Twitter bot Detection & Categorization

A comparative study of machine learning methods

Jan Novotny

Supervised by prof. Krzysztof Podgórski

Master's Thesis in Statistics
15 ECTS

June, 2019

Abstract

Automated Twitter accounts, or Twitter bots, have gained increased attention lately. In particular, a novel type of bot, so called *social bots*, are piquing peoples' interest, as these bots have recently been involved in a number of political events. Most of the previous work on how to detect bots has not distinguished between such novel bot types, and other, less sophisticated ones. Instead, they have all been lumped together, producing models to detect automated behaviour in general. Although indeed useful, the above approach might cause issues when the task at hand concerns one particular type of bot account. This thesis therefore attempts at making the classification of bots more fine grained, viewing social bots, traditional spambots, and fake followers as three separate bot categories, to be classified together with the category for actual human users (called "genuine users"). Four machine learning methods are trained and compared for this purpose, showing that the *random forest* performs slightly better than the rest in all performance measures used. However, all models yield an overall accuracy above 90%, which is relatively high compared to similar studies in the field. The analysis also indicates that data sampling has been biased, skewing the data to yield some unexpected results. For instance, genuine users show much more activity than would be expected of the average human-controlled Twitter account. Additionally, traditional bots, which are supposed to be the easiest to classify, instead appear to be the opposite. If the data sampling has indeed been biased, the validity of the models trained on this skewed data is called into question. Hence, more research into sampling techniques is suggested, and it is concluded that the models produced should be tested on more diverse datasets. Without these kinds of repeated studies, the impact of the supposed sampling bias, and consequently the usefulness of the models in real world situations, cannot be properly assessed.

Keywords: *twitter bots, multinomial classification, machine learning*

Contents

1 Introduction

1.1 Background

In March 2006, a group of people at the podcast company Odeo created the first prototype of what would later become one of the largest social networks in the world: Twitter. What began as an SMS-based app for friends to casually update each other on what they were doing, has today turned into a dominant force in the global media landscape [1]. At the time of writing, the site has more than 300 million monthly users [24], among which are politicians, activists, news outlets, and other public figures and organizations.

The way Twitter functions is relatively simple: you begin by creating an *account*. From this account you can then start *tweeting*, i.e. posting texts (tweets) up to 280 characters long (previously only 140 characters). Other people can choose to *follow* you, and you can choose to follow them. Following someone means that whenever that person tweets something from their account, it shows up in your *feed*, which is a continuously updated web page on which all tweets of those you follow are displayed. If you like something someone has posted, you can notify the user of this by pressing a *like-button* on the post. You can also choose to *retweet* it, which allows you to post the tweet from your own account, but with the original poster's username still attached.

Using these functions, the original idea behind the platform was for friends and communities to keep up to date and share information with each other. However, as Twitter has expanded, so has its multitude of areas of use.

Twitter bots

One of the rising phenomena, that has received much attention lately, are the so called *Twitter bots*. A Twitter bot can be many things, as the term lacks a precise, widely agreed upon, definition. In a general sense, however, a Twitter bot may be said to be any type of account that operates via some sort of automation.

Accounts of this type have been involved in a number of large political events during the last decade, where these accounts have attempted, in different ways, to exert influence over the public. Examples of where such actuating campaigns have occurred include the 2010 U.S. Midterm election [20] and the Massachusetts Special election in the same year [18], the 2016 U.S. Presidential election [21], the 2017 French Presidential election [9], and the ongoing conflict between Ukraine and Russia [13].

In light of this, one might understand why bots, on Twitter and elsewhere, have been discussed in news outlets as well as in academia. With such a large number of users, the impact of political campaigns on social media platforms could potentially be sizable.

Sticking to the very general definition of bots as simply any automated accounts, however, risks creating confusion in these discussions. For instance, with this definition, many media outlets and other large organizations using Twitter to post news and press releases, would be categorized as bots. While these accounts are indeed automated, they are generally of little interest to research concerning the detection of bot operations, for the simple reason that they are already known to be automated. Anyone trying to analyze or find ways of identifying bots on Twitter or any other social medium, therefore first needs to define what is meant by the word ”bot”.

