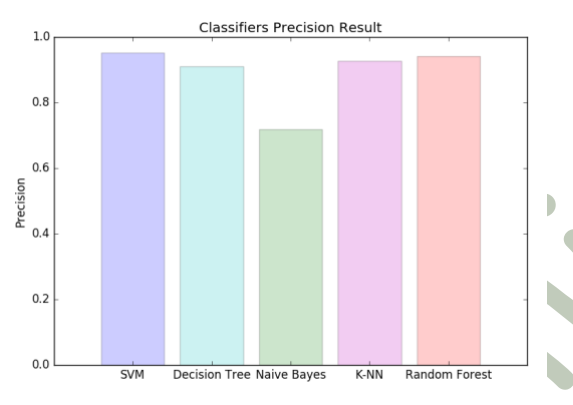
Table 7: Summary accuracy

<table border="1"> <caption>Classifiers Accuracy Result (Before selecting best features)</caption> <thead> <tr> <th>Classifier</th> <th>Accuracy</th> </tr> </thead> <tbody> <tr> <td>SVM</td> <td>0.95</td> </tr> <tr> <td>Decision Tree</td> <td>0.98</td> </tr> <tr> <td>Naive Bayes</td> <td>0.82</td> </tr> <tr> <td>K-NN</td> <td>0.95</td> </tr> <tr> <td>Random Forest</td> <td>0.98</td> </tr> </tbody> </table>	Classifier	Accuracy	SVM	0.95	Decision Tree	0.98	Naive Bayes	0.82	K-NN	0.95	Random Forest	0.98	<p>The results received for accuracy before selecting the best features. The result indicate that Ransom forest classifier has the highest accuracy and the Naïve Bayes classifier has the lowest accuracy.</p>
Classifier	Accuracy												
SVM	0.95												
Decision Tree	0.98												
Naive Bayes	0.82												
K-NN	0.95												
Random Forest	0.98												
<table border="1"> <caption>Classifiers Accuracy Result (After selecting ten best features)</caption> <thead> <tr> <th>Classifier</th> <th>Accuracy</th> </tr> </thead> <tbody> <tr> <td>SVM</td> <td>0.95</td> </tr> <tr> <td>Decision Tree</td> <td>0.98</td> </tr> <tr> <td>Naive Bayes</td> <td>0.88</td> </tr> <tr> <td>K-NN</td> <td>0.95</td> </tr> <tr> <td>Random Forest</td> <td>0.98</td> </tr> </tbody> </table>	Classifier	Accuracy	SVM	0.95	Decision Tree	0.98	Naive Bayes	0.88	K-NN	0.95	Random Forest	0.98	<p>The results received for accuracy after selecting the ten best features. The result indicates that Random forest, and decision tree classifiers have almost the highest accuracy and the Naïve Bayes classifier has the lowest accuracy.</p>
Classifier	Accuracy												
SVM	0.95												
Decision Tree	0.98												
Naive Bayes	0.88												
K-NN	0.95												
Random Forest	0.98												

#### 6.4. Performance Measurement – Precision

The precision measurement is a ratio between the properly predicted positive observation and the total positive predicted observations. The summary precision different algorithms are provided in table 8.

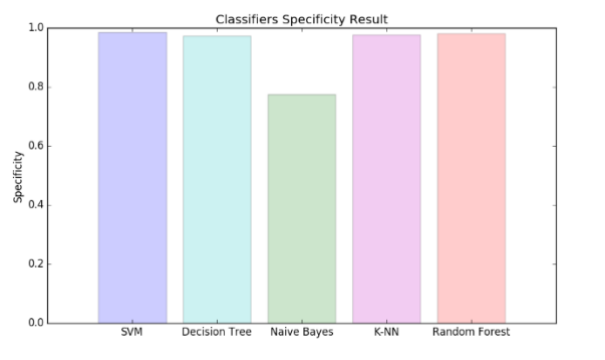
Table 8: Summary precision

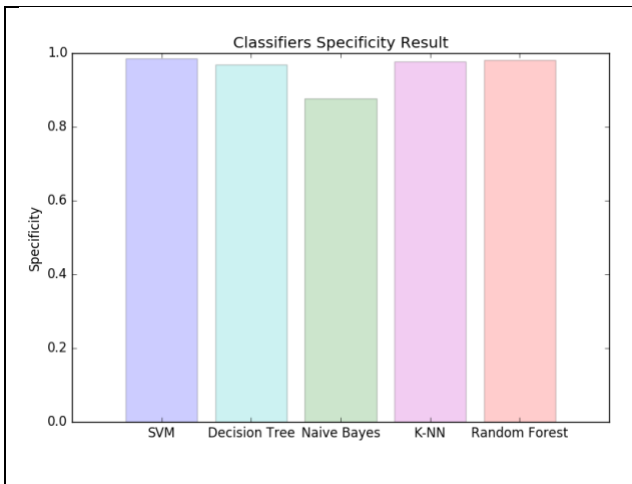
 <table border="1"> <caption>Classifiers Precision Result</caption> <thead> <tr> <th>Classifier</th> <th>Precision</th> </tr> </thead> <tbody> <tr> <td>SVM</td> <td>0.95</td> </tr> <tr> <td>Decision Tree</td> <td>0.92</td> </tr> <tr> <td>Naive Bayes</td> <td>0.60</td> </tr> <tr> <td>K-NN</td> <td>0.93</td> </tr> <tr> <td>Random Forest</td> <td>0.94</td> </tr> </tbody> </table>	Classifier	Precision	SVM	0.95	Decision Tree	0.92	Naive Bayes	0.60	K-NN	0.93	Random Forest	0.94	<p>The results received for accuracy before selecting the best features. The result indicates that Random forest, and SVM classifiers have almost the highest precision and the Naïve Bayes classifier has the lowest accuracy.</p>
Classifier	Precision												
SVM	0.95												
Decision Tree	0.92												
Naive Bayes	0.60												
K-NN	0.93												
Random Forest	0.94												
 <table border="1"> <caption>Classifiers Precision Result</caption> <thead> <tr> <th>Classifier</th> <th>Precision</th> </tr> </thead> <tbody> <tr> <td>SVM</td> <td>0.95</td> </tr> <tr> <td>Decision Tree</td> <td>0.92</td> </tr> <tr> <td>Naive Bayes</td> <td>0.72</td> </tr> <tr> <td>K-NN</td> <td>0.93</td> </tr> <tr> <td>Random Forest</td> <td>0.94</td> </tr> </tbody> </table>	Classifier	Precision	SVM	0.95	Decision Tree	0.92	Naive Bayes	0.72	K-NN	0.93	Random Forest	0.94	<p>The results received for accuracy before selecting the ten best features. The result indicates that Random forest, and SVM classifiers have almost the highest precision and the Naïve Bayes classifier has the lowest precision.</p>
Classifier	Precision												
SVM	0.95												
Decision Tree	0.92												
Naive Bayes	0.72												
K-NN	0.93												
Random Forest	0.94												

#### 6.5. Performance Measurement – Specificity

The specificity measurement is a ratio between the total number of true negative observation over the total number of true negative and false positive observations. The summary specificity of different algorithms is provided in table 9.

