Web Application for Detecting Twitter Bots with Machine Learning

Krzysztof Dworczyk

BEng Honours
Software Engineering
Edinburgh Napier University
2021

Abstract

Twitter is having to fight off wave of automated accounts pretending to be human. This project will investigate different classifiers that can be trained to predict whether a Twitter account is controlled by a human or bot. The classifiers will be trained on user and tweet features then a combination of both to see any increase of accuracy. Six different classifiers are trained to see the general accuracy we get from the datasets. From those results we take two of the most promising ones to further fine tune with different parameters to increase the accuracy. The models will be implemented on a web application which can be then used by users for prediction. The results show that it is possible to train an accurate enough classifier to detect bots. It showed a combination of user and tweet features yields the better results.

I, Krzysztof Dworczyk, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the School's ethical guidelines.

Signed: Krzysztof Dworczyk

Date: 06/04/2020

Matriculation no: 40340711

Table of Contents

Chapter 1

Introduction

1.1 The Context

The project was created in response with the ever growing problem on Twitter with non-human accounts flooding conversations to try and sway opinions. Twitter bots are not a new thing, they have been developed and used on Twitter for years. This was enabled by Twitter providing an API (Application Programming Interface) which helped in developing these bots. Most apps using this API used it for creative or novelty reasons like automated daily tweets with pictures of dog or cats. The API can be used to remotely download tweets with certain hashtags or find people in your geographical location. These tools allowed developers to access huge banks of data, with 6,000 tweets a second and 500 million a day, you can get a completely new never seen before dataset of random conversation by downloading the twitter feed.

Twitter has become so big that nearly every major country has a dedicated account for their presidents/prime minster.Twitter is a lot of peoples go to source for news or any other thing they are interested in. Bot accounts can also create “cascades”, turning an unpopular opinion or tweet into a highly visible one by creating artificial likes and retweet. These cascades even trick human accounts into retweeting and liking it further pushing the tweet into the visible sphere.

1.2 Motivation

As mentioned in [2] that NATO claimed as much as 70 percent of Russian accounts that direct their tweet towards Baltic countries are bots. Moreover, in the same paper it states Russian has also been documented to target countries like US, Germany, UK and Catalonia. This is to cause a disruption in conversation and by disguising themselves as people from those countries they can start to plant small seeds of doubt in people minds about certain topics be it political, military or health related. It all causes destabilisation in people's beliefs in important topics. USA has been surveyed asking its citizens how much they think the US spends on helping foreign countries, majority said around 30%, when it's less than 1%. This is a small fact but enough of these small facts may cause a person to change their minds and opinions.
While the twitter SEC filed a report which says that approximately 8.5% of twitter is made up of bots, the paper [27] states that as much as 9%-15% of accounts on twitter are controlled by non-human actors. As well as they are responsible for generating two thirds of links to popular websites. These are not all malicious bots, but dose show the immense power they have over distributing content in the twitter sphere. The further states that there is a black market for twitter accounts which can be bought and re-purposed to your liking. Bot accounts usually focus on generating a certain type of content with attracts followers and grows their numbers. They might be novelty type accounts that people retweet and like when they find them relatable. This in turn grows the account which eventually may close the account and sell it to a black market where it will be bought by a politics to be used in a campaign. Political bots have been found meddling with elections as far back as 2010 US midterm elections.

There are different types of bots which may include simple spam accounts that spread different topics, sometimes unrelated to any political party or idea. Some bots are paid bots where the person who creates these bots gets money for directing traffic to certain websites. To do this they take some reputable sources like news websites and create micro URLs which count the amount of traffics that used that link. This is not inherently bad; however, some news sites may lean towards a political side which may benefits the political party of more people read from those specific sources. Some more harming bots are influencer bots which exist to shape twitter conversation about certain topics to try garner support. In [5] the paper discusses the negative effects on health knowledge being spread on twitter. Topics such as anti-vaccinations are being spread and talked about which in turn may attack real people to start being sceptical of vaccinations.

The paper [10] has shown examples in which bots have even affected the stock market. The most effective way bots sway the public opinions on topics is by a network of bots called “botnets” which are controlled by a single bot master. Using this technique, they can launch coordinate disinformation campaigns on certain things like the stock markets. If a network of bots starts to tweet positive things about a new up-coming business, then in turn the bots who analyse twitter for market predictions may flag up that a company is going to be a valuable asset in the future causing it’s shares to increase.

1.3 Aims and Objectives

The main aim of this dissertation will be to develop and test the best machine learning algorithm in detecting bot accounts on twitter. It will predominately focus on two main classifiers ones being Random Forest and Super Vector Machines(SVM). These have shown the best results in general bot detection and with he right features choice could be implemented. I will discuss both classifiers strengths and challenges as well as what features it’s best used on.

A good dataset will need to be collected and for this I will be using a combination of different labeled ones to try and get a good mix of different accounts. These dataset will be broken down in later chapters with the number of accounts in each.

Once we have the best classifiers found and trained we can implement them on a web application. This will be the front end and will allows users to enter username for prediction.

- Research into the most current classification models and asses each one for its strengths and weaknesses
- Identify the most suitable ones
- Train the best classifiers for bot detection.
- Evaluate and compare the models

- Implement the models on a web application

Chapter 2

Background Research

2.1 Pre-Processing

“Data pre-processing aims to reduce the data size, find the relations between data, normalize data, remove any outliers and extract features from data” [1]

Before a body of text can be worked on, we first must sanitise the text by removing any duplicate words and turning each work into a token which can be fed into a machine learning algorithm. This is known as data pre-processing and is usually regarded as one of the most important first steps in analysing data. Below I will be describing some of the most commonly used techniques in data preparations.

Firstly, we need to separate all the individual words into tokens, also called tokenisation. It separates sentences into individual words [23]. This will allow us to create a list of words which then can be further reduced and cleaned up using more techniques. This however crates the base dataset which we can begin working with.

Our database may have many variations of the same types of words like numeric values for numbers, numeric representation of dates or acronyms and abbreviations. To group all these together we would normalise them into uniform style where all number are converted into alphabetic forms along with acronyms and abbreviations fully expanded.

Lemmatization is ”the task of determining that two words have the same root, despite their surface differences” [20]. It is changing words like jumping, jumped, jumper and jump so that it's all represented by one, "jump". It's reducing words to their simplest form or "lemma" which in this case is the word jump.

Stemming is a form lemmatisation but it's simpler in that we usually just remove the suffixes of a word [7]. This may cause some problems as removing suffixes like "ing", "es" or "is" from some words may result in a different word. Axes is plural of axe and axis, if its stem is removed "s", then it might have been wrongly translated into the wood chopping axe whereas we intended it to be a word for describing x-y planes.

To further reduce the vocabulary and removed redundant words that would just add unnecessary noise into out models we can remove common words like "the", "a" or "and" [20]. This is only done when you know that these words add no contextual value. Another common approach may be selecting the top 10 frequent words in your corpus and removing them as they usually don't add value in most cases.

The NGram is a continuous sequence of N number of items from the begging. We can have unigrams, which will have one item in each, bigrams will have two items and trigrams will include 3 items. This can go on "four-grams, "five-grams" [7]. This can be used in probabilities to see how often a set of N words occurs in a corpus. For example, the word "wine" is a single word however, the colour of wine "red" or "white" might precede it.

A bag of words (BoW) technique can be described as "an unordered set of words with their position ignored, keeping only their frequency in the document". This usually used to perform features generations. From the BoW we can deduct how frequently a word is used throughout a corpus. This is useful as some classifiers only allow numeric values [20]

2.2 Machine Learning Models

Machine learning has been used in text classification with successful results in many areas. Some examples of the most common ones are Naïve Bayes, Logistics regression and neural networks. In this paper I will go be discussing and using two other ones namely Super Vector Machine (SVM) and a decision tree called Random Forests. Both have been used extensively with positive results in multiple literature's.

(SVM) is a statistical probability classification model which will model depending on the classes it's given. Since the classifier need numerical data to properly work, we first need to reduce our data into numbers. To do this with words and sentences we can use the Bag of Words approach which will turn them into the frequencies of each word. The classifier will attempt to create a hyperplane which best separates the data points by using a hyperplane. In a one- or two-dimensional data this will create a simple line splitting the data apart. Anything on either side of the line will be assigned their respective class. [21].

To maximise the best results, we use a hyperplane to separate the data, we then try an maximise the margin between the hyperplane and the first data-points. This would allow for the hardest cases to be classified and therefore anything further away from it will be much easier to classifier. [3]

We may not always have linear data to work with and therefore we may need to apply a kernel function to each data-point in order to transform them into a higher dimension. For example, a point on a one-dimensional line may be squared to turn it into a two-dimensional point. Once all the inputs have been transformed, we may be able to draw a hyperplane which allows for a better separation of the data points. This also applies for 2-dimensional data which can be transformed into a 3-dimensional point.

Kernels are a key feature of SVM and can be used to save a lot of computational time. To name a few Polynomial kernel, Gaussian kernel, Hyperbolic tangent kernel and string kernel. The trick in using them is to know your research domain and data very well as each will transform the data into different forms some being more useful than others while some may be detrimental to the results.[15]

Random trees are an ensemble classifier which is created from of a classifier called decision tree and works best as a supervised machine learning model. The main components of a decision tree are the root node which is the starting node, internal nodes and leaf nodes. The internal nodes are where the decision making happens and will connect to nodes further down. Leave nodes are the final classification nodes which do not split further down as their impurity is as low as it can be. [19]

We decide if we want to split a node into further decision based on the impurity score ranging 0-1. This is calculated using the Gini Index or other forms like Shannon's entropy. These calculation lets us know which split will gain us the most amount of information on a certain class. The closer the score is to zero the better the information split will be. Once a node reaches its maximum split which is to say a further split would not yield a lower score, we can stop and make it a leaf node that will be used for classification.

In the case of Random Forest, we generate multiple of these decision trees with a slight variation by changing some parameters or the seed to create them. Once a single decision tree becomes deep, we may begin to experience some bias and over-fitting problems due to each node trying to classify a an increasingly smaller set of data. The main advantage of using multiple decision trees is that we can avoid these biases as instead of using all 20 features, we can split each decision tree to only focus on 5 at a time. With enough decision trees created we will eventually cover all the features. [18]

Finally, once all the trees have been created, we end up with an ensemble classifier. This is a much more powerful classifier as it creates a voting system, once the classification has been made the average is taken and the class with most votes is chosen. This allows for a more robust and less bias prone classification. [4]

2.3 Text Classification Models

There have been many attempts at developing the best machine learning model in classifying bot behaviour. Some simple one have had surprisingly good accuracy like in [13] where there were not in depth analyses of each features but rather they compared the features of a selected profile to its N number of neighbours. No classifier was used either just each accounts feature was looked at and scored with some features having more weight, like if similarity between two tweets is high then its more likely the account is automated. They found just under 11,000 unique bot accounts of which 2,586 have been suspended since. This showed that even the simplest approach can yield good results.

