The Truth about Twitter

Student Name:	Student ID:	Supervisor Name:
Max MacDonald	C15740661	Cindy Liu

Link to Software Repository:	https://github.com/TheBeardBeatsAll/The-Truth-About-Twitter

Table of Contents

Table of Figures

1. Project Statement

Social media sites, such as Facebook, Instagram and Twitter, have been flooded with numerous types of fake accounts in the past few years and each of these sites have an urgent need to be able to tell the difference between a real user and a bot user.

The goal of this project is to create a web application that users can use to find out the likelihood that a Twitter account is real or a type of bot account. Users of this application will be able to input a Twitter handle that will be used to hit the Twitter API, retrieving specific information about the account. This data will be passed into a machine-learning model which has been trained on a dataset of real and bot accounts. The model will then decide whether the account is real or not, displaying it's answer to the web front with relevant statistics.

2. Research

This section will cover all research done for this project including background research on Twitter and bot accounts, applications or solutions akin this project, all technologies researched for this project, research into data science and its sub topics big data, data mining and machine learning and finally the results of all this research such as chosen technologies and areas that will prove challenging.

2.1. Background Research

Twitter, a free social networking microblogging service, is one of today's leading digital platforms with 326 million active users worldwide in the third quarter of 2018.[1] Registered users can broadcast short posts called tweets which can be liked, reposted or retweeted and responded to by other users. Just like other social media platforms such as Facebook and Instagram, Twitter has and still is facing a massive problem with fake or bot accounts. Estimates place the percentage of bot accounts on Twitter anywhere from 9 to 15% of the total user count.[2]

2.1.1. What is a Twitter Bot Account?

A Twitter bot account is an account that is controlled by a software application, via the Twitter API, which will automatically generate and publish new tweets, follow specific users, retweet other tweets and liking specific sets of tweets all based on content or hashtags included, all depending on the settings of the controlling application.[2]

These bots can perform tasks at a much higher rate than a human user can and as such push out more content or tweets in the same timeframe, some even working around the clock. Bot accounts on other platforms are like this with any differences being based on the platform differences.

2.1.2. Types of Bots Accounts

There are many different types of bot accounts from helpful and informative ones tweeting spiritual wellbeing tips to ones which retweet tweets that push extreme ideologies to advertising accounts which are set up to tweet content about specific brands, products or services at certain times of the day.

Some accounts are even used to boost a person's fame or influence on Twitter by following that person's account and can be bought in packages. This is a massive industry in and of itself reportedly being a $40 to $360 million-dollar business annually.[3] Major celebrities such as 50 Cent and brands like Mercedes-Benz have come under scrutiny for possibly engaging in this practice.[4]

Then there is the complexity of the software applications behind these accounts. Older and more traditional bot accounts tend to be easier to detect as they follow much simpler patterns in their activities while newer social bots need far more complex algorithms to detect as they are set up to masquerade effectively as human accounts by mimicking human behaviour better.

2.1.3. Importance of Identifying Bots Accounts

The main reason why it is so important to be able to identify, unless the account states so itself, whether an account is real or not is the erosion of trust that can occur due to the accounts activities. If the account is followed by one million other accounts, even if most of them are fake themselves, and posts something that, while untrue, pushes a narrative that certain groups would be

inclined to believe then that post can gain a lot of traction and spread quickly all over Twitter and beyond to other social media platforms causing untold damage.

Many individuals or groups wish to affect the perception of specific events or entities through Twitter and this ranges from boosting their own profiles through fake followers as mentioned above to trying to influence public campaigns such as the 2016 US Presidential elections. Studies have estimated that in the lead up to this election, a fifth of all Twitter traffic related to the election came from a legion of bots.[5] That much traffic would have had a massive influence on people's views and how they voted and in turn the outcome of the election.

If a bot account is masquerading as a real human then, since it is inherently trying to deceive us, it is highly unlikely much good can come of its sustained existence and as such the sooner it is detected and shutdown the better.

2.1.4. Important Characteristics of Bot Accounts

When trying to identify if an account is a bot or not there are some key characteristics that can help [6,7] :

- How often per day an account tweets can lead to suspicions as this is a hallmark of automation.
- How anonymous that account is trying to be, does it have a profile picture and if so is it of a person? Same for the background picture. Does bio help identify them or add to their anonymity? Is the account handle just an alphanumeric scramble?
- Links in the bio as some bots' purpose is to redirect people to certain websites or have them download malicious software without them knowing although not complete indicative as some people do put links in their bio for example a link to their company's home page.
- Abnormal posting hours as for example an account that identifies as a British man living in London but is posting 9-5pm Moscow time.
- Generic bio or lack of one as the programs which create these bots are not set up to make completely unique bios.

- Lack of followers or high number of followers.
- Ratio of how many other accounts an account follows to how many follow it.

2.2. Alternative Existing Solutions to Your Problem

This section explains how Twitter deals with bot accounts and looks at a similar application to this project, Botometer.

2.2.1. Twitter

It has only been in the last few years that Twitter has taken the detection and suspension of bot accounts seriously. Brexit and the US Presidential elections were the deciding factors as the activities of bot accounts in the lead up to these proved to be a liability for the company. After an internal investigation, Twitter announced it would not be selling any more advertising to Russia media outlets Russia Today and Sputnik as these organisations were found to have interfered with the Presidential election on behalf of their government.[8]

Twitter has also been quiet active this year in detecting and shutting down bot accounts, between May and July around 70 million fake and suspicious accounts were shut down, same in October to a bot network of a few hundred accounts, that were involved in a coordinated campaign to defend Saudi Arabia's Government's role in the disappearance of Jamal Khashoggi, and most recently in November around 10 thousand more, that were all aimed at discouraging Americans to vote in the midterm elections.[9,10,11]

While the company has been trying, it is not an easy fight as they will always be on a reactive footing rather than a proactive one since the creation and running of bots, which are constantly evolving, can be automated but their large-scale detection relies on human intervention. This combined with the sheer volume of users and content through the site makes it a daunting and never-ending task.[12]

2.2.2. Botometer

Botometer is a joint project between Indiana University Network Science Institute (IUNI) and the Center for Complex Networks and Systems Research (CNeTS). It employs a machine learning algorithm trained to classify an account

as real or bot based on a labelled dataset comprised of over 10 thousand. It uses the Twitter REST API to gather public data on an account and then passed to the Botometer API which "*extracts about 1,200 features to characterize the account's profile, friends, social network structure, temporal activity patterns, language, and sentiment*".[13] These are passed onto its models to compute the various scores which in turn go towards the overall score.

It's web front allows a user to check the activity of a Twitter account, after having the user's Twitter account authorised, and gives it a score, out of 5, based on how likely the account is to be a bot with the closer the number is to 5 the more likely it is. There is also an option to check that accounts followers and the accounts it follows as well. It is simple, easy on the eye and informative

I used my own Twitter account to test it and the results are shown below. As you can see it rates my account with a bot score of 4.6/5 and a Complete Automation Probability (CAP) of 83% which is the probability that this account is fully automated. I set my Twitter account up a few years ago, followed some people, sent out one tweet and then completely ignored it so it is not surprising that Botometer's models gave back these results even if they are wrong.

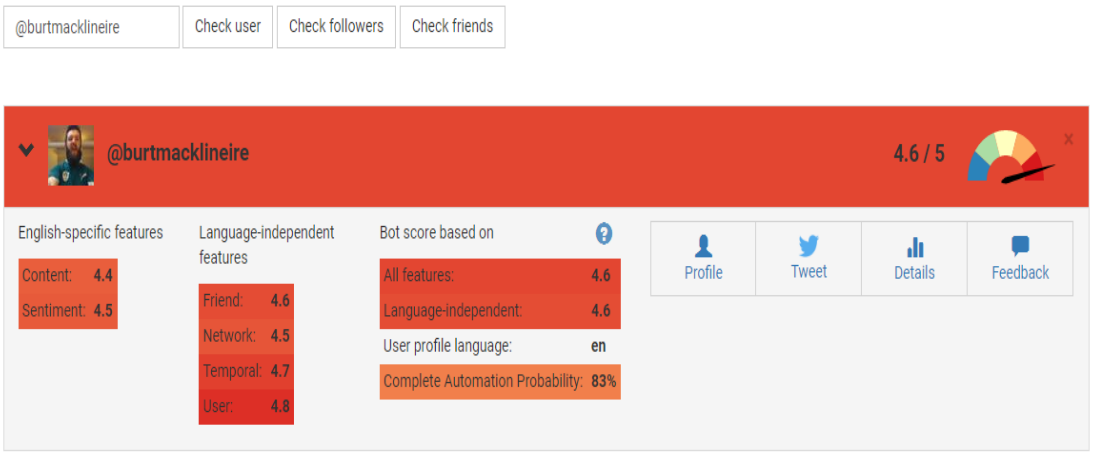

Figure 1 Botometer GUI [13]

2.3. Technologies Researched

This section deals with all research into the various possible technologies that could be used in this project and their benefits and limitations.

2.3.1. Technologies for Data Mining & Machine Learning Models

2.3.1.1. R

R, a GNU project, is a programming language and environment for statistical computing and graphics. It is a variation on the S language and can run code from other languages such as C, C++ and Foltran. It has a wide and enthusiastic community worldwide ensuring there is plenty of support for beginners and its functionality can be extended through numerous packages found online. It has a wide, coherent and well-developed suite of facilities for data handling, storage, data analysis and graphical displays.[14]

Even with all this it does have its limitations such as memory management, R can consume all available memory, since some packages are created by normal users they might not always be up to industry standard and a basic knowledge of statistical vocabulary is needed as it was written by statisticians for statisticians.