Table 9: Summary Specificity

 <table border="1"> <caption>Classifiers Specificity Result</caption> <thead> <tr> <th>Classifier</th> <th>Specificity</th> </tr> </thead> <tbody> <tr> <td>SVM</td> <td>0.98</td> </tr> <tr> <td>Decision Tree</td> <td>0.97</td> </tr> <tr> <td>Naive Bayes</td> <td>0.78</td> </tr> <tr> <td>K-NN</td> <td>0.97</td> </tr> <tr> <td>Random Forest</td> <td>0.97</td> </tr> </tbody> </table>	Classifier	Specificity	SVM	0.98	Decision Tree	0.97	Naive Bayes	0.78	K-NN	0.97	Random Forest	0.97	<p>The results received for specificity before selecting the best features. The result indicates that the SVM classifier has almost the highest specificity and the Naïve Bayes classifier has the lowest specificity.</p>
Classifier	Specificity												
SVM	0.98												
Decision Tree	0.97												
Naive Bayes	0.78												
K-NN	0.97												
Random Forest	0.97												

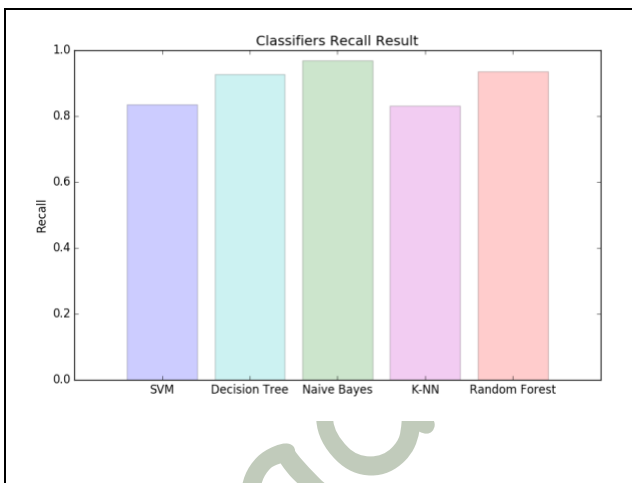


The results received for specificity after selecting the ten best features. The result indicates that the SVM classifier has almost the highest specificity and the Naïve Bayes classifier has the lowest specificity.

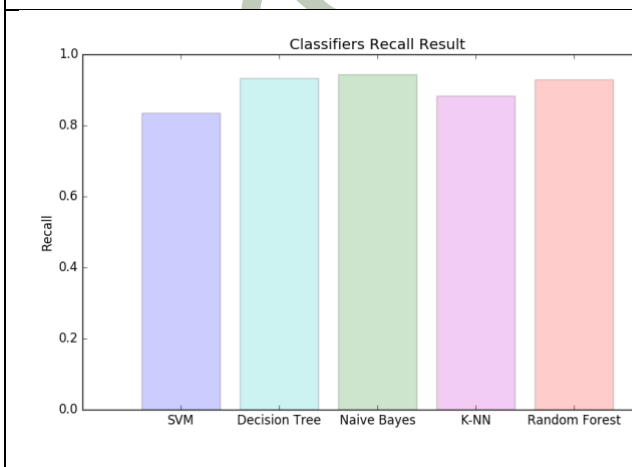
### 6.6. Performance Measurement -Recall

The recall measurement is a ratio between the predicted positive observation over all the observations in actual yes class. The summary recall different algorithms are provided in table 10.

Table 10: Summary Recall



The results received for recall before selecting the best features. The result indicates that Naïve Bayes classifier has almost the highest recall and the K-NN classifier has the lowest

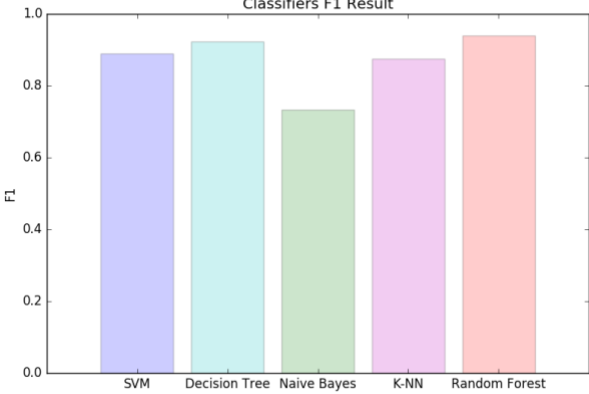
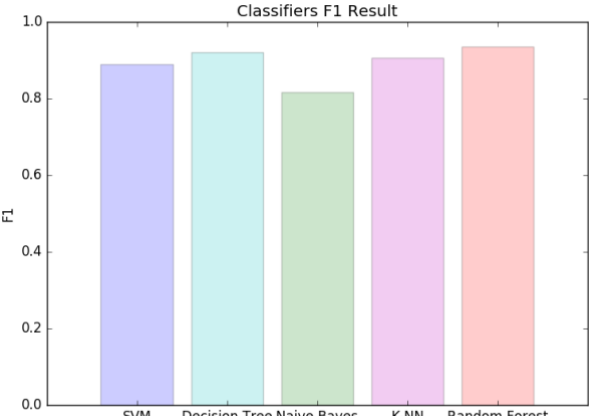


The results received for specificity after selecting the ten best features. The result indicates that Naïve Bayes classifier has almost the highest recall and the SVM classifier has the lowest recall.

### 6.7. Performance Measurement – F1 (F-measure)

The F-measure is an average between the recall and precision. The summary F1 different algorithms are provided in table 11.