To bring clarity to this issue, R. Gorwa and D. Guilbeault have suggested a typology for different kinds of automated or seemingly automated programs which are frequently referred to as bots [11]. This list includes the following definitions:

- **Crawlers and scrapers:** bots programmed to index and archive websites, e.g. to make the sites accessible via search engines.
- **Chatbots:** programs engaging in interactive dialogue with humans via natural language. Often used by companies for simpler communication with customers, such as answering frequently asked questions.
- **Spambots:** either programs, or computers taken over by malware and controlled by a third party, used to send out massive amounts of messages, advertisements, comments, etc..
- **Social bots:** bots active on social media, automatically generating content, and often even posing as real humans.
- **Sockpuppets:** real humans using fake identities, for instance to promote a certain product or cause. May act in ways similar to that of automated accounts.
- **Cyborgs:** a combination of automated and manually controlled programs, although the exact level of automation required for a program to fall into this category is not determined.

Although this list is not exhaustive, and although it is not always obvious where to draw the line between different types of automated programs (a bot on Twitter spreading spam messages; is that a social bot or a spambot?), the categorization presented above can nonetheless serve as a baseline for further discussions. Unless further specification is provided, the term ”bot” or ”Twitter bot” will here be used to denote social bots posing as humans or in other ways obscuring their true identity on Twitter, as well as spambots and sockpuppets operating on the social network. In practice, some of the above mentioned social bots might not be fully automated, so cyborg accounts on Twitter concealing their true purpose or origin will also be included under the ”bot” term. Accounts managed manually by humans, i.e. non-bots, will be called "genuine users". By using these definitions, news outlets and other accounts which are known to use automation are excluded from the analysis. Bot types completely irrelevant for the study, like crawlers and scrapers, are naturally also left out.

1.2 Research question

This paper takes aim at developing techniques for detecting and categorizing bots on Twitter. Previous works on the subject have almost exclusively focused on the detection part. That is, finding methods to differentiate between *bots and genuine users*. Here, the author attempts to make that differentiation even more fine grained, using multinomial classification to discriminate between different *types of bots*. The reason for this is that if the recent technological development continues, yielding an ever more diversified landscape of automated accounts, for certain types of tasks it might become irrelevant to talk about bots as a unified group. Any estimation of the number of sophisticated social bots or sockpuppets, for instance, risks being potentially gravely erroneous, if all types of bots are lumped together into the same group.

The goal of this paper is to find the best performing machine learning method for Twitter bot detection and categorization. While many works on bot detection have been written, there has, to the best of the author's knowledge, almost not been anything written about bot categorization, i.e. multinomial classification of different bot types. Moreover, the author has not been able to find anything written on comparing different multinomial classifiers with each other.

Seeing as the possible number of methods one could try is very large, only a handful will be examined in this paper. *Random forest*, which has yielded promising results for binary classification [25], [16], [22], is one of the candidates. Another is *logistic regression*, due to the fact that it is a linear model that is relatively easy to interpret. Thirdly, *artificial neural networks* will be tried out, mainly because there does not appear to have been much written on Twitter bot detection using this technique. Finally, the *k-nearest-neighbors* method will be employed, for very much the same reasons as the artificial neural networks.

The main idea behind using these specific methods is that they differ quite a lot, which makes for more interesting and revealing comparisons.

With this in mind, the following research question is stated:

- Which method, out of random forest, logistic regression, artificial neural networks, and k-nearest-neighbors, performs best in the multinomial classification of different types of Twitter accounts (i.e. classifying different bot types along with genuine users)?

Of course, the choice of features, data set, and which models to examine, are all things that will affect the final verdict of this study, and replications are necessary before any grand conclusion can be drawn. It is therefore worthwhile to underline that the purpose of this thesis is to serve as part of a larger body of evidence yet to be collected.