Another new approach is seen in [12] where a large 500,000 cluster of accounts was found by analysing their twitter IDs as they were created in “bursts”. When a twitter account is created it is given a unique identifier in chronological orders which gives a way to see when accounts were created. In the instance of the paper they have first identified a “starwars botnet” network using this technique where the account only tweeted Star Wars related quotes. Upon analysing the network, they spotted another similar acting network of bots called the “bursty botnet”. The accounts in question once created tweeted 3 times within the first hour then stopped. The tweets were filled with spam URLs and hashtags which were more than likely trying to spread them inside the Twitter-sphere. Clustering methods have also achieved good results in cases where supervised learning may fall short and allows to see suspicious groups of accounts that act in synchrony [25].

A fairly new idea has been implemented in [9]. Instead of using a reactive approach to bot detection, which means the detection techniques are improved only after a certain number of bots have been spotted. It uses a proactive in the sense that it improves at the same time as the bots. This is achieved by feeding a DNA sequence made up of a user activity online as a string to a custom designed genetic algorithm. It then mutates the profiles multiple times in hopes that it will create a superior bot. This then revealed some interesting features which can be applied in further detection like the entropy within the DNA sequence which allows for a better way of distinguishing bots from legitimate humans.

The most overall successful and adaptable machine learning models have been Random Forest and Support Vector Machine. These have both proven to be widely used in publicly available application BotOrNot [10]. It uses a supervised ensemble of classifier Random Forest and generates over 1000 features to analyse user profiles to determine their authenticity. Many papers have used the system as an additional feature score of a profiles “bot-ness”[18]. Additionally more papers [16] have used random forest and achieved satisfactory resulted of ACU exceeding 98%.

Another new approach suggested by [16] is to use neural networks. This approach has presented a nearly perfect score (ACU >99 percent) by combining tweet and profile level data then use it as auxiliary inputs for the LTSM deep nets. They then go and by only using a single tweet per account can achieve a high accuracy of 96 percent ACU

2.4 Ensemble Classifiers

Ensemble methods are methods that use multiple classifiers to achieve a higher accuracy rate by introducing a voting system between the classifiers. The data is inputted into each classifier and the outputs are read; whichever class is the output for more classifiers, wins. In this way we can mitigate certain lower accuracy scores since we have multiple models deducting if a certain body of text belongs to a certain class. In the example of [11] the researchers have build a custom classifier called "Senti-Bot" which is a collection of six classifiers: Naïve bayes, SVM, AdaBoost, gradient boosting, random trees and extremely randomized trees. With these they were able to achieve very satisfactory results.

Random Forest is itself an ensemble of classifiers as it is made up individual decisions trees. In this problem domain it has shown the best results. When trained on single datasets it has shown near perfect ACU scores. [26]

2.5 Challenges of machine Learning Models

The main challenge faced in the domain of creating classifiers to detect bot activity on twitter, is that it's a reactive process [9]. Meaning that in most cases the progress is only made once a various number of bots are spotted and collected to be able to train the next generation of detection classifiers. Moreover most of the time only relying on machine learning proves to be insufficient [14], the best results can be seen from a mix of human input to spot the first few and collecting their features to then use them in a classifier to find more.

It is also hard to get annotated data to train algorithms, this will be discussed in detailed in further sections. Furthermore bots nowadays display behaviours more similar to humans which makes it difficult to detect them. A new generation of sophisticated bots may requires different strategies to detect them.

However, it is stated in [27] that supervised machine learning methods are good in most cases but they fall short in detecting coordinated bots. These shortfalls can be improved on by implanting clustering algorithms [25] which spot behaviours in groups of users. It is also important to take into account as the detection system increase in complexity they may become more vulnerable to biases. [27]

2.6 Type of Bots

In general, we can split the types of bots into 2 broad categories, normal and malicious. These categories can be broadened and expanded, and it can be argued if certain bot fit in the category. Like marketing bots, these automated accounts may post tweets advertising certain products and link to their respective websites. These may be legitimate advertisements where it's clearly obvious that the account is a bot. The simple act of posting links to their website may be to generate some interest and get people buying. We however encounter a problem once the bots try and behave in human like ways to try and pass as a real human. A lot of bots on twitter that can be spotted by the naked eye are just novelty accounts designed to play about with the API and entertain people. Like the big ben account [27] which every hour posts a tweet simulating the "bong" of the real one. Useful bots do however exist, they can post news articles or public service announcement. [11].

2.6.1 Malicious Bots

The focus of this paper will be to use machine learning to find the best classifier and features in detecting political propaganda bots. However, there are many more bots whose goals may be different to the ones spreading political ideas. The recently more apparent one can be seen with the sudden merge of anti-vaccination movements. These accounts have been reported in [5] and have been mostly linked to a Russian origin. These accounts promote and encourage discussion about anti-vaccination which can lead to serious health problems in the public. Gullible people who may not know better may view this new "upsurge" of anti-vaccine talks as a reason to be sceptical of it. The simple act of taking about anti-vaccination may have a negative effect. In some cases, there have been monetary incentives in these bots spreading spam links and hashtags to generate views and clicks for certain websites. In [14] it's stated that bots exists which are programmed to spread URLs disguised behind micro-URLs to link to certain political leaning websites. These micro-URLs are assigned to each bot network which is controlled by a bot-master. Then for every click they achieve and redirect traffic to the website, they will receive a small sum of money. The amount will be very small but if the bots are deployed correctly and attract a lot of people, they may end up received a good amount for it. It is very unlikely that bots operate on their own. The best and greatest effects will be achieved using a network of these bots all controlled by a single human or software called the bot-master. [25]. These types of networks can produce huge influence amongst the followers if they can burrow themselves correctly in the right communities.

2.7 Features of Bot Accounts

Many papers use different techniques and features to detect bots. Below I will discuss some of them and which are more suitable in detection of bots on twitter. I have split them into two main groups, firstly the individual tweet features that the user posts which include retweet/favourite counts, number of hashtags/URLs/mentions inside the tweet. I will then further discuss the account level features like the number of tweets a user has, followers and friends count as well as profile features like dose it has its location displayed or dose it use a default profile picture. For good results in classifying bots it's best considered to use a mix of both the tweets and some metadata around the account posting those tweets [26]. Some of the papers as well as having their own features also include a "bot score", this is the score given by the popular application BotOMeter and proves to be a reliable tell if an account is a bot. [2]

2.7.1 Tweet level

Using tweet sentiment to detect bots has be successfully applied in [11] where the research create a system called "SentiBot" which uses mainly the sentiment of tweets to deduct if their humans, bots or cyborgs. The results showed that bots rarely changed their sentiment on topics while humans were more likely to. This is because the bots have been programmed and given parameters to tweet in a certain way so there's no much room for them to flip their opinions unless it is a more sophisticated bot. Furthermore, it showed that bots when expressing negative or positive sentiment, do it less strongly, possibly to not stand out as much. Lastly it showed that bots tend to disagree less with the twitter population, again it's possibly to maintain a good follower relationship. Drastic changes in opinions and sentiment may drive their followers away.

In terms of political accounts, the top features are by far the political ideology. To create this feature the papers [2][18] firstly collect a large collection of users. They then separate the users into their political camps like democrats or republicans. To do so they check each account and the website links it's posting, if the majority of links is to a democratic leaning news source or website then it it's labelled that. Same is applied for the opposing party. In the case of [2] it found that conservative bots' tweet 7.4 times per accounts which was nearly 3 times as much as liberal human account (2.5) and nearly twice as much for human's conservative (3.9).

2.7.2 Account Level

Looking at the accounts we can begin to see that certain features stand out more than others in detecting political bots or automation in general. The well-known application BotOMeter has over 1000 features which it analyses for suspicious activity. There can be a discrepancy between the time zones of users, if for example an account has a USA time zone but most of his followers are in others like Moscow. Device metadata is also important, as in [8] we are told that 42.39 of bots detected used some kind of unregisters 3rd party APIs while half of the humans (50.53) used the actual twitter website for posting. The deletion patterns of an account can also be a red flag. Some clever bot accounts may try and obscure their intention by periodically deleting their old tweets as to maintain a good following and not be spotted by any detection systems. Such things are hard to monitor as it requires for the account history to be recorded over time to see if anything has been deleted. [27]

2.8 Dataset collections

Supervised Machine Learning models are trained and tested on the data they will classify. This means that for a precise and robust classifier to exists we must first make sure the data it's being trained on is the best it can be. As a paper quotes "A supervised machine learning tool is only as good as the data used for its training." [27]. Further in the same paper it is said that the current issue with training models is the lack of ground truth. There doesn't exist a dataset that encompasses all of types of bots and features to detect them. There also isn't a consensus on what the exact features of a bot are.

Albeit the difficulties of making dataset there have still been good ones created from different techniques. Arguably the hardest part of creating a dataset is the annotation of individuals users which can be a costly endeavour as it takes a long time for a human to asses each one. In [11] the annotators were given an incentive to correctly identify bots, for every correct one they would receive 1 dollar, for every wrongly assigned one they would have 1 dollar subtracted from their pot. Having a monetary incentive would arguably improve the accuracy of detection which in turn would result in a better annotated dataset for training.

The most popular and accessible way to collect large quantities of data is through the provided Twitter API. In the paper [18] the researchers collect nearly 1 million users with a collective tweet count of 2.6 million within just 42 days. Now these aren't just any random user but rather they were able to create a query search for user with specific political hashtags in their tweet. This allows for datasets to be created around topics of interests (TOI) [11] and narrow down the search. Another paper, to get a set of legitimate users have went and mentioned users in tweets and once contact is made, they ask a question in their natural language. If answered correctly then they could be sure that the user is legitimate. [9]

In the early stages of collecting dataset for this specific topic of detecting twitter bots started as far back as 2011. One of the first major datasets collected was using a "honeypot" method [17]. They created a bunch of agents whose sole purpose is to attract suspicious accounts. This is achieved with moderate success as the content on those accounts is irregular and doesn't' resemble human activity. This means that any attempt to engage with these profiles be it by mention or follow is likely to be an automated account trying to enlarge its own following.

Twitter is very aware of its platform being overrun by malicious bots trying to spread political, health, marketing or ideological ideas. They are actively fighting against it but it's an enormous task to tackle as being too strict can lead to real accounts being suspended or the opposite, where some real bots are caught but the most sophisticated ones get past the firewall. In 2016 as part of their transparency report they have released an additional resource which includes all the suspended accounts it has caught. It separates them by the year and month as well as the language of origin. It further separates each dataset into 3 components: account information, tweet information and the media. The media is sometimes into hundreds of gigabytes making it very useful to separate the components. [24]

Moreover, it can be difficult to collect data about individual users from older dataset because twitter might have taken action against those accounts and suspended them off their platform which deletes all of the user's tweet history and any account information. In the paper [18] once the data was collected over the 2016 election period the accounts were fed through the BotOrNot API to return a bot score, however 3.5 percent of the accounts didn't return anything. By further inspecting the accounts it was clear that 99.4 percent of the account have been suspended by twitter with the remaining turning their profile to private.

2.9 Bot Detection Systems

Currently a popular solution out there for detecting automated accounts on twitter is a web application called "BotOrNot" [10]. With he system you are able to enter a username for it to use the twitter API to retrieve the profiles data along with some of their tweets. This is then used as inputs to the classifier. The web app collects over 1000 features from the account which are split into 6 broad categories of Network features, User features, Friends features, Temporal features, Content features and Sentiment features. All of them combined result in a fairly good prediction. The accounts "botness likelihood" is then displayed for the user on screen. The system allows for additional features like scanning the users friends or followers to see if they have large groups of accounts which are suspicious.