Table 11: Summary F1

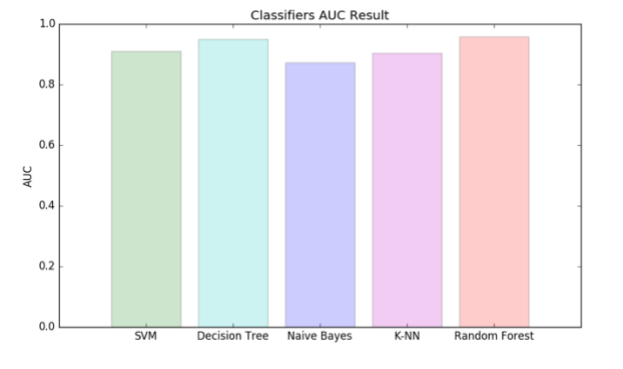
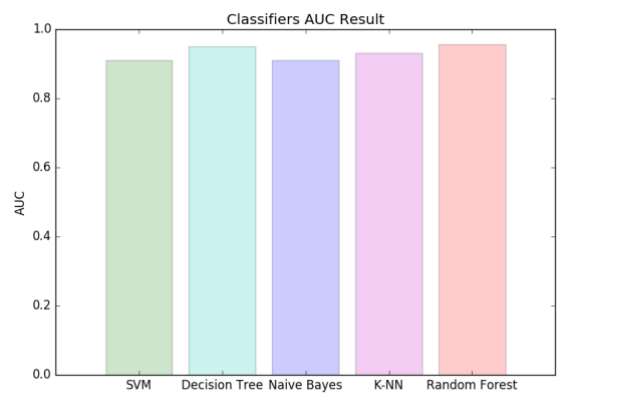
 <table border="1"><thead><tr><th>Classifier</th><th>F1 Score</th></tr></thead><tbody><tr><td>SVM</td><td>0.88</td></tr><tr><td>Decision Tree</td><td>0.92</td></tr><tr><td>Naive Bayes</td><td>0.75</td></tr><tr><td>K-NN</td><td>0.88</td></tr><tr><td>Random Forest</td><td>0.95</td></tr></tbody></table>	Classifier	F1 Score	SVM	0.88	Decision Tree	0.92	Naive Bayes	0.75	K-NN	0.88	Random Forest	0.95	<p>The results received for F1 before selecting the best features. The result indicates that the Random Forest classifier has almost the highest F1 and the Naïve Bayes classifier has the lowest</p>
Classifier	F1 Score												
SVM	0.88												
Decision Tree	0.92												
Naive Bayes	0.75												
K-NN	0.88												
Random Forest	0.95												
 <table border="1"><thead><tr><th>Classifier</th><th>F1 Score</th></tr></thead><tbody><tr><td>SVM</td><td>0.89</td></tr><tr><td>Decision Tree</td><td>0.92</td></tr><tr><td>Naive Bayes</td><td>0.81</td></tr><tr><td>K-NN</td><td>0.91</td></tr><tr><td>Random Forest</td><td>0.95</td></tr></tbody></table>	Classifier	F1 Score	SVM	0.89	Decision Tree	0.92	Naive Bayes	0.81	K-NN	0.91	Random Forest	0.95	<p>The results received for F1 after selecting the ten best features. The result indicates that Random forest classifier has almost the highest F1 and the Naïve Bayes classifier has the lowest F1.</p>
Classifier	F1 Score												
SVM	0.89												
Decision Tree	0.92												
Naive Bayes	0.81												
K-NN	0.91												
Random Forest	0.95												

### 6.8. Performance Measurement – AUC

The AUC measurement is an evaluation of the classifier as a threshold. The summary AUC different algorithms are provided in table 12.




Table 12: Summary AUC

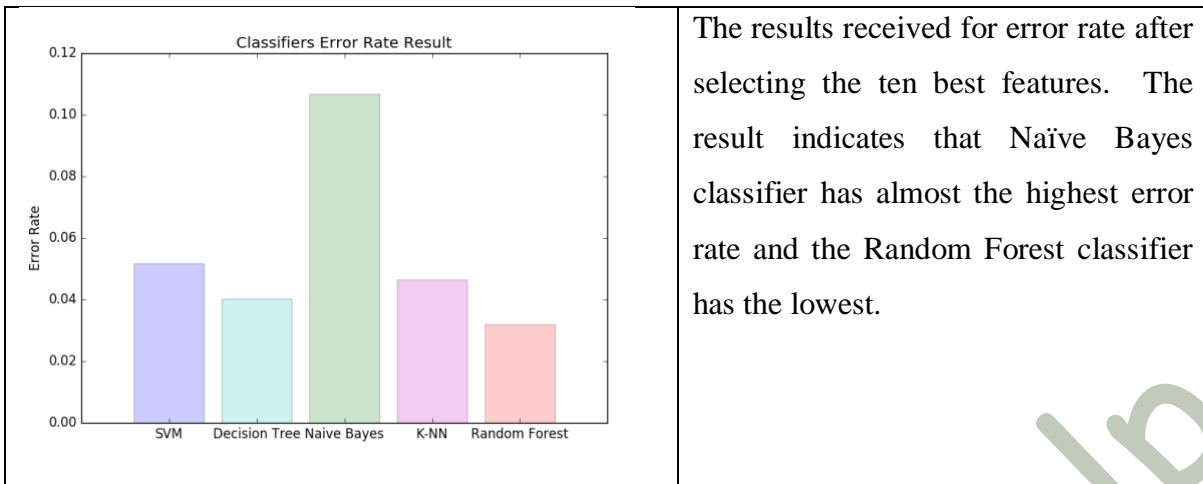
	<p>The results received for AUC before selecting the best features. The result indicates that the Random Forest classifier has almost the highest AUC and the Naïve Bayes classifier has the lowest</p>
	<p>The results received for AUC after selecting the ten best features. The result indicates that Random forest classifier has almost the highest AUC and the SVM classifier has the lowest AUC.</p>

### 6.9. Performance Measurement – Error Rate

The ERR measurement is computed by the ratio of a total number of incorrect observation over the total number of observations in the dataset. The summary error rate different algorithms are provided in table 13.

Table 13: Summary Error Rate

	<p>The results received for error rate before selecting the best features. The result indicates that Naïve Bayes classifier has almost the highest error rate and the Random Forest classifier has the lowest</p>
---	---



The results received for error rate after selecting the ten best features. The result indicates that Naïve Bayes classifier has almost the highest error rate and the Random Forest classifier has the lowest.

### 6.10. Conclusion

From the above analysis, it is reasonable to say that the Twitter cannot use a single machine learning algorithm to identify the spam and real account. There are three major categories that Twitter can use which is to select and not select any best features. In this project, the best ten features selection were used. However, the appropriate result can be achieved by selecting more features. The final finding summary for the best machine learning algorithm for different experiments is provided in table 14.

Table 14: Summary of Algorithms results

	No best feature Selection	Ten best feature Selection
CV Accuracy	Random Forest	Random Forest
CV Precision	SVM	SVM
CV Specificity	SVM	SVM
CV f1	Random Forest	Random Forest
CV Recall	Naïve Bayes	Naïve Bayes
CV AUC	Random Forest	Random Forest
CV Error	Random Forest	Random Forest

From the above table, it is reasonable to say Twitter should use SVM if their requirement is Precision and specificity. On the other hand, for accuracy, f1, AUC and error rate, the Random forest algorithm can be used. Besides, if the priority is a recall, then Naïve Bayes is an appropriate machine learning algorithm.

However, in general, it is reasonable to use Random forest that will provide accurate spam and real account classification compared to other machine learning algorithm for Twitter.

## 7.0. Discussion

The final results from the Twitter data analysis to identify the spam and real accounts are that random forest machine learning algorithm provides an appropriate solution compared to other algorithms. The best ten features selected from the code to identify the real and spam account are Favorite\_Count, default\_profile\_image, followers\_count, profile\_use\_background\_image, retweet\_count, friends\_count, total\_mentions, notifications, total\_hashtags and sample\_tweet. Most of the identified best features from the dataset have unique information which can be used to classify a specific account to spam or real account. The features that can individually categorise the account to spam or real account are profile\_use\_background\_image, retweet\_count, total\_mentions, notifications, and total\_hashtags.

The discussion on whether the best feature selection from the python code is provided in table 15.