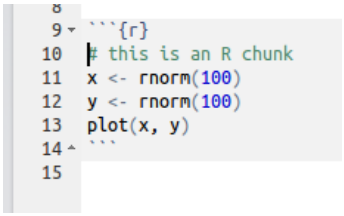

Figure 2 R code example

2.3.1.2. Python

Python is an interpreted, high level programming language that places a lot of emphasis on code readability. It is Open Source, friendly and easy to learn with one of the largest communities in the programming world.[15] It also has a wide variety of packages covering nearly any topic a user might need or need, entire frameworks that can be used to get a project up and running quickly and simply and is supported across multiple platforms and systems.

It does have its downsides though, due to the fact it is compiled at run time it can be quiet slow running, it is also not a good choice if mobile development is at the core of your work or if your project is a game with high-end graphics.

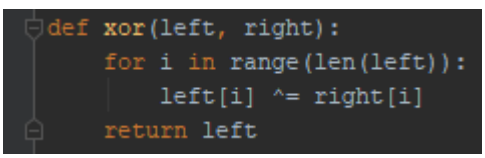

Figure 3 Python code example

2.3.1.3. PyCharm

PyCharm is a Python IDE developed by JetBrains that includes intelligent code compilations, error checking and quick fixes and easy project navigation.[16] The professional version being made available to students, allowing access to many great other features such as starting a project with a framework already in place, database and SQL support and a Python Profiler.

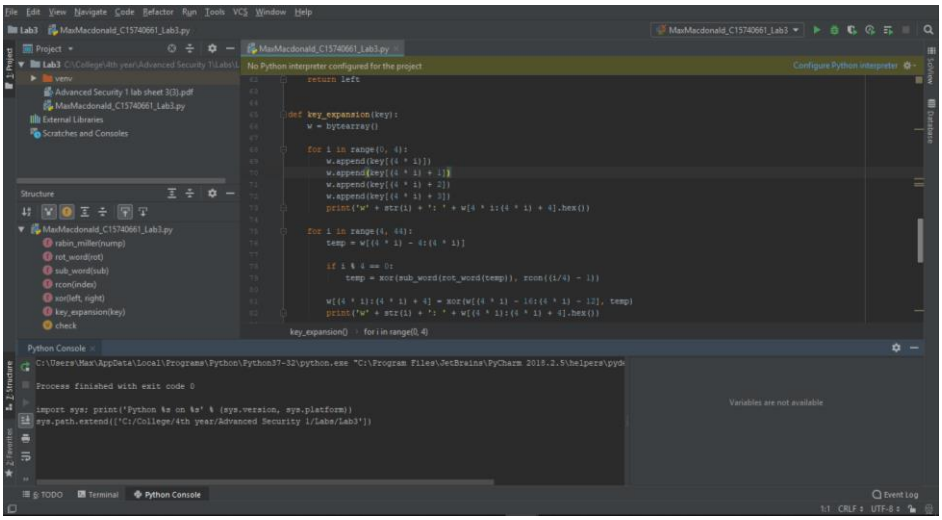

Figure 4 PyCharm IDE

Below are a few integral Python libraries for this project if it is to be used.

2.3.1.3.1. Scikit-learn

Easily the most important and fundamental library to this project, Scikit-learn facilitates machine-learning for users of all levels by supporting various classification, regression and clustering algorithms including gradient boosting, random forests, support vector machines and k-fold cross validation.[17] It allows a user to easily create models, run them and compare their accuracy scores.

```
from sklearn.pipeline import Pipeline


text_clf = Pipeline([('vect', CountVectorizer()),
                     ('tfidf', TfidfTransformer()),
                     ('clf', MultinomialNB())])
```

Figure 5 Scikit-learn code example

2.3.1.3.2. Pandas

The Pandas library provides high-performance, accessible data structures, such as Series and Data Frames, and data analysis tools. Data Frames are two-dimensional arrays while Series are only one-dimensional, and both offer huge

array of features across them such as easily sorting them, iterating through them, searching across them for a count of specific entries or gaining stats on each column like mean or mode.[18]

```
import pandas as pd


features = pd.read_csv('./data/feature_names.txt', header=None)
dataset = pd.read_csv('./data/dataset.txt', header=None)

cat_report = []
cont_report = []

# Go through the columns of the dataset one by one skipping id
for position in range(1, int(features.count())):
    feature_name = features[0][position].capitalize()
    count = int(dataset[0].count())
    card = dataset.apply(pd.Series.nunique)[position]
```

Figure 6 Pandas code example

2.3.1.3.3. NumPy

The NumPy library offers numerous features for scientific computation and works well with the Scikit-learn library. It offers a powerful N-dimensional array object which can be used as an efficient container of generic data, as well as several sophisticated functions and tools.[19]

```
import numpy as np


a = np.array([1, 2, 3])
print(type(a))
print(a.shape)
print(a[0], a[1], a[2])
a[0] = 5
print(a)
```

Figure 7 NumPy code example

2.3.2. Technologies for Web Application

2.3.2.1. Flask

Flask is a web micro-framework for Python that provides users with a simple and effective core of tools, libraries and technologies to build a web application while also allowing it to be easily extended.[20] This has its benefits, as it is light with little need to keep an eye out for security bugs, but also has its limitations as the user will still have to do a lot of work themselves or increase the list of dependencies within the project.

2.3.2.2. Django

Django is a full web framework for Python that enables rapid deployment and elegant, practical design. It was built by skilful developers that abstracted

much of the work required to get a web application of the ground such as managing views and templates, URL endpoints and security features, allowing users to focus on the nuts and bolts of their application instead.[21]

2.3.2.3. Python to Twitter

There are numerous Python libraries or wrappers that can connect to and gather data from the Twitter API such as Tweepy, Twython or Python Twitter. As Tweepy provides some great documentation and examples and is brilliantly supported I will be starting with that library.[22] If it does everything I need of it I will not need to use any of the other libraries.

2.3.3. Technologies for Web Server

As Django and Flask web hosting only truly fulfil the role of development servers another web hosting service must be chosen for deployment onto a production server.

2.3.3.1. Apache HTTP Server

Apache HTTP Server is a free and open-source HTTP server built to operate on numerous different operating systems such as UNIX, Windows or Mac.[23] It is developed and maintained by an open community of developers, has a strong community of users willing to help first timers and is the most widely used web server in the market today. It does have its restrictions though: a strict updating policy must be put in place and the ability to modify its configuration can potentially cause a serious threat to the security of the web application.

2.3.3.2. Heroku

Heroku is a cloud platform as a service (PaaS) that allows users to deploy applications onto its servers and supports several programming languages including Python. It "*makes the processes of deploying, configuring, scaling, tuning, and managing apps as simple and straightforward as possible*" enabling developers to focus on building their app.[24] With this though comes a lack of control as the exact configuration of an application is set by them and if there is a high volume of data traffic then there is a premium charged.

2.3.3.3. Amazon Web Services

Amazon Web Services (AWS) provides on-demand cloud computing platforms to users on a paid subscription basis.[25] This means that for many

users it eliminates capacity constraints while mitigating the costs involved as well as adding in global reach and scalability. It is a high-tier grade service, but you are also paying for it unlike many others. It does offer a first-year free tier of all its services for first time customers and for students and educators there is an AWS Educate account that gives credits enabling hands on experience with their services.

2.3.4. Technologies for Version Control

2.3.4.1. Git

Git is an open source distributed version control system that I am already quiet familiar with from use in previous college years. It is free and easy to use and learn and can be run from the Git Bash client or from its integration in PyCharm, making it even easier to use and track changes in the process. Using either of these ways, it is simple to connect to GitHub, a web-based hosting service for Git repositories and ensure that a project is backed up with required access given to specific team members as well as giving public access to view the project and its code base.[26]

2.3.4.2. Mercurial

Mercurial is a free, distributed source control management tool that prides itself on how fast and powerful it is, it claims it can handle any project no matter the size or type. It is easy to learn and offers an instinctive interface. It is platform independent and extensible. For Mercurial, “history is permanent and sacred.” It only allows the rollback of the last pull or commit although there are extensions if more is needed.[27]

2.3.5. Technologies for Data Storage

2.3.5.1. MySQL

MySQL is an open source relational database management system and one of the most popular systems in the world due to how easy it is to use, its nature as a relational database and how much investment and innovation has gone into it. It allows for powerful joins as well as standard features such as triggers, stored procedures and cursors.[28]

Due to its acquisition by Oracle though there have been some negatives: It is no longer completely open-source as some modules for it are now closed-source and it is no longer community driven.

2.3.5.2. PostgreSQL

PostgreSQL is an open source object-relational database management system with a big emphasis on extensibility and creating features which safely store and scale the most complicated data workloads.[29] It essentially is a combination of relational and NoSQL databases, giving the best of both worlds through its extensions. It is highly scalable and supports JSON

Even with this it has some drawbacks: It's documentation has been known to be spotty and its configuration can be confusing to an inexperienced eye.

2.3.5.3. MongoDB

MongoDB is a free and open-source distributed NoSQL, or document, database that is scalable and flexible. It stores data in JSON- like documents which can be of any desired structure, removing the need for schemas, as in relational databases and allows for powerful ways to access and analyse data using Ad-hoc queries, indexing and real-time aggregation.[30] What is given up for this is the lack of functions or stored procedures as well as loss of strength in terms of ACID (Atomic, Consistency, Isolations, Durability).