There exists another service called DeBot [6]. This system is always on and detects bots on a daily basis. It tracks accounts and detects if there are too many synchronous events occurring. The probability of 2 accounts not being bots when 40 or more activities occur within an hour is close to zero. With this the system dose not have to take any metadata or create features it has created it's own simple yet very effective detection strategy. It has so far from 2015 to 2017 detected over 700K bot accounts.

2.10 Ethical Implications

The first ethical aspect worth discussing is the documentation of all the systems used to detect these bots. Many researchers believe that the best way to combat these bots may be in opening more to the public by helping people better interpret the results of some the already existing applications [27]. This however may be counter-intuitive as it also allows for the creators of these bots to find out what flaws may flag up their systems as being a bot [9]. The current system is a kind of cat-and-mouse-game where researchers are in constant pursuit of creating better detection methods and the bot creators evolve their bots considering new developed techniques to catch their last generation bots.

Developing systems which can wrongly accuse people of being bots, leads to potential account suspensions of legitimate users. Currently the biggest downside would be an increased stress on verification times for each humans or potential dissatisfaction of wrongly suspended accounts [22]. It's important to not only improve the models so the detection accuracy is greater but to discuss the overall ethical aspect of the field. We have seen states like California introduce laws which would require bots to identify themselves when communication or interacting with humans online [27]. This may be a good step in the right direction as so far there is nothing illegal with creating bots on twitter and passing them as human beings, there needs to be a path for legal investigation of such behaviours as to deter future attempts.

This raises an interesting question where discussion have been formed around the rights for these automated systems [9]. Since some of these bots help spread news articles they could be classed as reporters and censoring them could be viewed as censorship. This is a much deeper topic of should robots and AI be even considered as human and if they must obey by our laws should they also receive the same human protections as we do.

We should continue discussing these topics as often as possible because there is not a clear right and wrong answer here and the only way we can know what to do is by communication and setting examples.

2.11 Literature Review Conclusion

Chapter 3 has explained some background knowledge to preparing data for classifier training and some common classifications models like Random Forest and Support Vector Machine and how they work. Chapter 2.3 went into details about the common classifiers used in detecting bots on Twitter with promising results. Chapter 2.4 mentioned the success in combining multiple classifier together to increase the accuracy's.

Chapter 2.5 discussed challenges like insufficient amount of training data. Chapters 2.6 and 2.7 detailed all the different feature levels you can gather from twitter accounts and what the best ones were. A solid mixture of different ones seemed to do the best. The last chapter touched on the ethical aspect of the project as we are dealing with human accounts that can have an affect on people. False positives are especially bad as it causes lots of inconveniencing for users.

Chapter 3

Technology

3.1 Introduction

In the following sections I will be going over the different technologies and libraries I have chosen to deliver the final web application. It will mention Tweepy and how I will be using it access the twitter API. The libraries picked for text cleaning and analysis will be discussed and the main two technologies which the entire application will be developed in will be mentioned and why they were chosen.

3.2 Tweepy

The main collection method of user data has been a library in python called Tweepy. This library allows calls to the Twitter API which can return any number of data on user profiles and timeline. Below is a table with some methods that come with Tweepy and allow me to gather data from users profiles

In this project I will be using Tweepy as my main access point to any type of action related with Twitter. This will include things like using the get_user() to collect my datasets. This method has many different arguments that can be passed like using twitter username or id to search for specific accounts. For the text analysis and tweet dataset collection I will make use of the *Cursor* combined with get_user_timeline() method. The search can be furthered refined with queries like collecting tweets which only contain a certain word or ones which don't.

This will also be used for the user access point for predicting as twitter requires

Table 3.1: Tweepy Methods

Method	Description
get_user()	Allows to call the API and find a user by the name or ID provided
get_user_timeline()	Allows to call and find a user. Then take X number of tweets fro their timeline
Cursor	The cursor allows for an easier way for accessing the tweets on users timeline.

you to have a secure access point which can only be obtained by logging in yourself or by creating an app on an account and providing access to the account using secret tokens and keys. This app has already been created and will be used throughout the project to provide this access.

3.3 scikit-learn

To train and test the classifiers I went with the extremely well documented and up to date library scikit-learn. It is mostly used with python which makes it further beneficial for the project as its the language of choice. The library allows really quick and easy out of the box training and testing of classifiers. It also provides a plethora of parameters to tweak which can fine tune the models yielding the best accuracy results. We will be able to quickly test some classifiers to see a base accuracy with the different precision's and recall scorers the library provides.

3.4 NLTK

Before model training we have to clean the data, for this I have taken the Natural Language Toolkit library to use its methods and classes to clean the data. Data will be normalized to see if performance can be increased. It provides us with a vocabulary of stop words and punctuation's to remove from out tweet texts. It will allow for easy text to be tokenazied followed by lemmetzaions and stemming. This will leave the text clean and ready for our processing and analysis by our models.

3.5 Flask

Flask is a relatively new web framework coming into existence in 2010. It is a micro and lightweight web framework which allows for rapid development and prototyping of applications. It is written in python and therefore can only be used with it. It is a well documented framework and being a lightweight one dose not take away from the complexity of application that can be developed on it. It provides all the security and flexibility features to deploy any number of web applications. The choice of this framework was due to the simplicity and how quickly a web page can be set up to start implementing the classifiers. It works on a ”route” based systems where each page is called upon when required and sent to the user for displaying.

3.6 Python

The programming language of choice for this project had to be python. It is an extremely popular general purpose language that also allows for great data analysis and manipulation. The language is light-weight providing only the most necessary functions but excels in the amount of different libraries provided for it.

Apart from Tweepy and scikit-learn, there have been other very useful libraries that have helped in developing the best classifier. These will be listed below.

- CSV
  - This simple library was used to read in the CSV (Comma-separated values) files of Twitter user IDs to extract data from. The user data was written back into a new CSV file. This is the most popular file type when dealing with datasets.
- Pandas
  - This library was used to allow for a better data manipulation and analysis. It comes with it’s own data frame structure which makes data manipulation much easier. It allowed for an overview of the data using commands like DATAFRAME.describe() which displayed a tabled with values like Count, Mean, Std as well as 25% to 100% values in a specific field. This gave an early insight into the data in case there were any problems.

- Numpy
  - Is a requirements of Matplotlib. It allows for a number of high-level mathematical functions. It also allows for a better implementation of graphs and plots using matplotlib
- Matplotlib
  - The library allows for implementation of mathematical plots using numerous different ways. It helped in visualising out datasets and any interesting features found in it.

Chapter 4

Analysis and Design

4.1 Introduction

In the following section I will conduct analysis and design discussions of the software requirements specifications. It will go into detail about the more general purpose and scope of the project as well as the type of users the application will be aimed at. The functional and non-functional requirements will be presented which will go over essential features which must be implemented and the once which won't. In the design section I will discuss the methodology I will be using to develop the software as well as show different diagrams that will be the first step in designing the application.

4.1.1 System Requirements Specification

4.1.1.1 Purpose

The web application will be made available in hopes to combat the ever increasing bot swarms on twitter trying to sway public opinions. It will give users a way to see if an account they are suspicious about may be controlled by AI. This will allow for a more streamlined way of detecting bots on twitter for the public to use.

4.1.1.2 Project Scope

The the project will aim to provide a cohesive solution to finding out if an account is controlled by an AI. The web application will allow for one username to be entered at a time which will be used to extract account metadata and tweet features for analysis. It will collect the user entered number of tweets to do text analysis on, this will further help in the strength of the classification. The application will allow the users to switch between classifiers

4.1.1.3 User Characteristics

The user will not have to have a technical background as they will only have to enter a twitter username. The application will display the prediction of bot or human very clearly which will make it very easy to understand. As more technical users may be interested in the application there will be more detailed numbers provided which were used in the prediction.

4.1.2 Functional Requirements

4.1.2.1 User Interface

Inputs

- Text box for inputting the suspicious account username
- A radio button selection to choose what classifier you wish to use
- Number field to enter amount of tweets you want to extract from the user for analysis

Output

- A multiple line breakdown of the predictions and the percentages of bot or human likelihood
- A list with all the tweets used for analysis and their individual predictions

4.1.2.2 Specification - MoSCoW Table

Below is as table with features accompanied with their IDs. All features get a priority check mark depending on if the feature is to be implemented or not. The **must** will need to be delivered or else the web application will no be complete. The **should** will be additional features that will be implemented if no major problems arise during development. The **could** are features which could potentially be implemented only if there is enough time and the rest of the features are all fully functional. The **wont** is self-explanatory, features which are acknowledged however won't be implemented in the final release.

Table 4.1: Application features break down

Req. ID	Description	MUST	SHOULD	COULD	WON'T
1	input a username of profile to analyse	✓			
2	select a classifier to use for classification	✓			
3	select number of tweets to analyse on	✓			
4	allow the user to log into their twitter account to begin classifications	✓			
5.1	display the results to user	✓			
5.2	displays the probabilities of predictions	✓			
6.1	display the analysed tweets		✓		
6.2	display probability of each tweet being bot/human		✓		
7	display all the user metadata used for classification	✓			
8.1	analyse suspects followers and friends			✓	
8.2	display each accounts probability of bot likelihood			✓	
9	display a visual representation of how the prediction was made				✓

There will be only one features which is planned to not be implemented. That is to visually display the prediction once done. this can be done however with the time constrains of the projects I will only be focusing on displaying the percentages with the user tweets and metadata.

4.1.3 Non-Functional Requirements

4.1.3.1 Software

The page will be written in HTML and be made up of basic input elements like text boxes, radio buttons and number fields. It will be implemented on the Flask framework combined with Python as the back-end for the logic.

The program colours will be followed:

- Background: Gradient colour(Purple/Red)
- Font: sans-serif
- Font colour: White
- Font size: Varies (12px-20px)

The minimum operational Windows 7 hardware specification is:

- OS: Any
- Memory: 2GB+
- Graphics: Any
- Network: Broadband or wireless connection required

4.1.3.2 Security Requirements

The application will not be storing any personal information and will be using the secure twitter API to make any calls to retrieve information. Any data retrieved will only be used to make a prediction and will not be stored anywhere after. This means the application is not vulnerable to any outside attacker trying to gain access to the database.

The application will be written with security in mind as to allow for the implementation of a multi layered security system that prevents XSS, CSRF and SQL-injection attacks. This will ensure that anyone using the website and logging into their Twitter account can be further sure that the website takes care of security. This will be on-top of the already secured Tweepy connection.

4.2 Design

4.2.1 Introduction

In the following sections I will be going over the design of the application which will be made presented using different variations of diagrams like USE CASE for user perspective and UML class from the software perspective. I will breakdown the user profile metadata features used for analysis and how I arrived at some of the derived ones. I will also mention the software development methodology being applied throughout development and why. The chapter will finish up with a final wire-frame mock-up of the web app and a description of the various elements and fields on the page.

4.2.2 SD Methodology

For the duration of the project I will be following an agile project management method called eXtreme Programming (XP) which will have a sprint duration of 1 week as it suits the weekly meetings I will have with my supervisor.