Table 15: Discussion of best feature selection

Best Features	Discussion
Favorite_Count	This feature is agreed to be the best feature because the spam accounts have a small number or even zero for favorite_count and real accounts has a large number for favorite_count. See figure 5 and 6 for the yellow coloured column. This feature tells us that spam accounts always have zero Favourite counts or a small number of Favourite count which indicates spam accounts do not interact with other accounts since they are controlled by the machine. Their main goal is to post what the machine control in tweet dataset
default_profile_image	This feature is agreed to be the best feature because the default provided image be 1 for spam account and 0 for real account. See figure 5 and 6 for the light blue coloured column. This feature supports images, but a spam account generally does not have a profile image because its goal is to tweet from the existing dataset
followers_count	This feature is agreed to be the best feature because the followers count is a tiny number for spam account and a large number of real accounts. See figure 5 and 6 for the dark blue coloured column. This feature tells us that spam accounts do

Best Features	Discussion
	not have followers. This is because people are not interested in their tweets since they tweet links and advertisement
profile_use_background_image	This feature is agreed to be the best feature because the profile use background image is 1 for all the spam account, but it can be 1 or 0 for real account. Even though it is helpful, it cannot be used alone to identify the twitter account, whether it is real or spam. See figure 5 and 6 for the orange coloured column. This feature ensures spam accounts since a machine controller does not care about account profile details. They only care to tweet and post what they want.
retweet_count	This feature is agreed to be the best feature because the retweet count is always high for all the real account, but it can be high or low for spam account (mostly low or even zero). Even though it is helpful, it cannot be used alone to identify the twitter account whether it is real or spam. See figure 5 and 6 for the purple coloured column. This feature ensures the spam accounts generally because they do not retweet from other accounts
friends_count	This feature is agreed to be the best feature because the friend count is a very small number for spam account and a large number for real accounts. See figure 5 and 6 for the red coloured column. This feature tells us how the accounts interact with each other; spam accounts generally, do not have friends or followers
total_mentions	This feature is agreed to be the best feature because the total mentions are always high for all the real account, but it can be high or low for spam account (mostly low or even zero). Even though it is helpful, it cannot be used alone to identify the twitter account whether it is real or spam. See figure 5 and 6 for the light green coloured column. This feature tells us about spam accounts' interaction on tweeter; spam accounts usually do not have to mentions since spam does not usually interact with other accounts

Best Features	Discussion
notifications	<p>This feature is agreed to be the best feature because the notification is 0 for all the spam account, but it can be 1 or 0 for real account. Even though it is helpful, it cannot be used alone to identify the Twitter account, whether it is real or spam. See figure 5 and 6 for the dark green coloured column. This feature tells us about spam account activities on Twitter; spam account controller does not need the notification feature since it is not related to its goal. They tweet and spread what they want to advertise or fake news</p>
total_hashtags	<p>This feature is agreed to be the best feature because the total hashtags are more for real account compared to spam account. Even though it is helpful, it cannot be used alone to identify the twitter account whether it is real or spam. See figure 5 and 6 for the dark red coloured column.</p> <p>This feature tells how spam accounts act on Twitter. Spam accounts usually have a large number of hashtags in their tweets since they try to reach a large number of audiences to have followers or friends and usually they will be tweeted in more than one hashtags to reach more audience</p>
sample_tweet	<p>This feature is agreed to be the best feature because the sample tweet is always high for all the real account, but it is low for a spam account. See figure 5 and 6 for the ash coloured column. This feature tells us how spam accounts act in Twitter generally, where spam accounts have a small number of simple tweet. This feature links with the hashtag feature. When comparing it together, it will tell us accounts interaction on Twitter if posting Malicious links or fake news or advertising.</p>

Figure 5: Spam Account Twitter Data

Figure 6: Real account Twitter Data

The statistical analysis for the best features are provided in table 16.

Table 16: Statistical analysis for the best features

Best Features	Count		Mean		std		min		max	
	Real	Spam	Real	Spam	Real	Spam	Real	Spam	Real	Spam
Favorite_Count	31197	10435	10182.2	8.2	246806.1	58.6	0	0	17309891	2550
default_profile_image	31197	10435	0	0.8	0.1	0.3	0	0	1	1
followers_count	31197	10435	2.4e+05	49.1	2.5e+06	1707.6	1	0	1.0e+08	100116
profile_use_background_image	31197	10435	0.9	0.9	0.2	0.1	0	0	1	1
retweet_count	31197	10435	51434.3	3227.7	218058.2	48615.8	0	0	6520907	3660257
friends_count	31197	10435	24595.6	73.1	106649.3	554.6	0	0	2331058	30285
total_mentions	31197	10435	71.4	5.3	52.9	20.3	0	0	1303	250
notifications	31197	10435	0	0	0	0	0	0	1	0
total_hashtags	31197	10435	44.9	25.31	62.2	75	0	0	915	421
sample_tweet	31197	10435	89.2	12.3	26.8	27.2	0	0	100	100

Currently, it is identified that the users are more concerned with spam accounts because there is a high increase in spam account in the Twitter platform. From the table 19, it is possible to say that 41632 twitter accounts were used to analyse and 74.9% of the accounts are real, while 25.1% of the account is spam. This is a very high percentage. Therefore, it is clear that Twitter has to do some immediate action to identify the spam accounts and block them. Therefore, this project suggests the random forest machine learning algorithm can be appropriate for the identifying the spam accounts.

From the results received, it is identified that the SVM algorithm performs better for precision and specificity. On the other hand, the Naïve Bayes algorithm can be used for recall measurement. Besides, the random forest can be used for the accuracy, f-measures, AUC, and Error Rate. From the results received, the random forest is one of the most accurate algorithms available which produces the high accuracy results to identify whether the account is spam for most of the datasets which includes the twitter account details dataset. Moreover, the efficiency of the random forest is more for the large dataset and can handle several variables without deletion. Additionally, the random forest can provide the list of features that are important for the classification. Besides, the random forest can provide the estimate of the missing data in the dataset.

### 7.1. How objectives are achieved

The discussion of how objectives are achieved is provided in table 17.

Table 17: object achievement

Objectives	Discussion
Objective 1: Perform secondary research to identify the similar methods used to identify the Twitter Spam accounts.	Some of the methods to identified from the literature to categorise the spam account from the real account are an account-based, tweet-based, graph-based and hybrid method.
Objective 2: Use Twitter API to collect row account details and clean up the data to perform the analysis.	The data collection using Twitter API was done using the python program. The explanation of the code used is provided in section 4. Section 4 also discusses the steps involved categorisation of data collected.
Objective 3: Perform analysis of the clean data using binary classification methods such as Support Vector Machine (SMV), Naïve Bayes, K-NN, and Decision Tree, and Random Forest.	Section 5 provides the explanation of the python code written to run the classifiers for the spam and real account excel sheet.

Objective 4: Compare the analysis and suggest the binary classifier that can be used to detect the spam account in the Twitter.	The analysis suggests that the random forest is an appropriate classifier to identify spam accounts and real accounts (See section 6)
---	---

## 8.0. Conclusion and Future works

This project retrieved 41632 twitter account information using twitter API and manually the spam and real accounts were classified. Then the python code was used to run the machine learning algorithms which are support vector machine (SVM), decision tree, Naïve Bayes, K-NN and Random forest to find out which algorithm is the suitable for Twitter to identify the spam and real accounts. The analysis indicates that 74.1% of the accounts are real and 25.1% of the account is spam. From the results received, it is identified that the random forest algorithm provides more accurate results compared to other algorithms. Moreover, the python program identified ten best features that can be used to identify the account, whether it is spam or not. The best features identified are Favorite\_Count, default\_profile\_image, followers\_count, profile\_use\_background\_image, retweet\_count, friends\_count, total\_mentions, notifications, total\_hashtags and sample\_tweet.