2.4. Other Relevant Research Done

This section covers all other relevant research done for this project. Research into approaches and methodologies will be dealt with in their own section later.

2.4.1. Big Data

Over the last few decades the amount of data we have gone from using has increased from kilobytes to megabytes to gigabytes and has now hit terabytes in the everyday home. Many businesses have gone beyond this and are now dealing in petabytes or even exabytes of data.

These large datasets, both structured and unstructured, that are beyond the scope of traditional techniques to process due to their size or complexity and

are used heavily within the domain of data science are what have been termed: Big Data.[31]

Organisations the world over have been investing into this area in the last number of years as the results that can arise from proper storage and use, through data mining and data analysis projects, of Big data can lead to massive returns or scientific breakthroughs.

2.4.2. Data Mining

Data mining is the process of detecting anomalies, correlations and patterns within Big data to make predictions using a wide range of methods including various machine-learning algorithms.[32] There are various project models that can be used, although all of them are built upon the same foundation of stages:

2.4.2.1. Data Acquisition

At the beginning of every data mining or analytics project, the first stage of the project is to ensure there is data to use. Where this data comes from varies from project to project as it may come from an inhouse databases or from surveys carried out with a business's customer base.

The data used is referred to as a dataset with each row in the dataset being an instance of the data and each column being a descriptive feature of the data.[32]

2.4.2.2. Data Understanding

Understanding the business logic and context behind the acquired data and having a base knowledge of the project's domain are integral parts of any data mining project as these help to make sense of the relationships between features and enable easier selection of machine learning algorithms and improving their accuracy.

2.4.2.3. Data Preparation

This stage deals with the pre-processing of the data to ensure it's in the correct state to be used for various business purposes such as in a machine-learning algorithm or data analysis. It includes the sub-stages of data cleaning and feature selection.

2.4.2.3.1. Data Cleaning

Data cleaning is the process of finding and removing entries in the data that has been either entered or formatted incorrectly.[32] Without proper data

cleaning, when passing the data into a model various errors can arise, leading to program failure or completely inaccurate results.

2.4.2.3.2. Feature Selection

In any data mining project, the aim is to produce accurate predictions as efficiently as possible. To do this, we want to minimise the number of features without affecting the accuracy.[32] Therefore, feature selection is a key component that must be carefully deliberated and decided on.

2.4.3. Machine Learning

Acquiring and preparing large sets of data is only part of the battle, the next major stage is to be able to detect patterns within this data and then make predictions from this. This is the core of Machine Learning, enabling us to extract significant insight from Big Data through complex, mathematical algorithms with minimal human intervention.[33] These algorithms are trained on sub-sets of the data to grow more accurate in their predictions.

2.4.3.1. Classifiers

Machine learning classifiers are divided into two sections:

- ***Unsupervised Learning***
  - A classifier is given a set of inputs without any outputs known. It learns by itself, through specific methods, what outputs it should prescribe to each input.
  - Uses of Unsupervised learning algorithms:
    - Find hidden patterns within data
    - Face recognition software
  - Examples of Unsupervised learning algorithms:
    - Clustering
    - Artificial Neural Networks
- ***Supervised Learning***
  - A classifier is given a set of inputs with all outputs known. Using these it learns what outputs to prescribe to any future inputs.
  - Uses of Unsupervised learning algorithms:
    - Predicting Football scores based on previous years data

- Selection of advertising to be displayed to specific users

- Examples of Unsupervised learning algorithms:
  - Support Vector Machines
  - K-nearest Neighbour
  - Naïve Bayes

2.4.3.1.1. Unsupervised Learning Algorithms

2.4.3.1.1.1. Clustering

There are multiple different types of clustering algorithms such as K-Means Clustering and Hierarchal Clustering. All of them revolve around grouping the data based on each input's feature similarity.

The difference between algorithms between, for example, K-Means Clustering and Hierarchal clustering is that the former separates the data points iteratively into K clusters based on the features of the data while the latter considers each data point a cluster then identifies the clusters that are closest to each other and merging them, while taking note of the hierarchal relationship between them, and so on until only one cluster remains with one large hierarchy.[32]

2.4.3.1.1.2. Artificial Neural Networks

Artificial Neural Networks are "*biologically inspired computer programs designed to simulate the way in which the human brain processes information*".[34] They detect patterns and relationships in data and from this infer knowledge and grow from their experience, learning to better classify data or perform tasks.

Using artificial neurons, the computerized version of a brain cell, a network is formed by connecting the output of specific neurons to the input of other neurons, forming a directed, weighted graph. A neurons weights and activation functions can be tuned over the learning process to increase the networks accuracy.[34]

2.4.3.1.2. Supervised Learning Algorithms

2.4.3.1.2.1. Naïve Bayes

Naïve Bayes classifiers belong to the family of probability-based classifiers and are based on Bayes' theorem with the added assumption of

conditional independence between all the features in the data.[33] This added assumption allows for the model to drastically reduce the amount of probabilities it must compute.

While this is quite a leap of faith to make, it still results in a robust model that delivers strong results and, when coupled with its scalability, efficiency and simplicity, is the reason it is normally the starting point for most data mining projects.

2.4.3.1.2.2. Support Vector Machine

A support vector machine classifier belongs to the family of error-based classifiers. It maps all the data as points in an N-dimension space, N being the number of features, and then tries to find a hyperplane, or decision boundary, that distinctly classifies the data points.[33]

It tries to maximise the distance between the hyperplane and data points from both classes. Those points closest to the hyperplane are called support vectors and have a significant impact on its placement.

New points are mapped to this space and classified depending which side of the hyperplane they belong to. It has a high degree of accuracy, takes up less computation power than other algorithms and can be used for both regression and classification task.

2.4.3.1.2.3. Linear Regression

Linear Regression is an algorithm used to model the relationship between two or more continuous variables, or features in this case, by fitting a linear equation to their data.[33] Understanding these relationships can help with fine tuning accuracy as well as using the fitted equation to make predictions. Unlike Naïve Bayes it is used to compute a numerical value rather than predict a class type.

Two Types of Linear Regression:

- Simple: Gives us the relationship between one explanatory variable and one dependent variable.
- Multiple: Gives us the relationship between multiple explanatory variable and one dependent variable.

2.4.3.1.2.4. K-Nearest Neighbour

The K-Nearest Neighbour (KNN) is a simple, non- parametric classifier and belongs to the family of instance-based classifiers and as such keeps has either no or a very small training phase to it.

To classify new data points, the feature similarity of its k-nearest neighbours is used, with the new data point going to the class with the majority count.[32] While it makes no assumptions about data and is versatile it can be computationally expensive and sensitive to irrelevant data.

2.4.3.2. Training & Testing

Once a dataset has been fully prepared for use in a machine learning algorithm it must be divided up into training and testing datasets. The training dataset is a sub set of the original dataset used to train the model while the testing dataset is what is left.[32]

The training dataset is passed through a working model, with the results compared against the actual outcomes enabling the accuracy of the model to be measured. There are various ways that the base dataset can be divided up into training and testing datasets.

2.4.3.2.1. Holdout

This is the most basic division of the original set into training and testing sets with the partitioning of the original into two mutually exclusive sets. The split is usually takes a 2:1 ratio. The main problem with this method is that as more training data is used there is less testing data to be used. Ideally you want both the training and testing sets to be as large as possible.[33]

2.4.3.2.2. K-fold Cross Validation

Using this method, the original set is divided up into K partitions of equal size. Then for each partition, that partition acts as the testing set with the remaining partitions becoming the training set. A model is fitted using this training set and evaluated using the testing set. The model is discarded with the results being held onto before moving onto the next partition.[32]

This method deals with the main issue of the Holdout method, ensuring the entire dataset is used for both training, each partition being used K-1 times, and testing, each partition used once, with the results being of significant use at the end for evaluation.

2.4.3.3. Evaluation

After a model has been created and data run through it, results will have been produced. The accuracy of these results must be measured carefully. Only by truly understanding the accuracy of the results and what influenced it will someone be able to improve the model and its accuracy.

2.4.3.3.1. Classification Accuracy

Simply put, the accuracy of a model is the amount of predictions it got right. Put into formulaic terms: Accuracy = No. of correct predictions / Total no. of predictions.[33] This in and of itself is not enough in terms of detail for a proper model evaluation and as such other methods must also be employed.

2.4.3.3.2. Confusion Matrix

A confusion matrix is a table layout for the visualisation of the performance of a model. The totals of correct and incorrect predictions are calculated and broken down by class. These values are placed into a matrix with predicted across the top and expected down the side.

When looking at a two-class instance or one class against all the others this matrix will then hold the values for True Positives, False positives in the 1st row and False Negatives and True Negatives in the 2nd row.[33] This data holds much more meaning then the previous method of evaluation and can help in knowing what part of the model needs to be tuned to gain a better accuracy level.

2.4.3.3.3. F1 Score

The F1 score is another method of measuring a model's accuracy. It is obtained by computing the weighted average of the Recall and Precision.[33] The closer this score is to 1 the more accurate the model is.

F1 = 2*(Recall * Precision) / (Recall + Precision).

Recall is found by dividing the total correct predictions by the sum of the total correct predictions and false negatives.

Recall = TP/ (TP + FN)

Precision is found by dividing the total correct predictions by the sum of the total correct predictions and false positives.

Precision = TP/ (TP + FP)

2.5. Resultant Findings and Requirements

This section will deal with what technologies I have chosen to use in my project and why, what datasets I will be using to create my machine-learning models and any challenges I foresee leading from research into development.