1.3 Related Work

Detection

As mentioned in the previous section, most of the work written on the issue of Twitter account classification is concerned with distinguishing between bots and non-bots, i.e. a binary classification problem. An example of this is the early work by A.H. Wang [26], which compared different classification methods for detecting Twitter spammers, using a dataset annotated by the author himself. It showed that out of the methods tested, the naïve Bayesian classifier performed best.

Another work using some kind of spambots is the work by O. Varol et. al. [25]. Here, the dataset originally sampled by K. Lee et. al. [16] around the year 2010 is used to train a random forest model, which is then evaluated on newly collected, manually annotated Tweet data. The reason the model is evaluated on newly collected data is to determine whether it is still valid for the new kind of bots that have most likely appeared on Twitter since 2010. It is, however, not made clear exactly what kind of bot behaviour the annotators have been able to detect. It could be that they have been able to detect mainly the same kind of spambots that existed in the training set, in which case the model evaluation might actually not confirm what it claims to do.

In the work by J.P. Dickerson et. al. [5], sentiment analysis of tweet contents is employed for feature creation. Manual annotation is yet again used to create labels for the observations, on both training and test sets. However, the accounts used come from a sample collected in connection with the Indian 2014 election, in which only accounts tweeting on topics related to that election have been included. It is therefore more likely that this work actually uses the type of novel, more sophisticated social bots in its models.

This novel type of social bot is also studied in the work by S. Cresci et. al. [3], using Twitter accounts which have been promoting political campaigns or consumer products, and which have later been verified by the authors as bots. The authors of this work develop a whole new unsupervised detection method inspired by DNA sequencing. It accomplishes detection by finding accounts with certain suspicious similarities which are shown to be a strong indicators of bot accounts.

The main reason for bringing up examples of previous works on bot detection is to show that there usually exists at least some obscurity regarding the collection of data in this field. How has the sampling been performed? Is there any potential bias? How have the accounts been labeled and how have those labels been defined? All these questions are good to keep in mind when analyzing the results of this thesis, which has used the dataset made available by S. Cresci et. al. in [4]. It, too, will come with certain question marks, as will be shown in section 4 and 5.

Categorization

All methods for binary classification of Twitter accounts suffer from one weakness: they cannot distinguish between different kinds of bots. For some tasks, this might not be a problem. However, if the goal is to, for instance, estimate the populations of different bot types, multinomial classification is more appropriate.

There indeed seem to be fewer works on this topic, but they do exist. One example is the work by Z. Chu et. al. [2]. Here the authors use a somewhat intricate technique, involving both Bayesian classification and LDA, to classify accounts as either humans, bots, or cyborgs, which is a mix between human controlled and automated account. As in many previous works, the data is manually annotated by the authors.

A work that does not use multinomial classification, but which none the less attempts to distinguish between different types of accounts, is the work by O. Varol et. al. [25] mentioned above. Although the authors use binary classification, they later employ K-means clustering to identify possible subcategories in the data. Their results indicate three subcategories for bots, ranging from simple spammers to more legit-looking, self promoting accounts, one subcategory for more sophisticated bots akin to the cyborgs in [2], and then several subcategories for genuine human accounts.

One issue which is amplified in multinomial classification of Twitter accounts is the definition and demarcation of account classes or categories. Although this is not a trivial problem in the binary case either, for each new bot type introduced into the classification environment, the difficulty of the task increases severely. As [25] shows, bots can be subdivided into numerous subcategories, making it hard to know exactly where to draw the line, and how many categories to actually use. This problem is further discussed in section 2.3.1, where the author defines the account classes used in the work, and presents some arguments for why using these definitions could be justified.

1.4 Outline

A brief introduction to Twitter and the issue of Twitter bots has been provided in section 1. Previous related work has also been discussed here.

Next up is a thorough discussion of the data used here in section 2. What types of accounts are actually available in the studied data set? How well does classification of these sampled accounts generalize to classification of Twitter accounts in the real world? These questions will be answered here.