### Future works

The future works that are suggested for this project are provided below:

- Collect more twitter account details using Twitter API.
- Run the Classification algorithm more than once to take an accurate value for the performance measurement.
- Create an excel model that will automatically classify the collected account into real and spam accounts.



## 9.0. Project Management

### 9.1. Risk Table

The risk identified and how it was avoided is also provided in table 16.

Table 18: Risk table

<b>Risk</b>	<b>Impact Level</b>	<b>How is was avoided</b>
Not be able to complete the project to meet the deadline.	High	The strict timeframe was used to track the progress of the project.
Writing python program is challenging	High	The online sources are used to study python programming
Collecting Twitter account details per day is only 1000	High	This can increase the time frame of data collection so; different twitter accounts were used to collect data.

### 9.2. Gantt Chart

See Appendix A for the Gantt Chart.

## Glossary

**SVM:** The Support Vector Machine is mapped in an exorbitant dimensional input space using this approach and then designs an ideal disjointed hyperplane within this expanse.

**Decision Tree:** Tree Classifier acts as an array of meticulously drafted enquiries concerning the features of the test record

**Naïve Bayes:** Naïve Bayes classifiers in machine learning can be regarded as a group of easy probabilistic classifiers founded on the usage of Bayes' hypothesis along with strong (naive) freedom presumptions betwixt the characters.

**K-NN:** K-Nearest Neighbour can likewise be referred to as a lazy learning classifier. Decision tree and rule-based classifiers are created to understand a prototype that charts the data characteristics to the class label immediately the tutoring information is ready. Therefore they are regarded as eager learning classifiers.

**Random Forest:** Random forests or random decision forests are an ensemble learning method for classification, regression including various operations, which utilize by setting up a large number of decision trees at learning duration and outputting the class which is the approach of the mean prediction (regression) or classes (classification) of the separate trees.

**Accuracy:** It is the rate of accurately classified tags on every forecast

**Precision:** The accuracy that is part of the most popularly utilized performance measure is the Precision.

**Recall:** Recall is the entirety that is likewise defined as responsiveness or verifiable affirmative ratio.

**Specificity:** The quantity of true inaccurate forecasts split by the aggregate amount of inaccurate is used to estimate Specificity.

**Error Rate:** Error rate is calculated as the number of all incorrect predictions divided by the total number of the dataset.

**AUC:** Area Under the Curve (AUC) can be calculated by performing a definite integral between the given two points in the graph.

AcademicianHelp

## References

- Huang, X., Shi, L. and Suykens, J.A., 2014. Asymmetric least squares support vector machine classifiers. *Computational Statistics & Data Analysis*, 70, pp.395-405.
- Vapnik, V., 1995. *The Nature of Statistical Learning*. Springer
- Bui, D.T., Ho, T.C., Revhaug, I., Pradhan, B. and Nguyen, D.B., 2014. Landslide susceptibility mapping along the national road 32 of Vietnam using GIS-based J48 decision tree classifier and its ensembles. In *Cartography from pole to pole* (pp. 303-317). Springer Berlin Heidelberg.
- Patil, T.R. and Sherekar, S.S., 2013. Performance analysis of Naive Bayes and J48 classification algorithm for data classification. *International Journal of Computer Science and Applications*, 6(2), pp.256-261.
- Bidder, O.R., Campbell, H.A., Gómez-Laich, A., Urgé, P., Walker, J., Cai, Y., Gao, L., Quintana, F. and Wilson, R.P., 2014. Love thy neighbour: automatic animal behavioural classification of acceleration data using the k-nearest neighbour algorithm. *PloS one*, 9(2), p.e88609.
- Ellis, K., Kerr, J., Godbole, S., Lanckriet, G., Wing, D. and Marshall, S., 2014. A random forest classifier for the prediction of energy expenditure and type of physical activity from wrist and hip accelerometers. *Physiological measurement*, 35(11), p.2191.
- Japkowicz, N. (2008). Classifier evaluation: A need for better education and restructuring. In *Proceedings of the 3rd workshop on evaluation methods for machine learning*.
- Powers, D.M., 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.
- A.H. Wang, Don't follow me: Spam detection in Twitter, in: *SECRYPT 2010 - Proc. Int. Conf. Secur. Cryptogr.*, Athens, Greece, 2010: pp. 1–10. doi:978-989-8425-18-8.
- Amlshwaram, A.A., Reddy, N., Yadav, S., Gu, G. and Yang, C., 2013, January. Cats: Characterizing automation of twitter spammers. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on* (pp. 1-10). IEEE.
- Arun, K., Srinagesh, A. and Ramesh, M., 2017. Twitter Sentiment Analysis on Demonetization tweets in India Using R language.
- Atefeh, F. and Khreich, W., 2015. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1), pp.132-164.
- Bai, Y., 2017. Tweets Win Votes: A Persuasive Communication Perspective on Donald Trump's Twitter Use During the 2016 US Presidential Election Campaign.

- Benevenuto, F., Magno, G., Rodrigues, T. and Almeida, V., 2010, July. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)* (Vol. 6, No. 2010, p. 12).
- Bilge, L., Strufe, T., Balzarotti, D. and Kirda, E., 2009, April. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web* (pp. 551-560). ACM.
- Bravo-Marquez, F., Mendoza, M. and Poblete, B., 2013, August. Combining strengths, emotions and polarities for boosting Twitter sentiment analysis. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining* (p. 2). ACM.
- Canali, D., Cova, M., Vigna, G. and Kruegel, C., 2011, March. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *Proceedings of the 20th international conference on World wide web* (pp. 197-206). ACM.
- Chakraborty, A., Sundi, J. and Satapathy, S., 2012. SPAM: a framework for social profile abuse monitoring. *CSE508 report, Stony Brook University, Stony Brook, NY.*
- Chen, C., Zhang, J., Chen, X., Xiang, Y. and Zhou, W., 2015, June. 6 million spam tweets: A large ground truth for timely Twitter spam detection. In *Communications (ICC), 2015 IEEE International Conference on* (pp. 7065-7070). IEEE.
- Chen, C., Zhang, J., Chen, X., Xiang, Y. and Zhou, W., 2015, June. 6 million spam tweets: A large ground truth for timely Twitter spam detection. In *Communications (ICC), 2015 IEEE International Conference on* (pp. 7065-7070). IEEE.
- Chen, C., Zhang, J., Xie, Y., Xiang, Y., Zhou, W., Hassan, M.M., AlElaiwi, A. and Alrubaian, M., 2015. A performance evaluation of machine learning-based streaming spam tweets detection. *IEEE Transactions on Computational Social Systems*, 2(3), pp.65-76.
- Chhabra, S., Aggarwal, A., Benevenuto, F. and Kumaraguru, P., 2011, September. Phi. sh/\$ ocial: the phishing landscape through short urls. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference* (pp. 92-101). ACM.
- Chu, Z., Gianvecchio, S., Wang, H. and Jajodia, S., 2010, December. Who is tweeting on Twitter: human, bot, or cyborg?. In *Proceedings of the 26th annual computer security applications conference* (pp. 21-30). ACM.
- Cova, M., Kruegel, C. and Vigna, G., 2010, April. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proceedings of the 19th international conference on World wide web* (pp. 281-290). ACM.
- Debatin, B., Lovejoy, J.P., Horn, A.K. and Hughes, B.N., 2009. Facebook and online privacy: Attitudes, behaviors, and unintended consequences. *Journal of Computer-Mediated Communication*, 15(1), pp.83-108.