This will allow me to make the necessary changes suggested by my supervisory over the week and show the changes in the next meeting. This approach will also allow me to stay the most agile in development and decrease any overhead risks and problems that may appear in development. I will be able to make big changes if absolutely necessary and still manage to to deliver the program as this approach is not document driven but rather development drive. XP advantages are its ability to adapt in real-time and allows for change late in development. The test cases will be written along with the code whihc means we are guaranteed a functional app if correctly done.

4.2.3 USE CASE Diagram

This diagram is meant to give an end-users point of view. You will be able to see all the interaction a user can do with the application and what it will lead onto next. This should roughy encompass all the MUST and SHOULD features we have seen in the analyse stage.

As seen in the diagram the user will have 3 main choices at the begging, the first big one being the options. In the options drop down menu the user will be able to

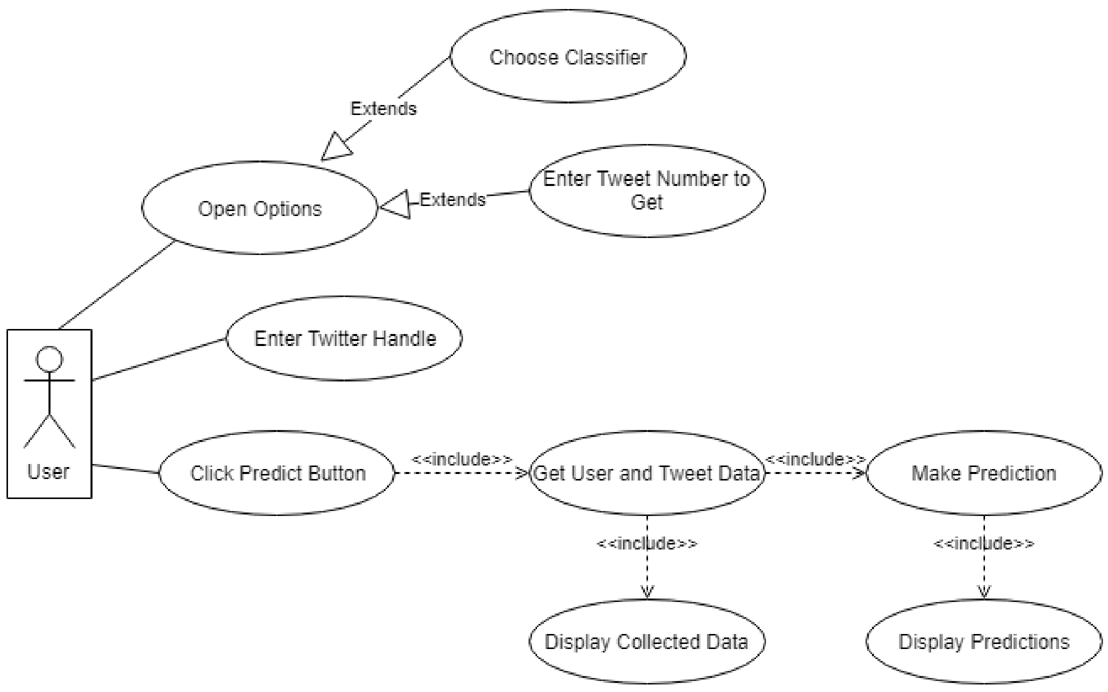


Figure 4.1: USE case diagram showing the interactions the user can make with the system

chose the different classifiers and number of tweets they want to predict on. They will be then able to press the predict button to begin the classification.

Once the prediction is made the results will be displayed on-screen. They will provide a breakdown of likelihood the user is a boot as well as each tweet used for the analysis. The user will be then able to enter another username or play around with the options with he current user.

4.2.4 UML Class Diagram

The UML diagram will give a break down of the classes which will make up the final application. The main class will be the ”App” and it will be the one which uses all the other ones to arrive at a prediction. It loads all the classifiers using *joblib* library at launch and waits for user interactions.

twitter_credentials.py

In this file the API token and key secrets are saved as to allow for Tweepy to successfully authenticate and allow for us to query and stream tweets. It will only contain a class which will return its private classes containing the strings for authentication.

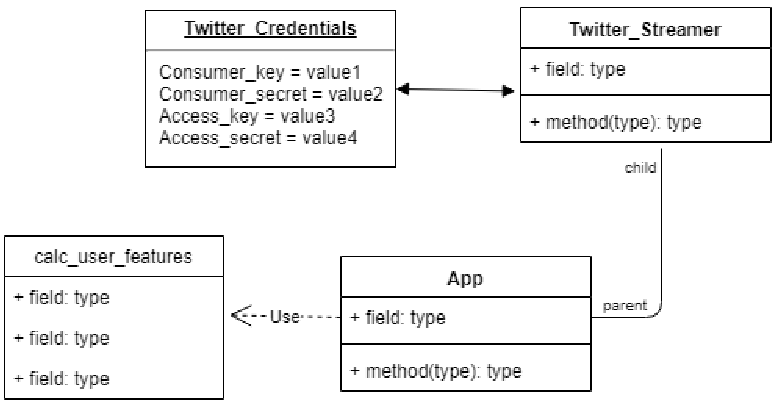


Figure 4.2: A picture of the same gull looking the other way!

twitter_streamer.py

This class will be responsible for all communication with the twitter API. It will create a Tweepy API object which we can use to perform all of our user data and tweet collection. Using the get_user method we can pull any profile from twitter based on the ID the user has entered. The object we get in return for calling the API is in the JSON format, when pulling the profile I perform a small change from the True and False values given to 0 and 1 to prepare it for the model training. This could be done in the text cleaning section however its a small step that saves time so we do it here. The class will also take the average amount of hashtags, mentions and links a user has in the number of tweets we have pulled from the account.

calc_user_features.py

This is the main processing class which we feed in the metadata from user profile like statuses count, followers, following, account creation date and we use those features to make some derived ones.

First step is to get the account age. This can be done by taking the current date and comparing it to the date we get from the *created_at* attribute in the JSON object. This will allows us to get some average data from other features.

Using the account age we just created we can further derive some features like tweet_freq by dividing the status_count by the account age. This gives us the monthly average of tweets the account makes. Similar rules apply for followers growth, we take the followers and divide by account age to get the monthly followers the account gets.

App

The app file is responsible for managing all the other classes together into a working software so that we can enter twitter usernames and make predictions on them. The class will load classifiers at launch to be ready for use. Once user presses predict button the class will check which model the user has selected and perform prediction using that model.

4.2.5 Data Structure

In this section I will go over the user features which are used to make the predictions. They are broken down in 3 main categories: metadata, user derived and tweet derived. Their descriptions and datatype will be described below.

Column Name	Description	Datatype
statuses_count	number of tweets a user has made.	Integer
followers_count	number of followers on account.	Integer
friends_count	number of accounts followed by user	Integer
favourites_count	number of favourites a user has. (Liking someones tweet)	Integer
listed_count	number of times the account has been placed in lists	Integer
default_profile	is the account newly created and has default features	Boolean
profile_use_background_image	dose the account make use of personal background images	Boolean
verified	has the account been verified by twitter	Boolean

Table 4.2: User profile metadata features

The above table is made up of user profile features which have been directly collected from the profile metadata like statues counts, followers and friends. They do not require any further processing and were saved as gathered from profile. These are mostly the surface area details about the account but nonetheless can be used in combination with the rest to produce interesting data points in our datasets.

Column Name	Description	Datatype
tweet_freq	average number of tweets in a month	Float
followers_growth_rate	average followers growth in a month	Float
friends_growth_rate	average friends growth in a month	Float
favourites_growth_rate	average favourites growth in a month	Float
listed_growth_rate	average listed growth in a month	Float
followers_friends_ratio	the accounts followers:friends ratio	Float
screen_name_length	length of screen name	Integer
num_digits_in_screen_name	number of digits in screen name	Integer
name_length	length of the username	Integer
num_digits_in_name	number of digits in username	Integer
description_length	length of description	Integer

Table 4.3: User profile derived features

The above table is made up of derived user profile features. These are made from various calculations. All the ones ending with **growth_rate** are calculated from averaging the activity in a month by month basis. The rest should be self explanatory like lengths and ratios. As mentioned in [26] it is best to leave out the account age as a features as its detrimental to the results.

The above table shows the final 3 features. These ones are derived from the tweet

Column Name	Description	Datatype
avg_num_hashtags	average number of hashtags in X amount of tweets	Float
avg_num_user_mentions	average number of mentions in X amount of tweets	Float
avg_num_links	average number of links in X amount of tweets	Float

Table 4.4: User tweet derived features

but not from the actual text. These are considered to be part of the profile and and be thought about as a profile metric of how frequently they include hashtags, mentions or link sin the tweets they post. These should be interesting features as bot and human accounts are likely to tweet in different ways.

The X is a placeholder number as when predicting a single user these averages are taken based on the amount of tweets the user has entered for analysis.

4.2.6 UI

In this section I will try and present mock-up of the final web application. The final product may vary with where the inputs are placed but the actual inputs themselves will all be the same.

The UI mock-up should serve as a good depiction of what the final web application will look like. The left side will be used for a drop-down menu box for options. The middle of screen will present the user with an input field for the username. Once it is entered and the button is pressed the data will be collected and predicted on. This will all be displayed under the predict button.

The collected profile metadata will be re-displayed to the user so they cans see the inputted data. once the prediction is complete and tweets have been analysed, they will be displayed at the bottom of the page with all the necessary probabilities.

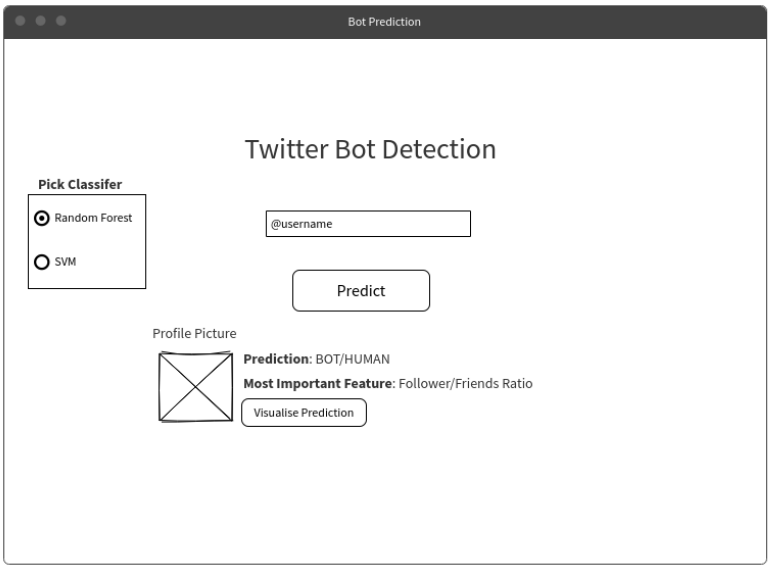


Figure 4.3: Mock-up of the final web application

Chapter 5

Implementation

5.1 Introduction

In the following sections I will be discussing the implementation approach taken to deliver the project. It will break down the datasets used for the training and testing of classifiers. It will describe the classifiers and settings used in the training as well as testing six of them to find the best two we can use for further parameter tuning. The final web application implementation will be presented with descriptions of the elements on screen. Lastly the section will discuss the testing that will be done to on the app to ensure its robustness and functionality.