2.5.1. Chosen Technologies

I will be using Python over R for this project as it is a language I am familiar with and like and it is highly extensible with a wide variety of libraries supporting every aspect of what is need in this project. I will be using the PyCharm Professional IDE with integrated Git support connecting to a GitHub repository as this IDE provides such a wide variety of features and Git is just so easy to use and widely recognised.

Django will be the framework of choice as it just provides more structure out of the box then Flask does. Amazon Web Services will be used for hosting the production server as the first-year free tier completely nullifies the costs and offers a great range of services, although if I was to extend this project beyond final year I would consider starting instead with or switching too Apache HTTP Server. For data storage I will be using PostgreSQL as it combines the best of both a relational and NoSQL databases.

2.5.2. Chosen DataSets

For the data that my machine-learning models will use, I have selected cresci-2017 dataset.[35] This dataset has been used in academic studies in the field of Twitter bot detection,[7] is part of the datasets used by the Botometer application and covers an excellent range of different accounts.

It is split further into several smaller datasets. First there is a dataset of genuine account, then there are three groups of traditional spambots, the first group are general spambots without any focus, second group are spambots attempting to promote a web URL to try and get users to click it and lastly a group of spambots attempting to push job offers on users as well as getting them to click a specific URL.

Next is a group of fake follower bots which exist purely to make a user appear more popular or influential on the platform. Lastly are three groups of social spambots, the first group are spambots that retweeted a specific political

candidate in Italy, second one group spambots attempting users to download a specific mobile application and lastly a group of spambots trying to sell products on Amazon.com.

2.5.3. Challenges

2.5.3.1. Time

Time management for this entire project will be a major challenge as I will also be juggling my modules, their workloads and exams. While I have planned everything out using a Gantt chart, unexpected situations can arise within college and in my outside life which can throw work timelines off.

2.5.3.2. Implementing Oauth2 authentication

Most applications that access the Twitter API currently use Oauth1 authentication where a user must login to their own account on Twitter, via a redirect window, and authorise the application to use that account to hit the Twitter API. Oauth2 authentication is a newer version where the user does not need to login and therefore not have a Twitter account and the application itself does all the authorising needed to hit the Twitter API.

2.5.3.3. Creating an accurate Model

Picking the correct classifier and in turn creating and tuning an accurate model will be quite difficult and take up a significant period as this is an area that I am only becoming familiar since the start of the college year. It is the most fundamental aspect that the success of the project relies on as without an accurate model the web application holds little value apart from experience gained.

2.5.3.4. Integration

Once each part of the project, the web app and the machine-learning models, are complete then the models need to be integrated into the web app resulting in the finished product. This could potentially be quite tricky and time consuming as it will involve connecting two separate code bases. Allowing for this though ensures each part can be worked on independently which overall should work out for the better.

3. Approach and Methodology

This project has been performed iteratively, using the Agile and Kanban methodology for the overall scope of the project, including the web application, while for the data mining aspect of the project the CRISP-DM methodology was chosen from a few data mining project management models.

Both the Agile and Kanban methodologies and their use in this project as well as the CRISP-DM methodology and a similar methodology: SEMMA, will be explained in this section. A comparison will be made between them and reasons given why I chose the former over the latter.

3.1. Agile & Kanban

3.1.1. Agile

The Agile concept is an approach to project management within the domain of software development and includes various principles such as Feature Driven Design, Scrums, XP and Kanban, the last of which I have incorporated into my project and discuss later.

Agile and its sub-principles revolve around being adaptable, collaborative and versatile and with the focus on iterative and incremental development.[36] Projects that adopt Agile need an approach that facilitates rapid and flexible responses to change as well as continuous improvement.

The 12 Agile principles outlined in the Agile manifesto are [37] :

Principles
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Business people and developers must work together daily throughout the project.
3. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
4. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
5. The best architectures, requirements, and designs emerge from self-organizing teams.
6. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
7. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
8. Simplicity—the art of maximizing the amount of work not done—is essential
9. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
10. Working software is the primary measure of progress.
11. Continuous attention to technical excellence and good design enhances agility
12. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Figure 8 12 Agile Principles

Some of the key differences between Agile and traditional methodologies are that the management style of a project is to lead and collaborate in the former and command and control in the later, that communication throughout

the project is informal vs formal, that the developmental model is an evolutionary-delivery model vs a life-cycle model and that implementation is the focus vs spending large quantities of time on design.[36]

3.1.2. Kanban

Kanban is a part of the Agile family with a heavy focus on continuous delivery while at the same time ensuring that the development team do not become overburdened.[38]

The Kanban methodology was named in 2007 after several presentations given by David Anderson of his management approach at various companies. The word "*kanban*" is the Japanese for sign or signal card and the methodology's roots date back to the middle of the 20th century in Japan where Taiichi Ohno, while working for Toyota, employed the first Kanban system to regulate the workflow in the company.[39]

Kanban has three main principles behind it [38] :

- Visualise the flow of work: Set up an environment, either through post its on a board or digitally to visualise all the work items to give them context.
- Limit the work in progress: Limit each team member to at most 3 pieces of work at any given time to ensure the team does not start and commit to too much work.
- Enhance the flow: Once a team member has finished a piece of work, they take the highest priority work piece in the backlog.

Some of the benefits of the Kanban methodology are shorter cycle times ensuring that new features can be delivered quicker, easy adaption to frequent changes in work piece priority and requires less oversight ensuring team and project leads have more time to focus on other activities.

3.1.3. Project Use

While Agile and Kanban methodologies revolve around a team of multiple members working together it isn't hard to adapt them to my case of a team of one for this project. I take on the roles of not only all members of the team and the team lead but also that of the shareholders that commission the project.

- Shareholders: All the requirements of this project were laid out by me at its outset as well as the one who has been and will continue to assign priorities to all the deliverables and smaller pieces of work.
- Team lead & members: Every piece of work on this project has been and will be worked on and completed by me. I am in charge of ensuring I do get the work done in a timely manner and to a sufficient quality.

I will be using Trello, a free to use web-based project management application, to keep track of and manage my work load through the Kanban method. This was a piece of software I was introduced to during my work placement and is extremely simple and easy to use.[40]

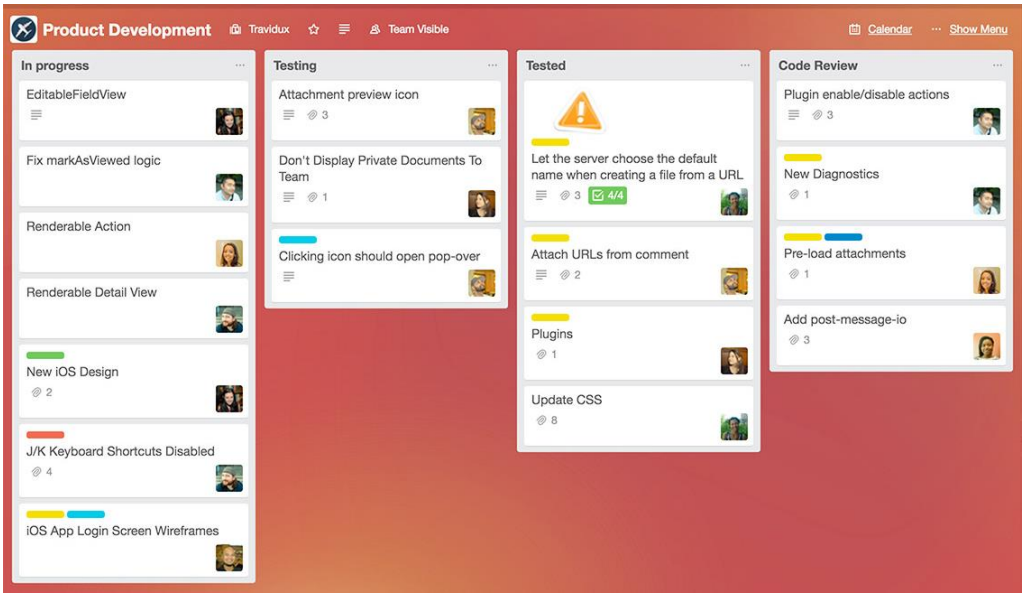

Figure 9 Sample Trello Board [40]

3.2. Data Mining Project Management Models

3.2.1. CRISP-DM

CRISP-DM model is an acronym that stands for Cross-Industry Standard Process for Data Modelling and is a cycle which consists of 6 stages. The sequence of these stages is not strict which allows for movement between any stage if so required.[41]

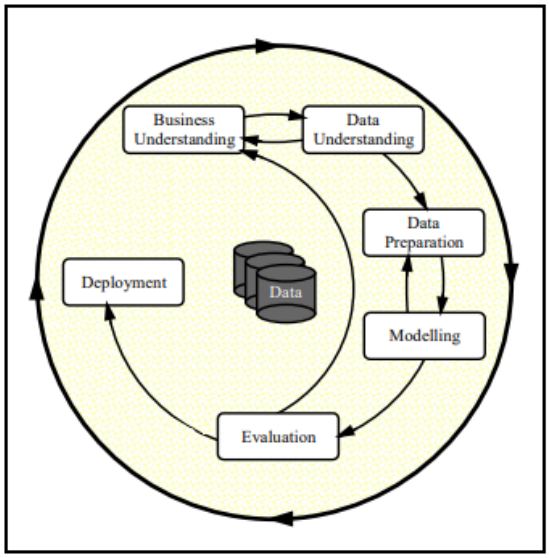

Figure 10 CRISP-DM Model [41]

3.2.1.1. Business Understanding

This stage revolves around understanding the business side of the data mining project:

- What is the domain of the business?
- Assessing the situation in terms of hardware, software, data sources and knowledge bases.
- Transforming business goals into data mining objectives.
- Producing a project plan.