- Gabielkov, M. and Legout, A., 2012, December. The complete picture of the Twitter social graph. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop* (pp. 19-20). ACM.
- Gao, H., Chen, Y., Lee, K., Palsetia, D. and Choudhary, A.N., 2012, February. Towards online spam filtering in social networks. In *NDSS* (Vol. 12, pp. 1-16).
- Gee, G. and Teh, H., 2010. Twitter spammer profile detection.
- Jagatic, T.N., Johnson, N.A., Jakobsson, M. and Menczer, F., 2007. Social phishing. *Communications of the ACM*, 50(10), pp.94-100.
- Kaur, P., Singhal, A. and Kaur, J., 2016, March. Spam detection on Twitter: A survey. In *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on* (pp. 2570-2573). IEEE.
- Kim, D., Jo, Y., Moon, I.C. and Oh, A., 2010, April. Analysis of twitter lists as a potential source for discovering latent characteristics of users. In *ACM CHI workshop on microblogging* (p. 4).
- Kwak, H., Lee, C., Park, H. and Moon, S., 2010, April. What is Twitter, a social network or a news media?. In *Proceedings of the 19th international conference on World wide web* (pp. 591-600). ACM.
- Lee, K., Caverlee, J. and Webb, S., 2010, July. Uncovering social spammers: social honeypots+ machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*(pp. 435-442). ACM.
- Lee, S. and Kim, J., 2013. Warningbird: A near real-time detection system for suspicious urls in twitter stream. *IEEE transactions on dependable and secure computing*, 10(3), pp.183-195.
- Lin, P.C. and Huang, P.M., 2013, January. A study of effective features for detecting long-surviving Twitter spam accounts. In *Advanced Communication Technology (ICACT), 2013 15th International Conference on* (pp. 841-846). IEEE.
- Ma, J., Saul, L.K., Savage, S. and Voelker, G.M., 2009, June. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1245-1254). ACM.
- Martinez-Romo, J. and Araujo, L., 2013. Detecting malicious tweets in trending topics using a statistical analysis of language. *Expert Systems with Applications*, 40(8), pp.2992-3000.
- Mccord, M. and Chuah, M., 2011, September. Spam detection on twitter using traditional classifiers. In *international conference on Autonomic and trusted computing* (pp. 175-186). Springer, Berlin, Heidelberg.
- Mccord, M. and Chuah, M., 2011, September. Spam detection on twitter using traditional classifiers. In *international conference on Autonomic and trusted computing* (pp. 175-186). Springer, Berlin, Heidelberg.
- Mccord, M. and Chuah, M., 2011, September. Spam detection on twitter using traditional classifiers. In *international conference on Autonomic and trusted computing* (pp. 175-186). Springer, Berlin, Heidelberg.

- McGrath, D.K. and Gupta, M., 2008. Behind Phishing: An Examination of Phisher Modi Operandi. *LEET*, 8, p.4.
- Myers, S.A., Sharma, A., Gupta, P. and Lin, J., 2014, April. Information network or social network?: the structure of the twitter follow graph. In *Proceedings of the 23rd International Conference on World Wide Web* (pp. 493-498). ACM.
- Perveen, N., Missen, M.M.S., Rasool, Q. and Akhtar, N., 2016. Sentiment Based Twitter Spam Detection. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 7(7), pp.568-573.
- Seward, Z.M., 2014. Twitter Admits that as Many as 23 Million of its Active Users are Automated.
- Song, J., Lee, S. and Kim, J., 2011. Spam filtering in twitter using sender-receiver relationship. In *Recent advances in intrusion detection* (pp. 301-317). Springer Berlin/Heidelberg.
- Song, J., Lee, S. and Kim, J., 2011. Spam filtering in twitter using sender-receiver relationship. In *Recent advances in intrusion detection* (pp. 301-317). Springer Berlin/Heidelberg.
- Stringhini, G., Kruegel, C. and Vigna, G., 2010, December. Detecting spammers on social networks. In *Proceedings of the 26th annual computer security applications conference* (pp. 1-9). ACM.
- Stringhini, G., Kruegel, C. and Vigna, G., 2010, December. Detecting spammers on social networks. In *Proceedings of the 26th annual computer security applications conference* (pp. 1-9). ACM.
- Stringhini, G., Kruegel, C. and Vigna, G., 2010, December. Detecting spammers on social networks. In *Proceedings of the 26th annual computer security applications conference* (pp. 1-9). ACM.
- Thomas, K., Grier, C., Ma, J., Paxson, V. and Song, D., 2011, May. Design and evaluation of a real-time url spam filtering service. In *Security and Privacy (SP), 2011 IEEE Symposium on* (pp. 447-462). IEEE.
- Ugander, J., Karrer, B., Backstrom, L. and Marlow, C., 2011. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*.
- Verma, M. and Sofat, S., 2014. Techniques to detect spammers in twitter-a survey. *International Journal of Computer Applications*, 85(10).
- Wang, A.H., 2010, July. Don't follow me: Twitter spam detection. In *Proc. 5th International Conf. on Security and Cryptography (SECRYPT)*.
- Wang, A.H., 2010, July. Don't follow me: Spam detection in twitter. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on* (pp. 1-10). IEEE.
- Wang, A.H., 2010, July. Don't follow me: Spam detection in twitter. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on* (pp. 1-10). IEEE.