5.2 The Datasets

Below is the table break down of all the data sets which will be using in training the model. Each dataset will be described and the number of bots and humans in each dataset. this number will vary from the actual ID list provided by the file as some account were suspended or deleted.

Dataset	Description	Account Types	# of Accounts
botometer-feedback-2019	Botometer feedback accounts manually labeled by K.C. Yang.	bot/human	54/243
botwiki-2019	Self-identified bots from https://botwiki.org.	bot only	157
cresci-rtbust-2019	Manually annotated bot and human accounts.	bot/human	139/205
political-bots-2019	Automated political accounts run by (AT)rzazula (now suspended), shared by (AT)josh_emerson on Twitter	bot only	13
verified-2019	Verified human accounts.	human only	1723
gilani-2017	Manually annotated human and bot accounts.	bot/human	584/903
varol-2017	This dataset contains annotation of 2573 Twitter accounts. Annotation and data crawl is completed in April 2016.	bot/human	1854/2
pronbots-2019	Pronbots shared by Andy Patel	bot only	1255
cresci-2017	A dataset of (i) genuine, (ii) traditional, and (iii) social spambot Twitter accounts, annotated by CrowdFlower contributors. Released in CSV format.	bot/human	6555/2458
celebrity-2019	Celebrity accounts collected as authentic users	bot only	3686
cresci-2015	A dataset of (i) genuine and (ii) fake Twitter accounts, manually annotated. Released in CSV format.	bot/human	662/1537
vendor-purchased-2019	Fake follower accounts purchased from several companies.	bot only	479

As to provide the machine learning model the best possible dataset to train on I gathered many labeled Twitter datasets from around the internet and crawled through all of them to build up the final dataset. The total number of datasets is 12 and some contain a mixture of human and bot account while some contain only one type. A large chunk of them came from the data repository on BotOrNot website where they proved many labeled datasets. They only consist of the user twitter ID and a label ”1” representing bot and ”0” representing human accounts.

The files were used as input files into a custom made Python program which went to each account and gathered the necessary features described in the datatype break down in previous chapter. The final file consists of 7,069 human and 15,438 bot accounts for a total of 22,507 data points for the classifier to train on. The file has 22 user metadata features along with a vocabulary matrix made from the tweet text also provided in the datasets.

The original input files with only user IDs were much larger. However a huge amount of twitter accounts have since been suspended by Twitter or they deleted themselves. This caused many Tweepy errors and make the dataset smaller. Furthermore some of the accounts listed were private which meant we couldn’t get any data from them. These errors accounted for more than half of the dataset not being able to be collected. This is simply due to the fact some of the dataset have been collected years ago giving twitter enough time to suspend them. Nevertheless the amount which was collected will be sufficient to develop a robust classifier.

5.3 Choosing the Classifiers

To find the best classifiers I have conducted multiple tests each with a different classifiers to see which one gives the best accuracy. The best two will be chosen to be further fine-tuned using parameters grid search to find optimal settings. Below you will find the tables with the accuracy scores of each model.

The classifiers: **Nearest Neighbours**(NN), **Random Forest** (RF), **AdaBoost Support Vector Machine** (SVM), **Multinomial Naive Bayes** (MNB), **Gaussian Naive Bayes** (GNB), **Bernoulli Naive Bayes** (BNB).

Table 5.1: User Features Only

NN	**RF**	**AB**	**SVM**	**MNB**	**GNB**	**BNB**
85.21%	90.23%	88.25%	74.76%	71.12%	73.06	66.76

From only using the account features we can see that Random Forest is by far the best exceeding accuracy of 90% which is a good sign. It is closely followed by AdaBoost and NearestNeighbour with both getting and above 85% accuracy. The remaining classifiers had respectable accuracy's with around 70% for all except for Bernoulli Naive Bayes which done the worst with 66.76%.

Table 5.2: Tweet Features Only

NN	**RF**	**AB**	**SVM**	**MNB**	**GNB**	**BNB**
51.94%	69.75%	63.47%	70.35%	68.61%	62.38	66.47

When using tweet only features we are doing text analysis. From the results we can see an overall drop in accuracy across all classifiers. This is mostly due to the text analysis having a way bigger feature set to train on. This also makes it harder to fully clean the text of any useless words. From the results we see that SVM came out on top with just over 70% followed by Random Forest with 69.75% and Multinomial Naive Bayes with 68.61%.

SVM is said to be better with text analysis problems which in this case is correct according to our test results.

Table 5.3: Tweet and User Features

NN	**RF**	**AB**	**SVM**	**MNB**	**GNB**	**BNB**
86.57%	90.94%	87.61%	80.80%	72.09%	60.36	83.73

In the final test I have combined user features and text features into one matrix of features. This was then used to train the same classifier to see if we can get better accuracy's. From the results we can see a small increase in accuracy in some of the models. Random Forest with the highest accuracy of 90.94% which is an increase of 0.71% from the previous best when only using account features. The second best was AdaBoost with 87.61% which was a small decrease in accuracy of 0.64% from the best one in only using account features. The biggest increase was seen in BNB with 83.73% compared to when only using account features with 66.76% and SVM with 80.80% compared to 74.76% when only account features were used.

Overall we from the results we can see that that is clear potential in developing a classifier to detect bot activity on twitter. Furthermore it is also beneficial to combine the account metadata with user tweet features. We have seen an increase of accuracy's in all classifier except AB and GNB which got worse result when combing features.

5.4 The Models

The chosen models are Random Forest and Support Vector Machine(SVM). These two models have shown promising results when doing out-the-box testing on all the classifiers. They have both seen an increase of accuracy when combining features with SVM seeing the biggest of the two. For this reason I will try and further tune these models to hopefully increase the accuracy's further.

5.4.1 Random Forest

The classifier creates a forest of decision trees, each on a different subset of the dataset. It uses averaging to help in preventing over-fitting and increasing the accuracy. The model behaves on a question based way, when reaching a node the model will compare values with the node and decide depending on it if it will continue left or right.

5.4.2 Tuning the Model

To get the best possible results from the classifiers I have done a CVGridSearch on the model. this will take a parameter list and run multiple tests to figure out which combination of settings yields the best results. This should push the model to increase it's accuracy by a few percent. The test will also do cross-validation to ensure that all data is used in training and receives more scoring metrics.

The param_grid contains all the parameters the search will be done on. Sample slit means the amount of internal samples which takes to split the node, estimators are the number of trees that will be generated in out forest. Max depth is the maximum depth of a tree. Lastly mac features is the number of features to consider when performing a split.

The scorers are passed into the GridSearchCV to be used as the metrics for scoring the classifiers. They each will help in deducing which parameters combination yields the best results.

```
param_grid = {
    'min_sample_split': [2, 4, 6, 8]
    'n_estimators': [100, 200, 300]
    'max_depth': [None, 3, 5, 15, 25]
    'max_features': [None, 5, 10, 15, 21]
}

scorers = {
    'precision_score': make_scorer(precision_score)
    'recall_score': make_scorer(recall_score)
    'accuracy_score': make_scorer(accuracy_score)
}
```

5.4.2.1 Final Model Settings

When performing the grid search the ideal parameters used for best results were 800 n_estimators and a minimum sample split of 2. This has helped in pushing the model into the 91% accuracy range.

RandomForestClassifier(n_estimators=800, random_state = 42, min_sample_split=2)

5.4.2.2 Most Important Features

The forest classifier allows for plotting of its most important features which can be found below. We can see that the model has put a lot of weight on favourites_count, statues_count and close ones followers_count and friends_count. On the other hand it found the number of digits in name, user has background image and default profile as least important ones.

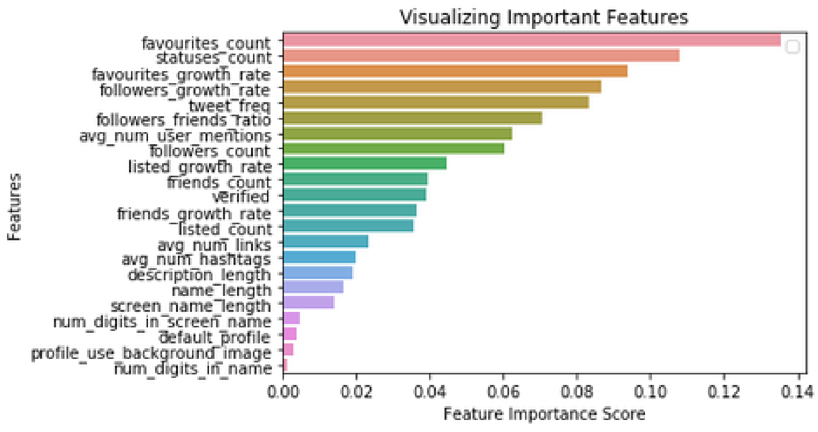


Figure 5.1: A picture of the same gull looking the other way!

5.4.3 Support Vector Machine

Support Vector Machine tries to split the data by created the best hyper-plane between the closest 2 data points. The bigger the distance between the data points and the plane then the better the precision scores and less false positives or negatives.

5.4.4 Tuning the Model

This is the same step as described in Random Forest section. We will be declaring a param_grid which will contains all the relevant parameters we want to test on. The scorers will be again sued as metrics and helping in finding the best results.

```
param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10]
    'gammas': [0.001, 0.01, 0.1, 1]
    'kernel': ['kbf', 'linear']
}

scorers = {
    'precision_score': make_scorer(precision_score)
    'recall_score': make_scorer(recall_score)
    'accuracy_score': make_scorer(accuracy_score)
}
```

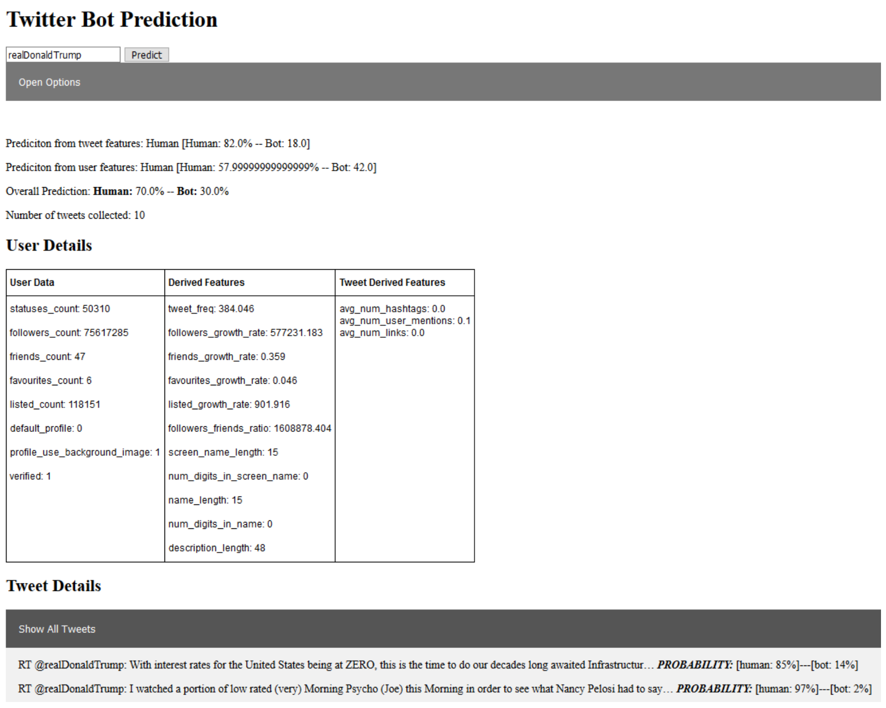
5.4.4.1 Final Model Settings

After some time the final settings can be seen below. These parameters managed to further increase the accuracy by a couple percent.