In section 3, a broad range of methodological aspects are looked into, stretching from balancing classes to model construction. Measures of accuracy, which will be crucial for determining model performance, are also brought up in this section.

Following are the results in section 4. Model performance on test data will be presented, together with tables and figures illustrating what features may have been most relevant for the classifiers.

Finally, section 5 draws from the experience of working on this thesis, discussing how to interpret the results, and giving recommendations for future research. Section 6 sums it all up with some concluding remarks, answering the research questions, but also touching on unforeseen outcomes of the analysis.

2 Data

The data used in this thesis was kindly provided by Stefano Cresci and the other authors of the 2017 paper *"The paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race"* [4]. It comes in *two folders*, one containing so called *user data* and the other containing so called *tweet data*. The data as such is organized in different spreadsheets. Each of the two folders contains multiple spreadsheets, each spreadsheet in turn containing data for only one of the four account categories *genuine users, social bots, traditional bots* and *fake followers*. Depending on which folder you are looking at, the rows and columns of the spreadsheets therein will denote different things.

Beginning with the *user data folder*, each row of any spreadsheet in this folder represents *one unique Twitter account*, and the columns hold different pieces of information about these accounts. For instance, one of the columns holds the *Twitter ID*, which is a unique number that every Twitter account has. Other examples of columns are *username, number of followers, the date the account was created*, and so on. This is why it is called the *user data*, because it deals with information regarding the account (user) as such. In other words, there is no information about tweet *contents* in these spreadsheets.

Specifically, the user data folder contains one spreadsheet for genuine users, three for social bots, four for traditional bots, and one for fake followers. When one category has several spreadsheets, it is due to the fact that the data points in it have not been sampled in the same process, and thus each sampling process is given its own spreadsheet.

In the *tweet data folder*, the spreadsheets have a different structure. Here, instead of representing a unique account, each row represents *one unique tweet*, with the columns signifying tweet specific information such as *tweet text, number* *of likes, date of posting*, and so on. However, there is also a column which contains the *Twitter ID* of the account that posted each tweet. Thanks to this, any row (tweet) in a tweet data spreadsheet can be connected to a row (account) in its corresponding user data spreadsheet. There may of course be multiple tweets for any given account, which is why the tweet data will need to be aggregated (see section 2.2).

Now, there is again one spreadsheet for genuine users, three for social bots and one for fake followers. These correspond to the ones in the user data folder. However, for the traditional bot category, there is here only one spreadsheet instead of four, meaning that only the spreadsheet in the user data folder corresponding to this tweet data spreadsheet will be used. Thus, since every user data spreadsheet will have a corresponding tweet data spreadsheet, together they will be referred to as *spreadsheet pairs*. In the subsequent analysis, each Twitter account will be viewed as an observation. These observations will have features computed both from the user data and tweet data connected to them, and will be assumed to belong to one of the four categories (classes) mentioned above.

In the next section, the actual accounts and how they were sampled will be described in a little more detail. After that follows an explanation of how the features used in the machine learning models were computed. Finally, the viability of drawing general conclusions from the available data will be discussed.

2.1 Data labels

There are four data labels given by the data: *genuine users, social bots, traditional bots* and *fake followers*. These represent the four categories that will be used in the classification. Note that the label *social bots* is here not referring to the generic term as used in section 1.1, but specifically to a type of fairly new and sophisticated social bots, often posing as humans on Twitter. The exact definitions and demarcations of the different categories are discussed in section 2.3.1.

Genuine users

The genuine user data was collected by the authors of [4], by randomly sampling thousands of Twitter accounts and asking each account a question in natural language. The 3 747 of these that actually responded were then verified as real humans. These accounts thus have one row each in the user data spreadsheet for genuine users. However, only a part of these, 1 083 accounts, actually have corresponding tweet data, and it is those which will be used in the classification. The reason why not all accounts have available tweet data is not clear.

Social bots

The *social bot* category is represented by three spreadsheet pairs of user and tweet data. The first pair was collected in connection to the 2014 Mayoral election in Rome. It contains 991 accounts which were used to promote one of the campaigns by retweeting any post that the candidate tweeted.