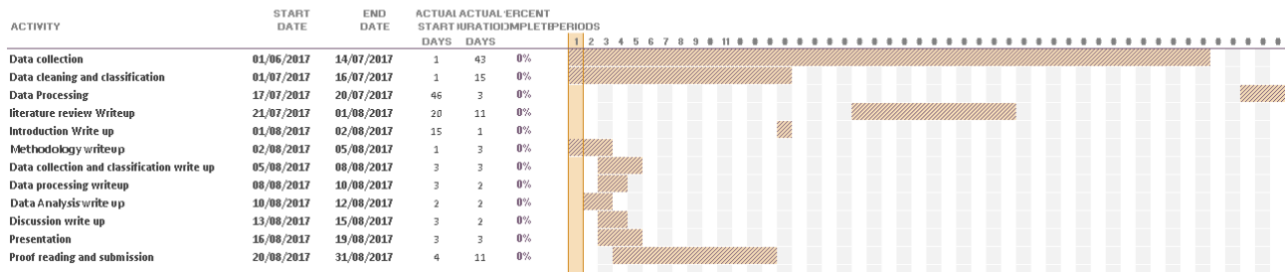
- Wang, B., Zubiaga, A., Liakata, M. and Procter, R., 2015. Making the most of tweet-inherent features for social spam detection on twitter. *arXiv preprint arXiv:1503.07405*.
- Wang, Y.M., Beck, D., Jiang, X., Roussev, R., Verbowski, C., Chen, S. and King, S., 2006, February. Automated web patrol with strider honeymonkeys. In *Proceedings of the 2006 Network and Distributed System Security Symposium* (pp. 35-49).
- Waters, D., 2009. Spam overwhelms e-mail messages. BBC.
- Weaver, J. and Tarjan, P., 2013. Facebook linked data via the graph API. *Semantic Web*, 4(3), pp.245-250.
- Whittaker, C., Ryner, B. and Nazif, M., 2010, February. Large-Scale Automatic Classification of Phishing Pages. In *NDSS* (Vol. 10, p. 2010).
- Yamaguchi, Y., Amagasa, T. and Kitagawa, H., 2011, July. Tag-based user topic discovery using twitter lists. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on* (pp. 13-20). IEEE.
- Yang, C., Harkreader, R. and Gu, G., 2013. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8(8), pp.1280-1293.
- Yardi, S., Romero, D. and Schoenebeck, G., 2009. Detecting spam in a twitter network. *First Monday*, 15(1).



# APPENDIX A Gantt Chart

## Gantt Chart

Period High 1



AcademicianHe

AcademicianHelp

## APPENDIX C Twitter API connection

### config.py

```
consumer_key = "xxxxxxxxxxxxxxxxxxxxx"
consumer_secret = "xxxxxxxxxxxxxxxxx"
access_token = "xxxxxxxxxx-xxxxxxxxxxxxxxxxx"
access_token_secret = "xxxxxxxxxxxxxxxxx"
```

### get-twitter-information.py

```
import time
import tweepy
import pandas as pd
import csv

config = {}
exec(open("config.py").read(), config)
## Intial twitter authentication #
auth = tweepy.OAuthHandler(config["khWZKMC74kvOe8qnZGVXuRZ5g"],
config["7zXwNQYpMxgAXQFL7DBa2sthFiVN2UdjC2QDj0iIK6z3MXa0A0"])
auth.set_access_token(config["836672321871998977-pJO6A5UCwKrtynzK9zolgGg90jGW2LC"],
config["cbbW267cHuOKxqIojKYtFEYWMe5q4qnpypRrDq6fHwRm"])
api = tweepy.API(auth)

fileName="oth_Data.csv"
twitter_account = ''
twitter_ids = []
fields = []
reading_from_file=0 ## 0 for getting user ids from an account in twitter,
if reading_from_file == 1:
    fields = pd.read_csv(fileName, header=0, delimiter=',')
    if fields.columns == "user_id":
        print("Correct header")
    else:
        print("Fixing Header to \"user_id\"")
        fields.columns = ["user_id"]
    twitter_ids = fields.user_id
else:
    twitter_ids = api.friends_ids(twitter_account)
    fields = pd.DataFrame(twitter_ids)
    fields.columns = ["user_id"]

twitter_ids_userName = []
twitter_ids_verified = []
twitter_ids_default_profile = []
twitter_ids_default_profile_image = []
twitter_ids_favourites_count = []
twitter_ids_followers_count = []
twitter_ids_geo_enabled = []
twitter_ids_notifications = []
```

```

twitter_ids_time_zone = []
twitter_ids_location = []
twitter_ids_listed_count = []
twitter_ids_created_at = []
twitter_ids_friends_count = []
twitter_ids_profile_use_background_image = []
twitter_ids_statuses_count = []
twitter_ids_tweets_sample = []
twitter_ids_number_of_hashtags = []
twitter_ids_number_of_links = []
twitter_ids_number_of_mentions = []
twitter_ids_number_of_fav_count = []
twitter_ids_number_of_retweet_count = []
for id in twitter_ids:
    try:
        ## getting user information
        user = api.get_user(id)
        screen_name = user.screen_name
        print("Processing: ", screen_name)
        verified = user.verified
        default_profile = user.default_profile
        default_profile_image = user.default_profile_image
        favourites_count = user.favourites_count
        followers_count = user.followers_count
        geo_enabled = user.geo_enabled
        notifications = user.notifications
        time_zone = user.time_zone
        location = user.location
        listed_count = user.listed_count
        created_at = user.created_at
        friends_count = user.friends_count
        use_background_image = user.profile_use_background_image
        statuses_count = user.statuses_count

        ## getting tweets information
        timeline = api.user_timeline(screen_name = user.screen_name, count = 100, include_rts
= True)

        number_of_hashtags=0
        number_of_links=0
        number_of_mentions=0
        number_of_fav_count = 0
        number_of_retweet_count = 0
        i=0
        for tweet in timeline:
            i += 1
            number_of_hashtags += len(tweet.entities['hashtags'])
            number_of_links += len(tweet.entities['urls'])
            number_of_mentions += len(tweet.entities['user_mentions'])
            number_of_fav_count += tweet.favorite_count
            number_of_retweet_count += tweet.retweet_count

        twitter_ids_userName.append(screen_name)
        twitter_ids_verified.append(verified)
        twitter_ids_default_profile.append(default_profile)
        twitter_ids_default_profile_image.append(default_profile_image)
        twitter_ids_favourites_count.append(favourites_count)

```

```

twitter_ids_followers_count.append(followers_count)
twitter_ids_geo_enabled.append(geo_enabled)
twitter_ids_notifications.append(notifications)
twitter_ids_time_zone.append(time_zone)
twitter_ids_location.append(location)
twitter_ids_listed_count.append(listed_count)
twitter_ids_created_at.append(created_at)
twitter_ids_friends_count.append(friends_count)
twitter_ids_profile_use_background_image.append(use_background_image)
twitter_ids_statuses_count.append(statuses_count)

```

```

twitter_ids_tweets_sample.append(i)
twitter_ids_number_of_hashtags.append(number_of_hashtags)
twitter_ids_number_of_links.append(number_of_links)
twitter_ids_number_of_mentions.append(number_of_mentions)
twitter_ids_number_of_fav_count.append(number_of_fav_count)
twitter_ids_number_of_retweet_count.append(number_of_retweet_count)

```

except tweepy.TweepError as e:

```

    if str(e).find("'code': 88") != -1:
        print("Twitter Limitation. Pausing for 15 minutes....")
        fields = fields[fields.user_id != id]
        time.sleep(60*15) #Sleep for 15 minutes
    else:
        if str(e).find("'code': 63") != -1:
            print(id, "has been suspended. Removing it from the list...")
        elif str(e).find("Not authorized") != -1:
            print(screen_name, "is protected. Removing it from the list...")
        else:
            print(e)
            print(id, "is a bad user ID. Removing it from the list...")
            fields = fields[fields.user_id != id]