SVC(C=10, gammas=0.001, kernel='linear')

5.5 The Web Application

Below is the final looking page. This is the second half of the whole project. The web application has been developed so that the trained classifier from above can be implemented on it and give access to user for their own predictions.



Twitter Bot Prediction

Prediciton from tweet features: Human [Human: 82.0% -- Bot: 18.0]

Prediciton from user features: Human [Human: 57.99999999999999% -- Bot: 42.0]

Overall Prediction: **Human:** 70.0% -- **Bot:** 30.0%

Number of tweets collected: 10

User Details

User Data	Derived Features	Tweet Derived Features
statuses_count: 50310	tweet_freq: 384.046	avg_num_hashtags: 0.0
followers_count: 75617285	followers_growth_rate: 577231.183	avg_num_user_mentions: 0.1
friends_count: 47	friends_growth_rate: 0.359	avg_num_links: 0.0
favourites_count: 6	favourites_growth_rate: 0.046	
listed_count: 118151	listed_growth_rate: 901.916	
default_profile: 0	followers_friends_ratio: 1608878.404	
profile_use_background_image: 1	screen_name_length: 15	
verified: 1	num_digits_in_screen_name: 0	
	name_length: 15	
	num_digits_in_name: 0	
	description_length: 48	

Tweet Details

RT @realDonaldTrump: With interest rates for the United States being at ZERO, this is the time to do our decades long awaited Infrastructur… ***PROBABILITY:*** [human: 85%]---[bot: 14%]

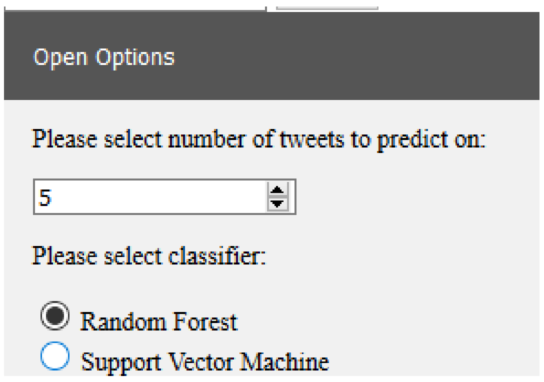
RT @realDonaldTrump: I watched a portion of low rated (very) Morning Psycho (Joe) this Morning in order to see what Nancy Pelosi had to say… ***PROBABILITY:*** [human: 97%]---[bot: 2%]

Figure 5.2: Final web application look

The user can enter a single twitter username for prediction. Once entered the username will be used to extract necessary profile and tweet data for classification. Once application finishes predicting it will output the results on screen and show the probability percentage of the account being human or bot. These can be seen under the "Open Options" menu. You can see the predictions split based on only using their user features, tweet features or both.

The tweet results can be seen further down in another drop down menu which will display the tweet along with the percentage probability it predicted on that tweet. The average of all of them is taken to produce the final overall score for just the tweet features.

Settings Menu:



Open Options

Please select number of tweets to predict on:

5

Please select classifier:

[✓ Random Forest] Support Vector Machine

Figure 5.3: Settings Options

Shown above is the settings drop down menu which will allow the user to change the classifier or the amount of tweet the application extract for analysis.

5.5.1 Error Handling and Data Validation

5.5.1.1 Data Validation

The application will be made up of the following input fields:

- Username (text box)
- Classifier choice (radio buttons)
- Tweets Number (number box)
- Begin Predicting (button)

The text box for username will have to do a few validations to ensure it is a correct username. It will have a maximum length of 15 characters and it can only be made up of alphanumeric characters (letters A to Z and numbers 0-9).

The other input fields are of types which are already validated like the number box only allowing number characters or the radio buttons which restrict the users choice.

5.5.1.2 Error Handling

To ensure that there are no unexpected errors causing crashes I have implemented a variety of *catches* throughout the the software as to expect them. These should alert the user of an error occurring and give the options to continue operation with another username.

5.6 Testing of Application

5.6.1 Introduction

This test plan will be written about the current task I have taken of creating a web application to aid in detection of automated twitter accounts. This section will focus on discussing the scopes of the testing followed by the methods and strategies used in testing the application.

All the testing for these will be planned out in this document and the test logs will be attached at the end of the chapter.

5.6.2 Objectives

I will be using different test logs to try and capture the full range of options the user has when interacting with the software. All major bugs and crashes will be recorded and fixed before any final deadlines to ensure the software operates smoothly. This section will target to eliminate all if not most major bugs so no hard crashes occur when user is interacting.

Moreover I will be testing the usability of the software by bringing in a small tester group as to give an indication of how easy the software is to use. This is a key step and will also be checking if the user can also understand the output provided.

5.6.3 Scopes

The testing will cover 3 main areas.The data validation will check if the application can handle unexpected strings from the user and shows correct errors. The event validation will test if all the even driven inputs trigger the correct events and code, in this case we will only have to check the predict button. The functional acceptance tests will ensure that the software is written to the customers satisfaction.

After these testes have been conducted I will be happy enough to say that the software is free of any major bugs and complies to the specification provided at the start of this dissertation.

5.6.4 Test Items

- Data Validation test Logs
- Event Validation Test Logs
- Functional Acceptance Test Logs

5.6.5 Testing Methods

The testing method I have chosen for this problem is the black box testing method. With it we don't bother checking the code itself or its structure but we focus on testing that we get the correct output once inserted and input. Its similar to how the classifiers work, we don't actually have access to the code - or even if we did we couldn't understand it - but rather we trust the process and only feed it input and check the predictions. This will apply similarly to the whole application as it's main focus is supplying the classifier to the user for predictions.

Some static code review will be done to make sure the application is as best written as possible providing the best possible performance.

5.6.6 Testing Strategy

The test strategy we will be using is the bottom-up testing. With this strategy we will be looking at the smaller less significant parts of the software and then work out way through the software testing more of the individual functions until we can start combining them all together into one single program. This strategy goes hand in hand with object-oriented programming as our program is already made up of multiple different objects that can be tested by themselves. Once they all behave and perform as intended they can be combined.

5.6.7 Data Validation

In data validation we are checking that the application takes all the inputs it should and rejects all the inputs it shouldn't.

Test Case	Test Items	Expected Results	Results
Username	realDonaldTrump	accepts the string	✓
	BBCWorld	accepts the string	✓
	thisstringwilleceedlimit	string too long. software displays error	✓
	EMPTY	can't be left empty, software displays error	✓

Table 5.4: Data Validation log

There were no real problems encountered as we only had to check the username input box.

5.6.8 Event Validation

In this section we are checking if the all the elements like buttons and drop down menus trigger the correct evens like display full menu or starting the prediction.

Control	User Interaction	Expected Results	Actual Results
(Button) ”Predict”	clicks button	software begins prediciton process	✓
(Dropdown) ”Show Tweets”	click menu	software expands menu shwoing all tweet predictions with percentages	✓
(Dropdown) ”Show options”	click menu	software expands menu showing all options	✓

Table 5.5: Event Validation log

5.6.9 Functional Acceptance

In this section we are testing all the features and functions the application should perform like starting the prediction or correctly changing the classifier to use.

Test Case	Expected Result	Actual Result	Comments
user pressed predict button	predicts lable based on collected data	✓	
user switched classifer using radio button	correct classsifer is used as prediction	✓	
user changes tweet number	correect number of tweets is pulled from twitter for prediction	✓	sometimes tweepy only manages to get 1 tweet. this is fixed by repressing the button.

Table 5.6: Functional acceptance log

There was a small bug encountered when the application tries to gather the amount of tweets the user specified. Sometimes tweepy only returns 1 tweet. This is fixed by repressing the predict button which should make tweepy regather the tweets.

Chapter 6

Evaluation and Results

6.1 Introduction

In the following section will discuss the experiments performed with classifiers and will further breakdown the results. Finally I will conduct a user evaluation by getting real users to user the app and answer questions.

6.2 The Experiments

1. What are the differences in accuracy of Classifiers based on year of dataset collection. (**Dataset By Year**)

2. Test accuracy of Classifiers by only using user profile features, only tweet features and both. (**User vs Tweet or Both**)

The first experiment that will be done is one the datasets collected. The account were detected and were collected during different years and as discussed in [27], bots sophistication levels tend to get more complex as years go by and bot makers gain more knowledge on how to avoid the current detection methods. For this reason I have decided to see if there is any significant difference in accuracy given by classifiers biased on the year a dataset was collected.

This will be broken down into three years: 2017, 2018 and 2019. These have been made up from the datasets listed in the implementation chapter in the dataset break down section.

Table 6.1: bot/human split based on year

Year of Collection	# of bots/humans
2015	662/1538
2017	8993/3364
2019	2091/5857

The second experiment will have the already best two chosen classifiers be further tested after some parameter tuning to get the best results. This was the main experiment of the whole project and dissertation to see what was the best model we could get using twitter account metadata and tweet analysis. The best accuracy were seen from Random Forest with above 90%, the other chosen classifier is support vector machine which showed promising results with 80.80%. SVM was however not the second highest but it did have the best accuracy in text only analysis which I believe if correct parameters are chosen could boost the tweet and user features accuracy. The results and discussion will be seen in the following chapters.

6.3 Dataset By Year

Below is the table which shows the accuracy score of of the classifiers trained on dataset from different years.

Table 6.2: Different dataset accuracy's

2015	2017	2019	Accuracy (RF / SVM)
x			93.69% / 92.20%
	x		87.90% / 81.97%
		x	95.55% / 86.44%
x	x		88.30% / 81.83
x		x	93.97% / 84.44%
	x	x	90.20% / 79.75%

Results show us that the individual year of collection doesn't have any patters of accuracy but rather are all different. When mixing some years together we end up with an in-between accuracy of the two data sets. These results may also be affected by the number of account in each dataset as they were not all the same.

6.4 Overall Accuracy by User and Tweet features

In this experiment we are trying to see if either the user or tweet data is a better predictor on their own or are both of them important and needed for stronger accuracy.