3.2.1.2. Data Understanding

This stage is where data is acquired, and time spent exploring it, trying to understand any correlations between features as well as finding any data quality issues present in the data through a data quality report and data visualisations.

3.2.1.3. Data Preparation

This is the most important stage of the model and requires an excellent understanding of the previous two stages. This is where numerous tasks are performed on the data such as the pre-processing of data, cleaning and reformatting data to remove missing or fix corrupt values and feature selection. Aggregation or merging of data may also occur here.[41]

If the data is not prepared properly then the accuracy of any models it is passed to will suffer greatly and can take multiple iterations to get right.

3.2.1.4. Modelling

This stage revolves around the selection of modelling technique or machine-learning algorithms, decide how to measure the model's validity or accuracy, the building of the model and its assessment to fix any mistakes that can occur in the building process.[41]

These mistakes arise due to *noise*: outliers or missing values in the data that have managed to slip through the previous stage. This then leads to *overfitting*, where the model is too complex for the data and therefore is highly sensitive to noise, or *underfitting*, where the model is too simplistic and unable to detect patterns within the data. Either of these have a negative impact on the accuracy.[32]

3.2.1.5. Evaluation

Evaluation of results from the models, once data has been passed through, is done here. This allows us to get an understanding of the suitability of the models as well as any errors we may have made at an earlier stage. The entire process is reviewed and the next iteration, if needed, is planned.

3.2.1.6. Deployment

Once a model has been created that hits a satisfactory accuracy level it can be deployed. This can have a variety of different meanings, dependent on the project itself, from being used in a work report or a scientific paper or in a data mining application. The deployment must be planned with any monitoring and maintenance considered.

3.2.2. SEMMA

SEMMA is another model, developed by the SAS Institute, which is used to manage a data mining application and is an acronym for the 5 stages that comprise the model [42] :

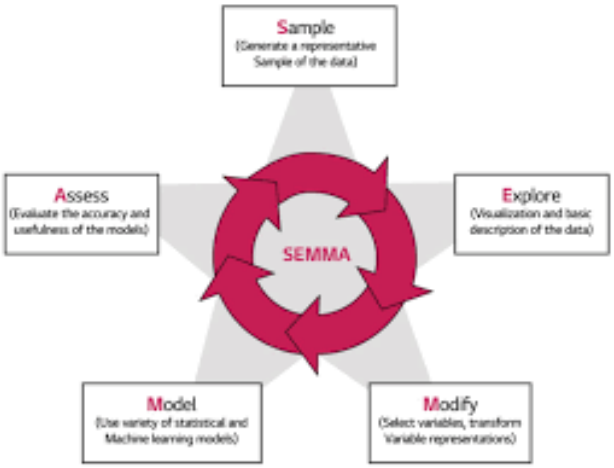

Figure 11 SEMMA Model [43]

5.2.2.1. Sample

This stage focuses on taking a sample of the dataset for use in the model. The data must be of sufficient size such that accurate patterns can be drawn from it while small enough that it can still be used efficiently. Data partition of the sampled dataset into training and testing sets is also done here.

5.2.2.2. Explore

Exploration of the data is done here to detect any unexpected patterns, anomalies such as missing or corrupted data, or instances in the data that prove to be unnecessary while also gaining a better understanding of the data. This is done through visual representations and statistical techniques.[42]

5.2.2.3. Modify

The data is modified by creating, deleting, selecting and/or transforming variables within it to ensure the data being passed into the model is of high quality and considers any issues that arose in the exploration stage. Not every issue will be perfectly solvable, and it will be a case of applying the best fit solution.[42]

5.2.2.4. Model

The creation of the model is dealt with here, where selected modelling techniques are employed to build a model that will accurately make predictions based upon the data that is passed to it.

Again, like in CRISP-DM, the problems of *overfitting* and *underfitting* due to *noise* must potentially be dealt with.

5.2.2.5. Assess

The outputs of the model, after inputting the training set, that was set aside during the sampling stage, are compared to the actual outcomes of that dataset with the model's accuracy and usefulness being evaluated.

3.2.3. Differences in Models

Looking at both models it is easy to see that they are quite similar, in fact SEMMA appears to be akin to a slimmed down version of CRISP-DM with the first, Business Understanding, and last stage, Deployment, omitted such that all the focus is on data modelling aspect of the application, though realistically some knowledge of the business domain must be known or else the final product will be lacking in various areas.

This table clearly demonstrates this comparison:

CRISP-DM	SEMMA
Business Understanding	----------
Data Understanding (Part 1)	Sample
Data Understanding (Part 2)	Explore
Data Preparation	Modify
Modelling	Model
Evaluation	Assess
Deployment	----------

3.2.4. Conclusion

The CRISP-DM model was chosen for use within this project as it is an industry standard model that is used widely and as such is well proven, is robust and versatile allowing for movement between any stage if needed and finally inclusion of the deployment stage, which SEMMA is missing, as the data mining aspect of the project will need to be integrated and deployed within the overall web application.

4. Design

This section details the technical architecture chosen for this project, a diagram of it and all other design documents including Use Case and Class diagrams and an Entity Relationship Diagram.

4.1. Technical Architectures

4.1.1. Model View Controller

The Model View Controller (MVC) architecture is used across a wide range of applications where there is a need to provide a User Interface through a desktop or web front-end. It is a three-tier architecture which uses the Controller, comprised of several classes such as a Command Factory class and Command, Service, and DAO classes, to pass information between the View, i.e. the front end, and the Model, i.e. the backend.[44]

This ensures the separation of roles between the different sections of code in a project. This makes it easier to divide up the work in a project as team members can focus on specific sections without worrying too much about the other parts enabling better development and testing.

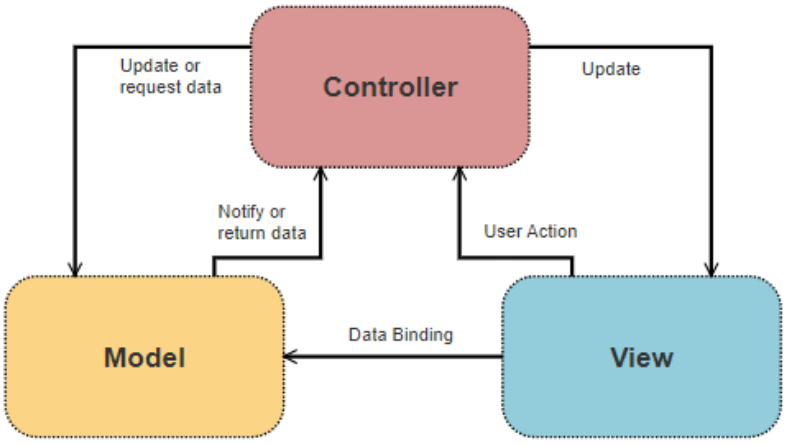

Figure 12 Model View Controller

4.1.2. Model View Template

For this project I have chosen to use the Django framework which uses its own modified version of the MVC called the Model View Template (MVT). In this adaption Django takes care of the Controller role and replaces it with the Template section, which takes the role of the presentation layer by containing all

the HTML, CSS and Forms files while the View section deals with all business logic and handles all requests from and responses to the User. The Model section stays the same and deals with everything to do with the database.[45]

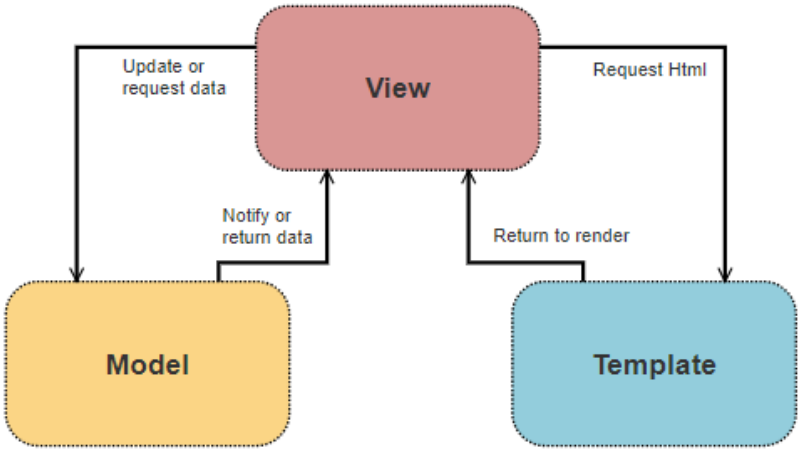

Figure 13 Model View Template

4.2. Technical Architecture Diagram

As this is a web-based data mining application I have adapted the MVT architecture to suit this project by adding another layer between View and Model, called Mining, to consider the data mining and machine learning aspects of this project. This layer deals with everything from data pre-processing to the creation and evaluation of the machine-learning models to analysis of new data passed to it.