The second pair contains 3 457 accounts who were found promoting a phone application for hiring people to do artistic work such as digital photography and music.

Continuing on the theme of product promotion, the bots in the third pair advertised products with reduced price on Amazon.com. This pair contains a total of 464 accounts.

Traditional bots

As mentioned in the beginning of section 2, the traditional bot category has four spreadsheets with user data, but only one of them has a corresponding spreadsheet with tweet data. There is thus only one spreadsheet pair for this category. The data in these spreadsheets was introduced by [27], and is a collection of accounts which were found to be spamming links to different types of malware, such as computer viruses. In total there are 1000 accounts in this category.

Fake followers

Finally, *fake followers* are accounts which only serve the purpose of boosting someones follower count. There are websites offering such accounts for sale, which is how the authors of [4] have acquired the data in the fake followers spreadsheet pair. It is made up of 3 351 accounts, purchased from three different Twitter online markets. Like with genuine users, for some reason not all these accounts have tweet data available, so only the 3 202 accounts which do will be used.

2.2 Features

Figure 1 visualizes how each Twitter account, i.e. each *observation*, is represented in the data, with one row in a user data spreadsheet giving information about account statistics, and a number of rows in a tweet data spreadsheet, each row giving information about an individual tweet posted by the account. The information available is exemplified by some column names for both spreadsheet types. The figure further illustrates how this information is processed and merged into a full set of features, ready to be used.

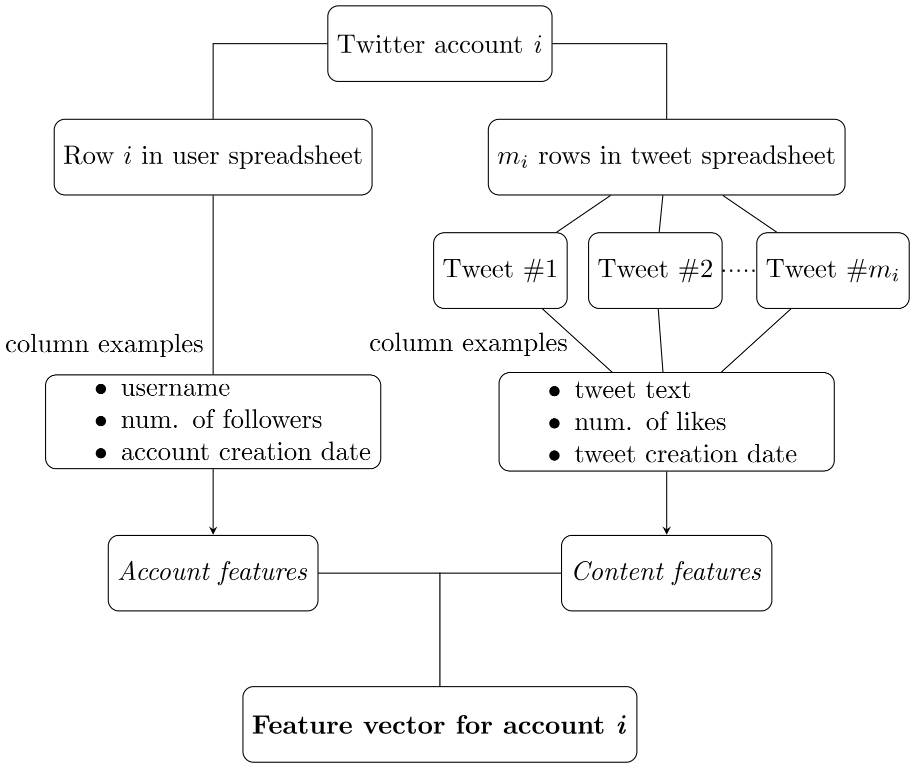

Figure 1: Flowchart describing the data structure. Arrows going from variable examples to account and content features respectively emphasize that features are calculated from the available variables (for instance, using the "tweet text" from all m_i tweets for account i, the average length of tweets can be calculated and used as a feature. Similarly, the username for account i can be used to calculate length of username, which can also be used as a feature).