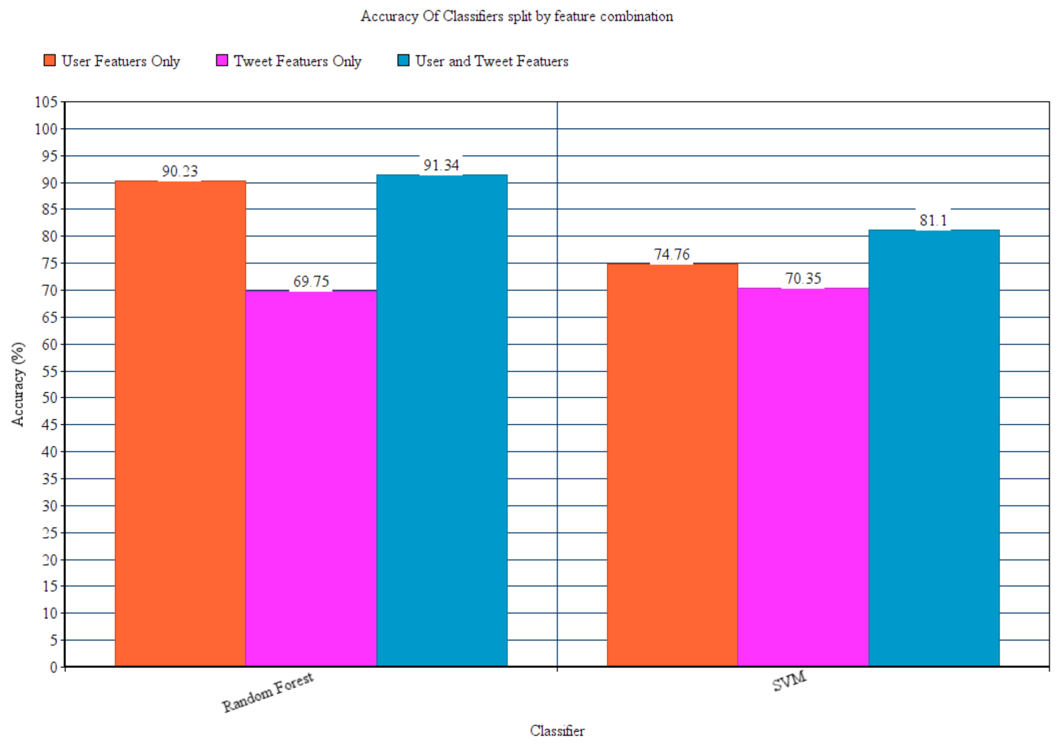


Figure 6.1: Classifier Results

From the graphs we can see that overall Random Forest has the best accuracy across all different features combination. The highest being 91.34% with Random Forest and 81.10% with SVM. We can also notice a similar patter between the two classifiers. Tweet only is always the lowest accuracy followed by user features and for both yielding best results when combining the two together. The improvement may be small but when in 90% range already then any extra percent is a significant increase.

6.5 User Experience Evaluation

The final test will be conducted on the overall usability of the application from the point of view of the end user. To do this I have created a questioner which the user will be fill out once the they goes through the normal interaction on the software. These interaction have been derived from the requirement specification and will consider all the features the application should be implementing.

Table 6.3

QUESTION	ANSWER
Name:	-------------
can you enter a username for analysis?	
can you open options and choose a model for classification?	
can you change the number of tweets to analyse on?	
can you find and press the predict button?	
Did the results display on screen?	
Did the probabilities percentages also display correctly?	
Can you check what tweets have been collected for analysis? (drop down menu at the bottom of page)	
Is the percent probability displayed next to all the tweets?	
Did all the user profile data load correctly into the table?	
Did you find the web application easy to use?	
How would you rate the overall look of the website (1-10)	
Did you understand all the displayed information?	
Did any errors occur?	
Did you find the application helpful in detecting bot accounts?	
Would you ever use such application if available online?	

Due to unforeseen circumstances the university campus was closed until the start of next year. This meant I was not able to do a proper user experience evaluation with questioners presented above. Instead I will describe how I would have conduct the test.

A group of differing students would be picked and explained the overall purpose of the web application. Once understood I would ask them to begin using the application and fill out the sheets as they go along. The questions on the sheet are in a somewhat chronological order so first question is about username input field then followed by the use of options menu.

6.6 Summary Of Results

Overall we seen that half of the initial classifiers showed promising results with accuracy's shows in the implementation chapter. Random forest and SVM were the best choices and were the ones which I took further to tune settings with.

For the first experiment we checked if there is any difference in accuracy based o year of collection. This experiment came from the idea that bot become more sophisticated over time. This may be true in cases where a more in depth analysis is done on the text side of the prediction as this will be the more difficult part to fake when being a bot. My project only turns the words into a number representing of frequencies and importance of each word in document. Both year 2015 and 2019 had the best results with accuracy's over 90%. 2017 was close by with around 87%. This can possibly attributed to the fact that dataset 2017 is made up of more accounts which means the features complexity is larger resulting in a worse accuracy.

After changing the settings both of the classifiers were showing good results with accuracy's ranging around 91% and 81% for SVM. This was an increase on the accuracy's which we got from only using single set of features. This concludes that it is more accurate to combine the user and tweet features for training as they yield the best results.

Chapter 7

Conclusions

7.1 Areas of Further Work

The results of the experiments showed a promising start in the development of a bot detection system for online use. However I encountered a couple of areas which can be worked on further which will be discussed in this section.

The first is the text analysis which is done to the tweet. When training the classifiers I only used one method of representing the text with term frequency - inverse document frequency. This area could be improved in many different ways not only by testing other words representation techniques. The tweets could be analysed for sentiment or the actual structure of the tweet.

The overall accuracy of classifiers was satisfactory with Random Forest reaching 91.34% accuracy when using both user and tweet features. Even thought the model is very good it can always be further fined tuned to help in accuracy.

Another area which could be improved is the general features of an account could have been represented differently. For example I only take the length of the description and use the integer number as a feature. I could have futures analysed the accounts description for using Part-of-Speech tagging to analyse the structure. The description could be checked for the amount of hashtags in it or if it contains any links.

In the case of the application I have thought of expanding it in many different areas as this was only the first functional iteration of the web application. For functionality the web application could have allowed for scanning of chosen users friends and followers to see if a large portion of their network may be bots, this would help in the strength of a given prediction. The application could have used more CSS to make it more user friendly however the functionality was delivered which was sufficient.

7.2 Conclusion

This projects aims were to find the best classifiers which can aid in detecting bot activity on twitter. From the results received we can make a strong case for there being a possibility of getting high enough accuracy for this to be reliable application. The experiment showed that the datasets split by year had a small decrease of accuracy in the more recent years however there was not any significant trends.

Overall the chosen classifier for further tuning showed even better results with Random Forest as high as 91.34% accuracy while Support Vector Machine managed to achieve 81.10%. The web application allows the use to make use of these models by submitting their own choice of usernames to be predicted on. The results are displayed on screen for the user to see.

The final application showed promising results and if this was to be taken further it could become a useful tool in helping people see if accounts they are conversing with are automated.

References

[1] Suad A. Alasadi and Wesam S. Bhaya. Review of data preprocessing techniques in data mining, 2017.

[2] Adam Badawy, Kristina Lerman, and Emilio Ferrara. Who falls for online political manipulation? *The Web Conference 2019 - Companion of the World Wide Web Conference, WWW 2019*, pages 162–168, 2019.

[3] Kristin P Bennett and Erin J Bredensteiner. Duality and Geometry in SVM Classifiers(use for reference).

[4] LEO BREIMAN. Random Forests. *Kluwer Academic Publishers*, pages 5–32, 2001.

[5] David A. Broniatowski, Amelia M. Jamison, Si Hua Qi, Lulwah AlKulaib, Tao Chen, Adrian Benton, Sandra C. Quinn, and Mark Dredze. Weaponized health communication: Twitter bots and Russian trolls amplify the vaccine debate. *American Journal of Public Health*, 108(10):1378–1384, 2018.

[6] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. Temporal patterns in bot activities. *26th International World Wide Web Conference 2017, WWW 2017 Companion*, (April):1601–1606, 2019.

[7] Peter Christen. *Data matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection.* 2012.

[8] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. Detecting automation of Twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing*, 9(6):811–824, 2012.

[9] Stefano Cresci, Marinella Petrocchi, Angelo Spognardi, and Stefano Tognazzi. Better Safe Than Sorry: An Adversarial Approach to Improve Social Bot Detection. 2019.

[10] Clayton A. Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. BotOrNot: A System to Evaluate Social Bots. pages 4–5, 2016.

[11] John P. Dickerson, Vadim Kagan, and V. S. Subrahmanian. Using sentiment to detect bots on Twitter: Are humans more opinionated than bots? *ASONAM 2014 - Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, (Asonam):620–627, 2014.

[12] Juan Echeverria, Christoph Besel, and Shi Zhou. Discovery of the Twitter Bursty Botnet. pages 145–159, 2018.

[13] Jeremy Fields. Botnet Campaign Detection on Twitter. 2018.

[14] VS Subrahmanian Amos Azaria Skylar Durst Vadim Kagan Aram Galstyan Kristina Lerman Linhong Zhu Emilio Ferrara Alessandro Flammini Filippo Menczer Andrew Stevens Alexander Dekhtyar Shuyang Gao Tad Hogg Yan Liu Onur Varol Prashant Shiralkar Vinod Vydiswaran Hwang. The DARPA TWITTER BOT CHALLENGE. 66:37–39, 2016.

[15] Alexandre Kowalczyk. SVM Best Tutorial. page 114, 2017.

[16] Sneha Kudugunta and Emilio Ferrara. Deep neural networks for bot detection. *Information Sciences*, 467:312–322, 2018.

[17] Kyumin Lee, Brian David Eoff, and James Caverlee. Seven Months with the Devils: An in-depth characterisation of Bots and Humans on Twitter. *The Fifth International AAAI Conference on Weblogs and Social Media*, pages 185–192, 2011.

[18] Luca Luceri, Adam Badawy, Ashok Deb, and Emilio Ferrara. Red bots do it better: Comparative analysis of social bot partisan behavior. *The Web Conference 2019 - Companion of the World Wide Web Conference, WWW 2019*, pages 1007–1012, 2019.

[19] Nabil Mohammed, Ali Munassar, and A Govardhan. A Comparison Between Five Models Of Software Engineering. *International Journal of Computer Science Issues*, 7(5):94–101, 2010.

[20] Joy Pollock, Elisabeth Waller, and Rody Politt. Speech and language processing. *Day-to-Day Dyslexia in the Classroom*, pages 16–28, 2010.

[21] Bruno Stecanella. An introduction to Support Vector Machines (SVM).

[22] Andree Thieltges, Florian Schmidt, and Simon Hegelich. The devil's triangle: Ethical considerations on developing bot detection methods. *AAAI Spring Symposium - Technical Report*, SS-16-01 -:253–257, 2016.

[23] E.L. Tonkin. *A Day at Work (with Text)*. Elsevier Ltd., 2016.

[24] Twitter. Twitter Transparency Report. Retrieved from, 2020.

[25] Onur Varol, Emilio Ferrara, Clayton A Davis, Filippo Menczer, and Alessandro Flammini. Online Human-Bot Interactions : Detection , Estimation , and Characterization. (Icwsm):280–289, 2017.

[26] Kai-Cheng Yang, Onur Varol, Pik-Mai Hui, and Filippo Menczer. Scalable and Generalizable Social Bot Detection through Data Selection. 2019.

[27] Kai-Cheng Yang, Onur Varol, Clayton A. Davis, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. Arming the public with artificial intelligence to counter social bots. *Human Behavior and Emerging Technologies*, 1(1):48–61, 2019.