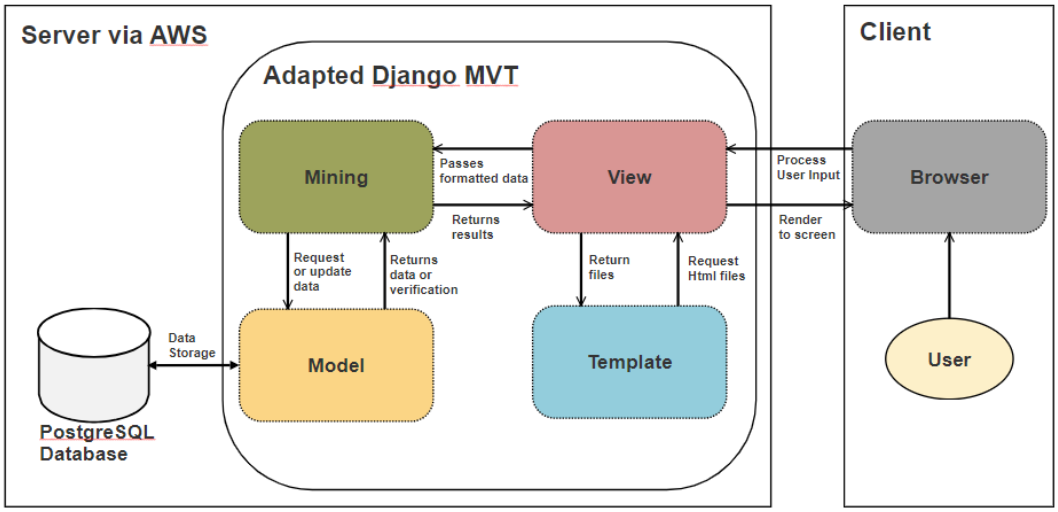

Figure 14 Application Technical Architecture

4.3. Other Design Documents

4.3.1. Use Case Diagram

The use case below details how a user will interact with the system. The user can enter in their own Twitter username or any other one they wish, be it a celebrity's, one of their friends or any other account they know off. They will then be able to view the results about how likely that account is a bot and be able to either share those results on social media or download them into a file.

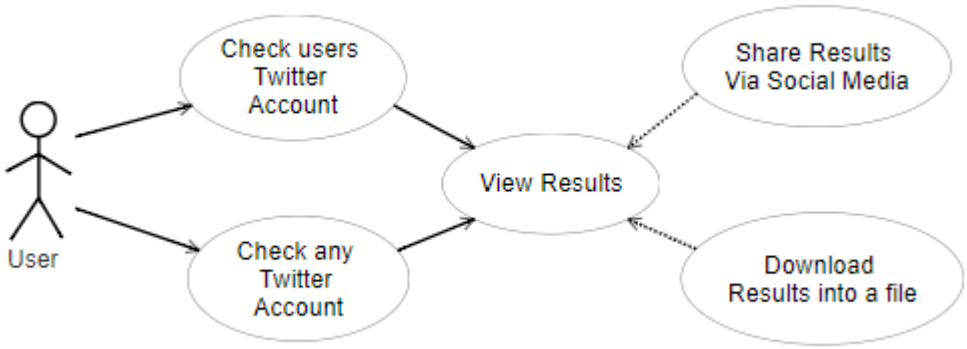

Figure 15 User Use Case Diagram

4.3.2. Entity Relationship Diagram

This diagram is subject to change due to trying to achieve a more accurate result in further iterations of the development cycle.

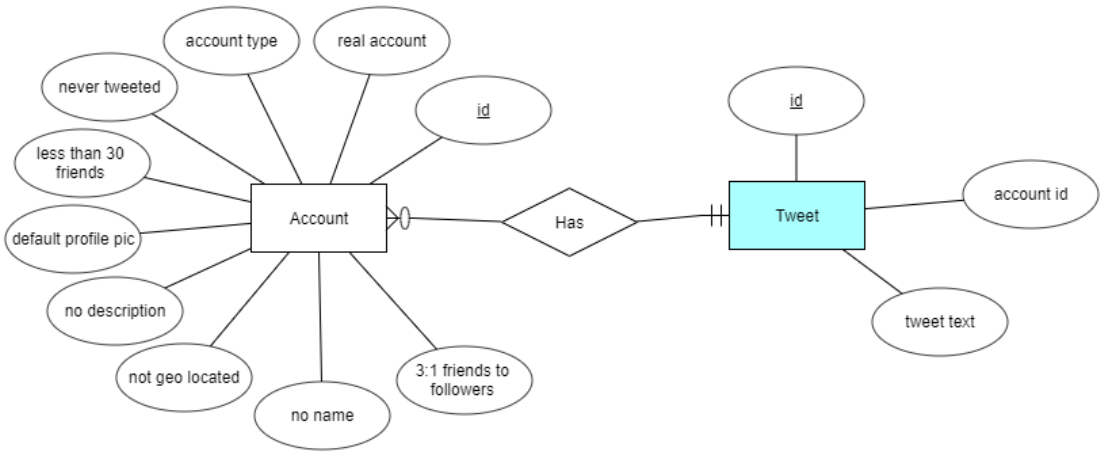

Figure 16 Entity Relationship Diagram

5. Prototyping and Development

This section explains what prototyping and development has been completed to date, giving details on building the web front-end and the creation of a basic machine-learning model.

5.1. Vertical Prototype

The prototyping for this project revolves around creating a vertical prototype, which shows the basic structure and functionality of both sub-sections of the project. This will then be reviewed over the December break and if found satisfactory, built upon heavily to create the final application.

5.1.1. Web Front-End

For this section of the vertical prototype, the goal was to have a working web-app that could connect to the Twitter API and retrieve data from it, in this case the twenty most recent tweets from the account linked to the username chosen by the web-app user.

Below are two screenshots of the working web page which asks for the user to input a Twitter username then retrieves and outputs the data received back from the Twitter API to screen.

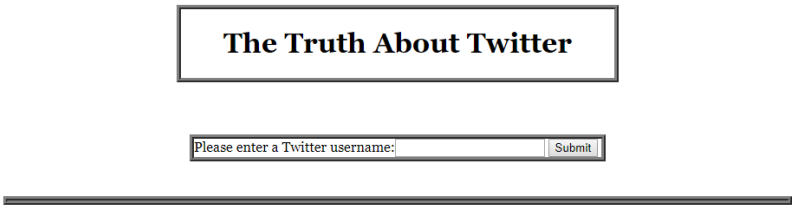

Figure 17 Web Front End

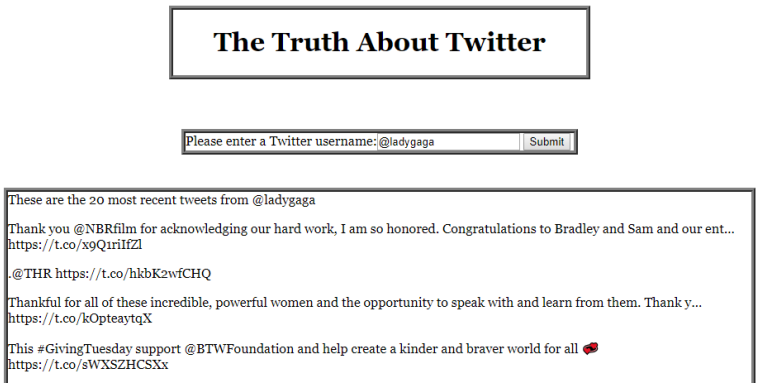

Figure 18 Web Front End with returned tweets

5.1.2. Basic Machine-Learning Model

For this section of the vertical prototype, the goal was to have a basic machine-learning model that uses data from the acquired datasets and using k-fold cross validation, uses the entirety of the selected data as training and testing sets, outputting, to console, the average accuracy result across the partitions, repeated five times.

Seven features were chosen for this initial model and will be re-evaluated and changed further into development: Whether the account has the default profile picture, has a screen name, has a description, has less than 30 friends, has more than 1000 friends, has never tweeted, the account is geo located and the ratio of friends to followers is 3:1.

A Naïve Bayes classifier with Bernoulli distribution was chosen for this as it a good classifier to start with any data mining project and the inputs are of a binary format, 0 and 1's with 2000 accounts chosen, 1000 random genuine accounts and 1000 random traditional bot accounts. Below is the output from this completed section the accuracy sitting at around 60% depending on which accounts are selected at the start.

```
Data comprises of 2000 shuffled accounts:
    1000 random Genuine accounts
    1000 random Traditional accounts
K-fold Cross validation used with k = 10
Classifier: Naive Bayes with Bernoulli distribution
Mean of the partition scores for each run:

    Run 0: 0.6080000000000001
    Run 1: 0.607
    Run 2: 0.6085
    Run 3: 0.6085
    Run 4: 0.608
```

Figure 19 Results of Model being run

5.2. Development

This section deals with all the development done to date in the creation of the vertical prototype using the Django framework and various python libraries such as Tweepy and Scikit-learn.

5.2.1. Web Front-End

There were several steps to creating the web front-end shown above in the previous section and each will be explained with code snippets where needed.

A new Django project was created within PyCharm Professional, allowing a lot of the tedious groundwork for a web application to be taken care. This meant a bare-bones skeleton app was ready for use and to be built upon.