```

```

fields.insert(1,'screen_name',twitter_ids_userName)
fields.insert(2,'verified',twitter_ids_verified)
fields.insert(3,'default_profile',twitter_ids_default_profile)
fields.insert(4,'default_profile_image',twitter_ids_default_profile_image)
fields.insert(5,'favourites_count',twitter_ids_favourites_count)
fields.insert(6,'followers_count',twitter_ids_followers_count)
fields.insert(7,'friends_count',twitter_ids_friends_count)
fields.insert(8,'geo_enabled',twitter_ids_geo_enabled)
fields.insert(9,'notifications',twitter_ids_notifications)
fields.insert(10,'time_zone',twitter_ids_time_zone)
fields.insert(11,'location',twitter_ids_location)
fields.insert(12,'listed_count',twitter_ids_listed_count)
fields.insert(13,'created_at',twitter_ids_created_at)
fields.insert(14,'profile_use_background_image',twitter_ids_profile_use_background_image)
fields.insert(15,'statuses_count',twitter_ids_statuses_count)
fields.insert(16,'sample_tweet', twitter_ids_tweets_sample)
fields.insert(17,'total_hashtags', twitter_ids_number_of_hashtags)
fields.insert(18,'total_links', twitter_ids_number_of_links)
fields.insert(19,'total_mentions', twitter_ids_number_of_mentions)
fields.insert(20,'favorite_count', twitter_ids_number_of_fav_count)
fields.insert(21,'retweet_count', twitter_ids_number_of_retweet_count)

```

```

fields.to_csv(fileName, encoding='utf_8', index=False)
print("The result has been saved to: ", fileName)

```

## APPENDIX D Processing

```
#!/usr/local/bin/python

import pandas as pd ##loading tabular data from fexcel

import math ##Module that use math functions

from random import shuffle ## The module is used for random shuffle

from sklearn import preprocessing ## module used classifier processing

from sklearn import svm ## importing Support Vector machine classifier

from sklearn.tree import DecisionTreeClassifier ## importing decision tree classifier

from sklearn.neighbors import * ## import K-NN classifier

from sklearn.metrics import *

from sklearn.naive_bayes import GaussianNB # importing naive bayes classifier

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier ## import random forest classifier

from sklearn.model_selection import cross_val_score

from sklearn.model_selection import cross_val_predict

from sklearn import metrics

from sklearn.feature_selection import RFE, SelectKBest, chi2, GenericUnivariateSelect

from sklearn.preprocessing import MinMaxScaler

#kbest selection doesn't accept negative values so we normalize our features

def feature_selection_with_kbest(features, classes, feature_amount):

    mm = MinMaxScaler() ##Transforms features by scaling each feature to a given range.

    normalized_features = mm.fit_transform(features) ##Fit to data, then transform it.

    sk = SelectKBest(chi2, k=feature_amount) #Select features according to the k highest scores.

    transformed_features = sk.fit_transform(normalized_features, classes) ##Fit to data, then transform

it.

    selected_features_and_scores = sorted(enumerate(sk.scores_), key=lambda x:x[1],

reverse=True)[:feature_amount] ##Identify the features and its score

    return list(zip(*selected_features_and_scores))[0] #zip the selected features and return it

def printResult(labels, result):

    cm = confusion_matrix(labels, result) #generation matrix is created

    tn = cm[1, 1]

    fn = cm[0, 1]

    tp = cm[0, 0]
```

```

fp = cm[1, 0]
p = tp + fn
n = tn + fp
accuracy = float(tp + tn) / (p + n)
precision = float(tp)/(tp+fp)
specificity = float(tn) / (tn + fp)
recall = float(tp)/(tp+fn)
f1 = float(2*tp)/(2*tp+fp+fn)
# print(accuracy, precision, specificity, recall, f1)
print("\tCV Accuracy: {:.2f}%".format(accuracy*100))
print("\tCV Precision: {:.2f}%".format(precision*100))
print("\tCV Specificity: {:.2f}%".format(specificity*100))
print("\tCV f1: {:.2f}%".format(f1*100))
print("\tCV Recall: {:.2f}%\n".format(recall*100))

```

```

file_name_1 = "positive.csv" #Link the real account Excel file
file_name_2 = "negative.csv" #Link the spam account Excel file
positive = pd.read_csv(file_name_1, header=0, delimiter=',') ## read the file
negative = pd.read_csv(file_name_2, header=0, delimiter=',') ## read the file
## adding labels (negative and positive)
positive.insert(23,'label',1)
negative.insert(23,'label',0)
data = pd.concat([positive, negative], ignore_index=True)
labels = data.label
# removing unnecessary features
del data['protected']
del data['verified']
del data['user_id']
del data['screen_name']
del data['label']
data_norm = preprocessing.scale(data)
# 1=SVM, 2=DT, 3=GNB, 4=KNN, 5=RF
clf = svm.SVC(kernel='rbf')
result_1 = cross_val_predict(clf, data_norm, labels, cv=10)

```

```

print('\nSupport Vector Machine')
printResult(labels, result_1)

clf_dt = DecisionTreeClassifier(splitter='best',criterion='gini',random_state=0)
result_2 = cross_val_predict(clf_dt, data_norm, labels, cv=10)
print('Decision Tree')
printResult(labels, result_2)
clf_g = GaussianNB()
result_3 = cross_val_predict(clf_g, data_norm, labels, cv=10)
print('Naive Bayes')
printResult(labels, result_3)
KN = KNeighborsClassifier(n_neighbors=6, weights= 'distance', algorithm= 'ball_tree', p=2)
result_4 = cross_val_predict(KN, data_norm, labels, cv=10)
print('K-NN')
printResult(labels, result_4)
clf_rf = RandomForestClassifier(max_features= 6, n_estimators= 50, criterion= 'gini', max_depth= None)
result_5 = cross_val_predict(clf_rf, data_norm, labels, cv=10)
print('Random Forest')
printResult(labels, result_5)
print('\n\n\t\tResults after performing feature selection\n\n')
k = 5 ## mention the number of features the user want to use to run the performance measurements.
selected_features = feature_selection_with_kbest(data_norm, labels, k)
print('Best { } Features based on KBest'.format(k))
count = 1
for i in selected_features:
    print '\t', count, '-', data.columns.values[i]
    count += 1
selected_features = sorted(selected_features)
data_selected_features = data_norm[:, selected_features]
clf = svm.SVC(kernel= 'rbf')
result_1 = cross_val_predict(clf, data_selected_features, labels, cv=10)
print('\nSupport Vector Machine')
printResult(labels, result_1)

clf_dt = DecisionTreeClassifier(splitter='best',criterion='gini',random_state=0)

```



```
result_2 = cross_val_predict(clf_dt, data_selected_features, labels, cv=10)
print('Decision Tree')
printResult(labels, result_2)
```

```
clf_g = GaussianNB()
result_3 = cross_val_predict(clf_g, data_selected_features, labels, cv=10)
print('Naive Bayes')
printResult(labels, result_3)
```

```
KN = KNeighborsClassifier(n_neighbors=6, weights='distance', algorithm='ball_tree', p=2)
result_4 = cross_val_predict(KN, data_selected_features, labels, cv=10)
print('K-NN')
printResult(labels, result_4)
```

```
clf_rf_2 = RandomForestClassifier(n_estimators= 50, criterion='gini', max_depth= None)
result_5 = cross_val_predict(clf_rf_2, data_selected_features, labels, cv=10)
print('Random Forest')
printResult(labels, result_5)
```

AcademicianHelp