Chapter 8

Appendices

8.1 Project Proposal

Initial Project Overview
SOC10101 Honours Project (40 Credits)
Title of Project: Developing an NLP text Classifier for detecting Fake/Bot Accounts on The Twitter Feed

Overview of Project Content and Milestones
It seems that people prefer to discuss and talk to each other over the internet more and more. This makes it important to keep places like twitter free of any malicious manipulation. This may come in many forms like government propaganda, spreading misinforming or taking sides in elections. This is usually never accomplished with one bot, but rather a network of bots which may communicate to help and support each other to remain legitimate looking by retweeting or liking content.

The main task will be to research and develop the best NLP classifier that accurately identifies twitter accounts that have suspicious activity. A suspicious account may be flagged up by one of many reasons but usually needs multiple to have any credibility. Some of these are: recent account creation (no profile picture or bio), numbers in the name which may be used to generate random accounts, the tweeting speed and frequency may also be a pointer towards bot accounts. I will look at similar apps and research published papers on the topic.

I will develop a front-end software which will allow users to enter a twitter handle.

The profiles tweets will be analysed using the classifier to check for suspicious activity. The profile will be given a “score” which will indicate how likely is it that the profile is operated by an AI. A breakdown of reasons why the account is/isn’t depicting suspicious activity will be displayed.

The Main Deliverable(s): Text classifier which can be applied in analysing twitter feed. A software which will allow you to enter any twitter handle (username). A dissertation

The Target Audience for the Deliverable(s): Any person who wants to know if the twitter profile they are conversing with is in fact a real person. For political research, would allow see how many users in each dataset are controller by an AI. By twitter to flag accounts which have a high chance of being a bot account. This would violate twitters TOS and the account would be shut down.

The Work to be Undertaken: Research the already existing bot detectors and analyse what type of data they display. Investigate which twitter activities or characteristics a bot account most frequently has. Research the best NLP technology and develop an accurate classifier to be used in detecting bot accounts. Compare different classifiers to figure out the best one. Develop a software which the user can use to enter twitter handles and analyse the profiles,

Additional Information / Knowledge Required: Some NLP technology will need to be learned (NLTK/spaCy/gensim) Improve my python knowledge which will be used to develop the model. (datasets) A website which has many different datasets on twitter accounts. Could use different datasets to refine a classifier and improve the accuracy.

Information Sources that Provide a Context for the Project: Example of websites which do a similar tasks (TweetBotOrNot)(botcheck)(botOmeter) (Link 1) Twitter has an increasingly difficult time finding and terminating bot accounts. Thousands of apps also authorised on twitter have been identified as being links to spam or malware. Twitter has even asked for help from experts outside the company asking to propose ideas on how to keep twitter conversation healthy. (Link 2) In the American Journal of Public Health, it is stated in the results that “polluter” accounts spread misinformation on vaccinations while “Russian trolls” prompted the discord on the topic while taking sides. This has led to vaccines being put under scrutinization and some go as far as to say it causes harm or even autism. In 2017 Social Science Computer Review published a paper just over a year after the Brexit vote stating, approximately 13,500 twitter accounts vanished just after the ballot were counted. These accounts were all active and tweeting about the referendum leading up to it. This has led people to believe they may have made an impact by injecting themselves in conversations to sway people's opinions on the matter.

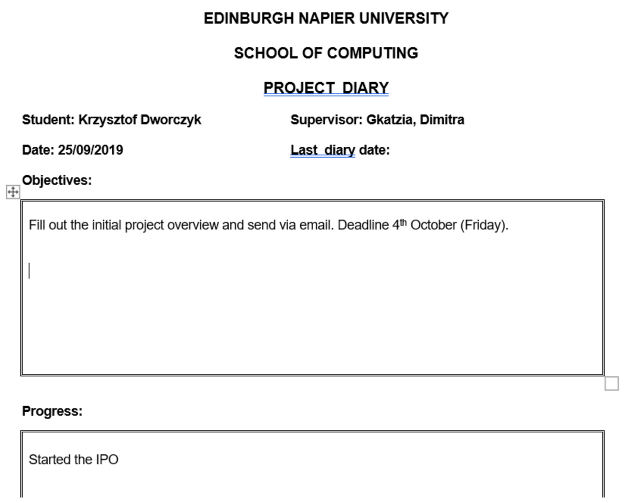
The Importance of the Project: With twitter being the biggest discussion platform, it is important to police it and ensure that malicious accounts don't interfere with the conversation. Twitter has huge influence and reach, it is used by nearly every country which means if you found a way to control the conversation, you could control what people talk. This could sway people opinions. With the recent re-surge of anti-vaccination and meddling in election by foreign governments, it is very important to be able to detect when accounts are trying to push certain agendas. These accounts may be in a cluster of accounts trying to change people opinions.

The Key Challenge(s) to be Overcome: Becoming proficient enough at language processing techniques to be able to develop an accurate enough model to be used. Figuring out the key characteristics that may point to a bot account. I will need to find a way to use my created classifiers in a different environment like Visual studio. The front end would be implemented using C#.

References 1. Bastos, M. T. and Mercea, D. (2019) 'The Brexit Botnet and User-Generated Hyperpartisan News', Social Science Computer Review, 37(1), pp. 38–54. doi: 10.1177/0894439317734157. 2. Broniatowski, D. A. et al. (2018) 'Weaponized Health Communication: Twitter Bots and Russian Trolls Amplify the Vaccine Debate', American Journal of Public Health, 108(10), pp. 1378–1384. doi: 10.2105/AJPH.2018.304567. 3. https://blog.twitter.com/official/en_us/topics/company/2018/twitter-health-metrics-proposal-submission.html, 2018

8.2 Diary Entries

8.2.1 Week 2



EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Krzysztof Dworczyk	**Supervisor: Gkatzia, Dimitra**
Date: 25/09/2019	**Last diary date:**

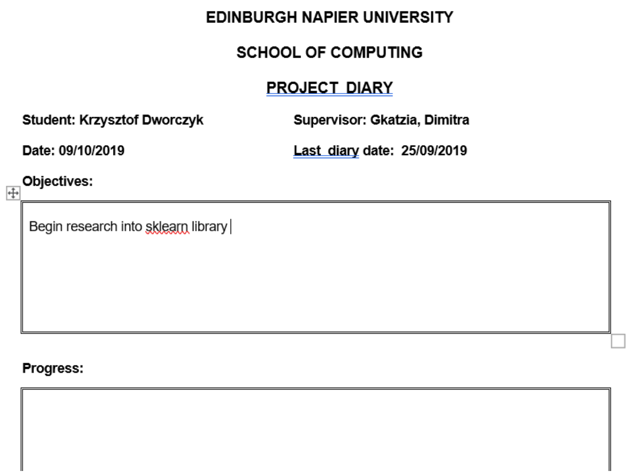
Objectives:

Fill out the initial project overview and send via email. Deadline 4th October (Friday).

Progress:

Started the IPO

8.2.2 Week 5



EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

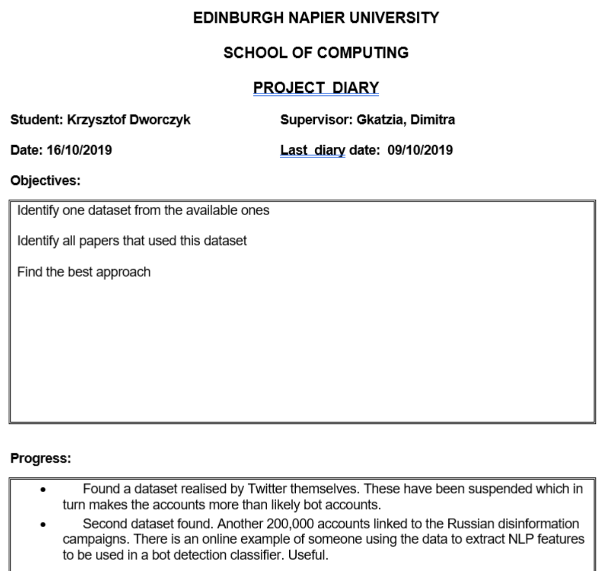
Student: Krzysztof Dworczyk	**Supervisor: Gkatzia, Dimitra**
Date: 09/10/2019	**Last diary date: 25/09/2019**

Objectives:

Begin research into sklearn library

Progress:

8.2.3 Week 6



EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Krzysztof Dworczyk	**Supervisor: Gkatzia, Dimitra**
Date: 16/10/2019	**Last diary date: 09/10/2019**

Objectives:

Identify one dataset from the available ones

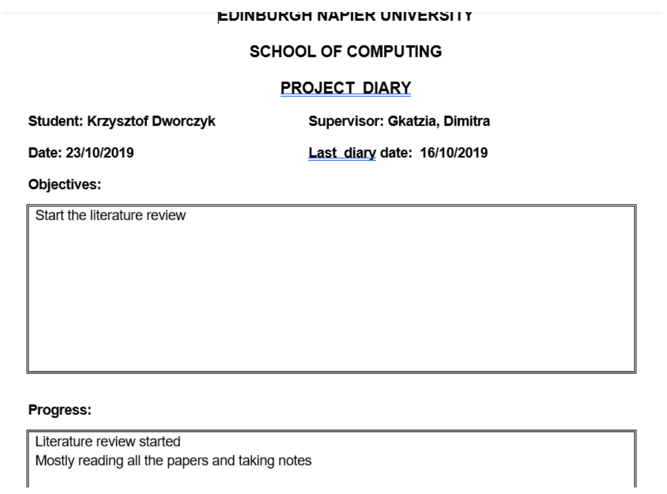
Identify all papers that used this dataset

Find the best approach

Progress:

- Found a dataset realised by Twitter themselves. These have been suspended which in turn makes the accounts more than likely bot accounts.
- Second dataset found. Another 200,000 accounts linked to the Russian disinformation campaigns. There is an online example of someone using the data to extract NLP features to be used in a bot detection classifier. Useful.

8.2.4 Week 7



EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Krzysztof Dworczyk	**Supervisor: Gkatzia, Dimitra**
Date: 23/10/2019	**Last diary date: 16/10/2019**

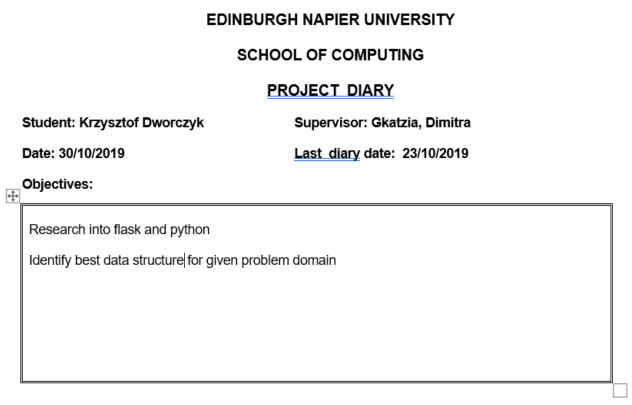
Objectives:

Start the literature review

Progress:

Literature review started
Mostly reading all the papers and taking notes

8.2.5 Week 8


EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Krzysztof Dworczyk	**Supervisor: Gkatzia, Dimitra**
Date: 30/10/2019	**Last diary date: 23/10/2019**

Objectives:

Research into flask and python

Identify best data structure for given problem domain