Next a Twitter developer account was created using my own Twitter account and a Twitter app created, noting its Consumer Key and Secret Key. An Access Token and Access Token Secret were created and noted next. These were outputted to a json file: twitter_credentials.json, using twitter_credentials.py, for use later.

```
import json

# Enter your keys/secrets as strings in the following fields
credentials = {}
credentials['CONSUMER_KEY'] = 'ivlilqYNCuNhNLgm2EX6Zrkj8'
credentials['CONSUMER_SECRET'] = 'X9MWGxSVgJu6ExO5dgEMikJoHDrPmrC6uDNmOVorZmVJFHsRU7'
credentials['ACCESS_TOKEN'] = '3330453371-53qtrIFDcyqzzO0f3wg4yV5jo7ACtzESMhSHOPv'
credentials['ACCESS_SECRET'] = 'THGzrqUaPgrlpNPubudNBGj92slibYfOlhM2CwmNJgZ6U'

# Save the credentials object to file
with open("twitter_credentials.json", "w") as file:
    json.dump(credentials, file)
```

Figure 20 Saving Twitter credentials to file

Next are all changes made to the Django skeleton code:

- Creating a basic form called UsernameForm, in forms.py, to take in user input.

```
class UsernameForm(forms.Form):
    username = forms.CharField(label='Please enter a Twitter username', max_length=16)
```

Figure 21 Basic form

- This form was added to views.py within the index method. If the request method was POST, the existing form instance was read in and the user input read into username for use later otherwise a new form instance was created.

```
if request.method == 'POST':
    form = UsernameForm(request.POST)
    if form.is_valid():
        username = form.cleaned_data['username']

else:
    form = UsernameForm()
```

Figure 22 Creating or reading in form

- The values from twitter_credentials.json are read in and are used in combination with the username inputted in the form to access the Twitter API to return the most recent twenty tweets from that username.[22]

```
file_name = os.path.join(settings.BASE_DIR, 'twitterstruth\\credentials\\twitter_credentials.json')
with open(file_name, "r") as file:
    creds = json.load(file)

auth = tweepy.OAuthHandler(creds['CONSUMER_KEY'], creds['CONSUMER_SECRET'])
auth.set_access_token(creds['ACCESS_TOKEN'], creds['ACCESS_SECRET'])

api = tweepy.API(auth)
tweets = api.user_timeline(username)
tweets_header = 'These are the 20 most recent tweets from ' + username
```

Figure 23 Twitter Authentication and tweet retrieval

- Then a response is returned, asking to render index.html with passed variables: username, tweets and tweets_header for use in the Html file.

```
return render(request, 'index.html', {'form': form, 'tweets': tweets, 'tweets_header': tweets_header})
```

Figure 24 Render Index.html with variables passed

- The templates section of settings.py was altered so to know where to look for template files suck as index.html

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'twitterstruth\\templates\\twitterstruth')]
        ,
        'APP_DIRS': True,
```

Figure 25 Template directory added

- Index.html was been altered to show the form and tweets using the variables passed to it via views.py

```
<div id="form">
    <form method="post">
        {% csrf_token %}
        {{ form }}
        <input type="submit" value="Submit">
    </form>
</div>
<div id="tweets">
    {{tweets_header}}
    {% for tweet in tweets %}
        <p>{{tweet.text}}</p>
    {% endfor %}
</div>
```

Figure 26 Index.html

- Index.py uses a CSS file web_style.css, which is located within the twitterstruth/staticfiles/css sub-directory. [46]

```
{% load static %}
<link href="{% static 'css/web_style.css' %}" rel="stylesheet" type="text/css">
```

Figure 27 Load web_style.css

- To enable this CSS file, and others within the twitterstruth/staticfiles sub-directory, to be found urls.py was altered [46] :

```
urlpatterns = [
    path('index', views.index, name='index'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

urlpatterns += staticfiles_urlpatterns()
```

Figure 28 Enabling CSS file load part 1

- As was settings.py [46]:

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

MEDIA_URL = '/media/'

STATIC_ROOT = os.path.join(BASE_DIR, 'static')

STATIC_URL = '/static/'

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'twitterstruth\\staticfiles'),
)
```

Figure 29 Enabling CSS file load part 2

5.2.2. Basic Machine-Learning Model

There were several steps to creating the basic machine-learning model shown above in the previous section and each will be explained with code snippets where needed.

- As the files for this section are being run separately but still need access to certain files within the Django framework, the environmental variable, DJANGO_SETTINGS_MODULE, must be set and Django setup within each file:

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'The_Truth_about_Twitter.settings')
django.setup()


from twitterstruth.models import Account
from django.conf import settings
```

Figure 30 Enabling Django file use

- The file read_store.py deals with all reading in and storing of the datasets.
- Each dataset is read in from their CSV files one at a time using the pandas library [18], with the tweets files being ignored until the next phase of development.

```
def read_in_all_data():
    Account.objects.all().delete()
    read_in_csv('genuine_accounts', 1)
    read_in_csv('fake_followers', 2)
    read_in_csv('social_spambots_1', 3)
    read_in_csv('social_spambots_2', 3)
    read_in_csv('social_spambots_3', 3)
    read_in_csv('traditional_spambots_1', 4)
    read_in_csv('traditional_spambots_2', 4)
    read_in_csv('traditional_spambots_3', 4)
    read_in_csv('traditional_spambots_4', 4)


read_in_all_data()
```

Figure 31 Read in all data

```
def read_in_csv(directory, account_type):
    BASE_DIR = getattr(settings, "BASE_DIR")
    users_file = os.path.join(BASE_DIR, 'data/' + directory + '/users.csv')
    # tweets_file = os.path.join(BASE_DIR, 'data/' + directory + 'tweets.csv')

    users = pd.read_csv(users_file)
    users = users.fillna('')

    # tweets = pd.read_csv(tweets_file)
    # tweets = tweets.fillna('')
```

Figure 32 Read in single dataset from CSV

- Checks are done on certain columns in the dataset and binary outputs given depending on the result to form the data going into the database:

```
if getattr(row, 'statuses_count') == 0:
    never_tweeted = 1
else:
    never_tweeted = 0
```

Figure 33 Data check example

- The data is then read into the database:

```
Account.objects.create(id=account_id, real_account=real_account, account_type=account_type,
                       default_profile_pic=default_prof_pic, no_name=no_name, no_desc=no_desc,
                       lt_30_friends=lt_30_friends, gt_1000_friends=gt_1000_friends,
                       not_geo_located=no_geo_location, never_tweeted=never_tweeted,
                       three_friends_one_followers=three_friends_one_follower)
```

Figure 34 Adding data to database

- In machine_learning.py, data is read out from the database and passed to through a model, giving output to the console of the model's accuracy with each run.
- All the randomly chosen, accounts are read out from the database, being split into features and corresponding targets lists:

```
features, targets = get_accounts(features, targets, 1, 1000)
features, targets = get_accounts(features, targets, 4, 1000)
```

Figure 35 Get 2000 accounts

```
accounts = Account.objects.filter(account_type=account_type)
random_accounts = random.sample(list(accounts), sample_size)

for acc in random_accounts:
    dum_features.append([acc.default_profile_pic, acc.gt_1000_friends,
                         acc.lt_30_friends, acc.never_tweeted,
                         acc.no_desc, acc.no_name, acc.not_geo_located,
                         acc.three_friends_one_followers])

    if acc.real_account:
        dum_targets.append(0)
    else:
        dum_targets.append(1)

return dum_features, dum_targets
```

Figure 36 Return random sub-set of accounts from database

- The lists were converted into NumPy arrays [19] and the sklearn library used for k-fold cross validation and classifiers initialisation [17]:

```
features = np.asarray(features)
targets = np.asarray(targets)

kf = KFold(n_splits=10, shuffle=True)
clf = BernoulliNB()
```

Figure 37 Convert arrays and initialise model

- This model was run five times, with the mean accuracy score across the partitions outputted each time:

```
for index in range(5):
    scores = cross_val_score(estimator=clf, X=features, y=targets, cv=kf)
    print('    Run ' + str(index) + ': ' + str(np.mean(scores)))
```

Figure 38 Run model and output results to screen

6. Testing

This section will explain how testing for the various parts of this project will be done. The testing is split up into 3 parts: The data mining and machine learning section, the web front-end and lastly the fully integrated combination of these two parts.

6.1. Data Mining & Machine Learning section

This part will be employing the K-fold cross validation procedure, talked about during the research stage of this document, to perform my testing on all the models that I build or tweak in this project.

This method involves partitioning up a dataset into K partitions of equal size and for each one, taking that as the testing set with the remaining partitions as the training set.

The dataset I am using is already divided up into multiple sub sets. One of these represents a collection of real Twitter accounts while the rest represent different types of bot accounts. In turn, each of the bot datasets will be mixed separately with the dataset of real accounts and K-fold cross validation will be applied ensuring that the models are trained and tested using the entire mixed dataset each time.

6.2. Web Front-End

This part will be relatively simple and as such, until the previous part is integrated with this one, taking on the role of tester and trying to break every part of it should suffice.

6.3. Web-based Data Mining Application

Once everything is integrated together I will again take on the role of tester while also asking for several testers from my friends and family to help find any bugs that might arise from the integration stage or missed at an earlier stage.

7. Issues and Risks

The main issues still to resolve in this project are as follows:

- Rebuilding the basic machine learning model to improve its accuracy by the selection of different classifier and techniques and by going back to the business side and seeing do I need to add, remove or change the features being passed to the model.
- Enabling the use of Oauth2 authentication such that the application logins to Twitter instead of the user. This will be solved by learning more about Oauth2 authentication, as it is a newer technology, and working on its implementation.
- Integration of the data mining aspect into the web aspect to create the full application. This should just be a case of careful hooking up the pieces of code that need to be able to communicate with one another from the separate parts but may prove trickier.

8. Plan and Future Work

After the submission of this report and subsequent presentation and demo of my vertical prototype the three key deliverables left will be finishing my dissertation and the full web application and preparing for the final demo by the end of April.

As shown in the Gantt chart below, most of the development time will go to continued building and accuracy tweaking of the basic models from the vertical prototype and then integrating these, once finished, into the web application. The rest of the time will go into more background reading and morphing this report into the first part of my dissertation and then writing the rest of it. Time has also been set aside to focus on my Winter exams when they are due.

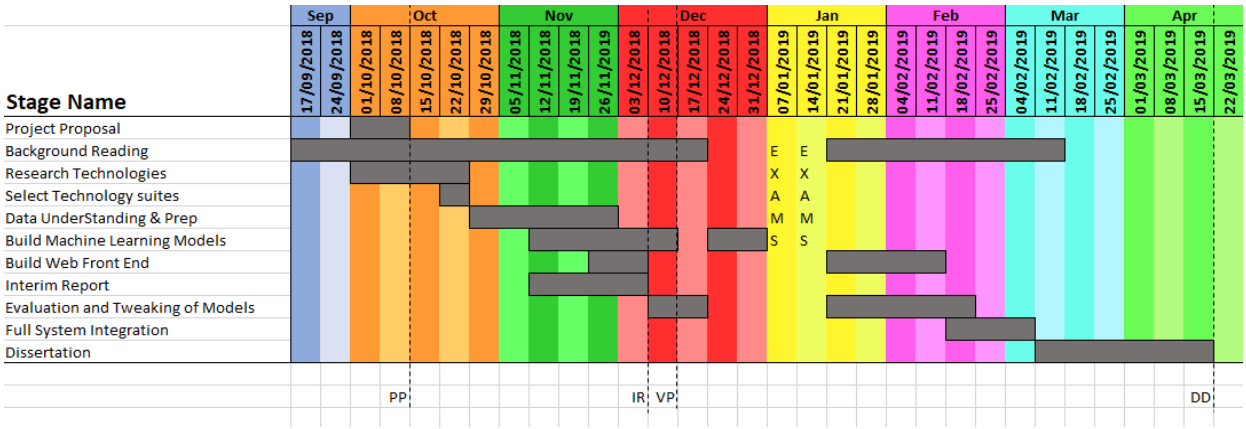

Figure 39 Project Gantt chart

9. Bibliography

[1] Statista; (October 2018), Twitter: number of active users 2010-2018, www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/, Date Accessed: November 2018

[2] Varol, Onur; Ferrara, Emilio; Davis, Clayton A.; Menczer, Filippo; Flammini, Alessandro; (2018) "*Online Human-Bot Interactions: Detection, Estimation, and Characterization*", ICWSM'17, Page 1-9.

[3] Perlroth, Nicole; (April 2013), Fake Twitter Followers Become Multimillion-Dollar Business, bits.blogs.nytimes.com/2013/04/05/fake-twitter-followers-becomes-multimillion-dollar-business, Date Accessed: October 2018

[4] Perlroth, Nicole; (April 2013), Researchers Call Out Twitter Celebrities With Suspicious Followings, bits.blogs.nytimes.com/2013/04/25/researchers-call-out-twitter-celebrities-with-suspicious-followings, Date Accessed: October 2018

[5] Ferrara, Emilio; (November 2016), How Twitter bots affected the US presidential campaign, theconversation.com/how-twitter-bots-affected-the-us-presidential-campaign-68406, Date Accessed: October 2018

[6] Nimmo, Ben; (August 2017), #BotSpot: Twelve Ways to Spot a Bot, medium.com/dfrlab/botspot-twelve-ways-to-spot-a-bot-aedc7d9c110c, Date Accessed: October 2018

[7] Efthimion, Phillip George; Payne, Scott; Proferes, Nicholas; (2018) "*Supervised Machine Learning Bot Detection Techniques to Identify Social Twitter Bots*", SMU Data Science Review, 1(2) , Article 5.

[8] Rushe, Dominic; (October 2017), Twitter bans ads from RT and Sputnik over election interference, www.theguardian.com/technology/2017/oct/26/twitter-bans-ads-from-russia-today-and-sputnik-over-election-interference, Date Accessed: October 2018

[9] BBC news; (July 2018), Twitter 'shuts down millions of fake accounts', www.bbc.com/news/technology-44682354, Date Accessed: October 2018

[10] Leskin, Paige; (October 2018), Twitter shuts down bots pushing pro-Saudi reports on missing columnist, uk.businessinsider.com/twitter-shuts-down-pro-saudi-bots-missing-columnist-2018-10?r=US&IR=T, Date Accessed: November 2018

[11] Netimperative; (November 2018), US elections: Twitter shuts down 10,000 bot accounts 'discouraging voting', www.netimperative.com/2018/11/us-elections-

twitter-shuts-down-10000-bot-accounts-discouraging-voting/, Date Accessed: November 2018

[12] Sattler, Jason; (September 2017), blog.f-secure.com/4-reasons-so-hard-for-twitter-to-shut-down-bots/, Date Accessed: October 2018

[13] OSoMe; (-), Botometer by OSoMe, botometer.iuni.iu.edu/#!/faq, Date Accessed: October 2018

[14] R; (-), R: The R Project for Statistical Computing, www.r-project.org, Date Accessed: October 2018

[15] Python; (-), Welcome to Python.org, www.python.org, Date Accessed: October 2018

[16] JetBrains; (-), PyCharm: the Python IDE for Professional Developers by JetBrains, www.jetbrains.com/pycharm, Date Accessed: October 2018

[17] Scikit-learn; (-), scikit-learn: machine learning in Python, scikit-learn.org/stable/, Date Accessed: October 2018

[18] Pandas; (-), Python Data Analysis Library, pandas.pydata.org, Date Accessed: October 2018

[19] NumPy; (-), NumPy -NumPy, www.numpy.org, Date Accessed: October 2018

[20] Ronacher, Armin; (-), Welcome | Flask (A Python Microframework), flask.pocoo.org, Date Accessed: October 2018

[21] The Django Software Foundation; (-), The Web framework for perfectionists with deadlines, www.djangoproject.com, Date Accessed: October 2018

[22] Tweepy; (-), Tweepy Documentation, tweepy.readthedocs.io/en/v3.5.0/index.html, Date Accessed: October 2018

[23] The Apache Software Foundation; (-), Welcome! – The Apache HTTP Server Project, httpd.apache.org, Date Accessed: November 2018

[24] Heroku; (-), Cloud Application Platform, www.heroku.com, Date Accessed: November 2018

[25] Amazon; (-), Amazon Web Services (AWS) – Cloud Computing Services, aws.amazon.com, Date Accessed: November 2018

[26] Git; (-), Git, git-scm.com, Date Accessed: November 2018

[27] Mercurial; (-), Mercurial SCM, www.mercurial-scm.org, Date Accessed: November 2018

[28] Oracle; (-), MySQL, www.mysql.com, Date Accessed: October 2018

[29] PostgreSQL; (-), PostgreSQL: The world's most advanced open source database, www.postgresql.org, Date Accessed: October 2018

[30] MongoDB; (-), Open Source Document Database, www.mongodb.com, Date Accessed: October 2018

[31] Mauro, Andrea De; Greco, Marco; Grimaldi, Michele; "*What is big data? A consensual definition and a review of key research topics*", AIP conference proceedings, (2015), 1644(1), pp. 97–104.

[32] Bramer, Max; (2016) "*Principles of Data Mining*", Salmon Tower Building New York City: Springer.

[33] Kelleher, John; Mac Namee, Brian; D'Arcy, Aoife; (2015) "*Fundamentals of machine learning for predictive data analysis*", Cambridge Massachusetts: MIT Press.

[34] Agatonovic-Kustrin, S; Beresford, R; "*Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research*", J Pharm Biomed Anal. (2000), 22(5), pp.717-27.

[35] Botometer; (-), Bot Repository, botometer.iuni.iu.edu/bot-repository/datasets.html, Date Accessed: October 2018

[36] Martin, Robert C.; (2013) "*Agile Software Development, Principles, Patterns, and Practices*", London, Pearson.

[37] Manifesto Authors, (2001), Manifesto for Agile Software Development, agilemanifesto.org, Date Accessed: October 2018

[38] Anderson, David J.; (2010), "*Kanban: Successful Evolutionary Change for Your Technology Business*", Seattle, Blue Hole Press

[39] Toyota, (2004), Toyota Traditions, www.toyota-global.com/company/toyota_traditions/quality/mar_apr_2004.html, Date Accessed: October 2018

[40] Atlassian; (-), Trello, trello.com/en, Date Accessed: October 2018

[41] Wirth, Rüdiger; Hipp, Jochen; (2000) "*CRISP-DM: Towards a Standard Process Model for Data Mining*".

[42] SAS, (-), Data Mining Using SAS® Enterprise Miner™: A Case Study Approach, Third Edition, support.sas.com/documentation/cdl/en/emcs/66392/HTML/default/viewer.htm, Date Accessed: October 2018

[43] Azevedo, A. I. R. L.; Santos, M. F.; "*KDD, SEMMA and CRISP-DM: a parallel overview*", IADS-DM, (2008).

[44] Gamma, Erich; Vlissides, John; Johnson, Ralph; Helm, Richard; (1994), "*Design Patterns: Elements of Reusable Object-Oriented Software*", Boston Massachusetts, Addison-Wesley Professional

[45] The Django Book; (2018), The Model-View-Controller Design Pattern, djangobook.com/model-view-controller-design-pattern/, Date Accessed: October 2018

[46] Catherine; (2013), Include CSS and Javascript in my django template, stackoverflow.com/questions/15491727/include-css-and-javascript-in-my-django-template, Date Accessed: November 2018