Now, the features are divided into two categories: *account features* and *content features*. They correspond schematically to the user data and tweet data in the sense that account features are computed from user data, while content features are computed from tweet data.

2.2.1 Computing features

Account features are those describing the *account as such.* Length of username, age of the account, number of followers, and so on. In many cases, these features are given directly in the user spreadsheets. An example is the *number of followers*, which exists as a column in these spreadsheets, and thus needs no particular computations to be obtained.

Other account features, however, need additional computations for their creation. *Length of username* is one such example. In the user data spreadsheets, there is a column in which the username of the account in question is given. This, of course, cannot be used directly as a feature, since it is just a text. Instead, a computer program counts the number of characters in the username, thereby creating the new feature *length of username.*

As for the content features, the idea is to give an aggregated picture of an account's *tweets and tweeting behaviour.* Now, none of these features are available directly in any spreadsheet, but must all be computed in one way or another. This is due to the *one-to-many* relation between an account, represented by *one* row in a user spreadsheet, and the tweets posted by that account, represented by *many* rows in a tweet spreadsheet. Thus, any type of tweet data must first be aggregated in order to be used as a feature.

An example of such aggregation is the computation of different statistical measures over a number of tweets. One case where this comes into play is in the length of tweets. In all tweet spreadsheets, there is a column giving the actual text content for each tweet. Using this, a program can count the number of characters in each tweet posted by a given account. The program can then compute any desired statistical measure, like mean or standard deviation, for these counts. By doing this, the *one-to-many* relation between an account and its tweets is replaced by a *one-to-one* relation between an account and a measure computed over those tweets. Subsequently, since each account is viewed as an observation, this makes it possible to use those measures, which are basically aggregated tweet data, as features in a classifier.

There are other examples of how tweet data can be aggregated in order to yield content features. These will be discussed in section 2.2.3, where a more thorough explanation of the features actually used in this thesis is provided.

2.2.2 Feature selection

The reason feature selection is even needed in the first place is of course that the possible number of features that can be computed is simply too large. A natural way to perform this selection is to use some set of features that has been shown to work well in previous works.

For this thesis, the work done by J. Fernquist et. al. [8] has been chosen as a guide for picking features to include in the models. This is mainly due to the fact that in their work, the authors have used the same data as the one used here (i.e. the data from [4]), meaning that most of the features they use should be possible to compute from the available data, without having to download additional information from Twitter. Furthermore, the selection seems well balanced and relevant, and has yielded promising results for binary classification of Twitter accounts.

The authors of [8] have used 140 features in total. However, not all of these features will be used in this thesis. Some of them have been added by the authors themselves by downloading additional Twitter data, and so will not be used. For a few other features, it has been difficult to understand exactly how they have been computed. These features have been left out as well. Additionally, in [8], the content features include a couple of population standard deviations of different measures, i.e. standard deviation of certain measures (e.g. *length of tweets*) calculated for all tweets available, and not only the most recent tweets. Due to computational complications, these features have been removed too.

All in all, 20 out of the original 140 features have been omitted. Only one of them was on the list of the highest ten feature importances in [8], so this reduction in the number of features should not have too large of an impact on model performance.

2.2.3 List of features

Table 1 shows the full list of all features that have been used in the classification. It is important to note that some entries in the list actually encompass multiple features. One example of this is *weekdays tweeting* in the content feature column. This entry actually represents seven features, each feature holding the count of tweets posted during one of the seven week days. Additionally, for content features, every entry marked with an asterisk (*) signifies that for that entry, five features in the form of mean, median, standard deviation, minimum value and maximum value of the variable in question, have been computed.

However, unless specified in the explanations below, the entries in the content feature column correspond to one feature, namely the mean value of whatever variable the entry denotes. For instance, the entry *hashtags per tweet* is a feature measuring the mean number of hashtags per tweet for each account. For the account feature column, all entries correspond to one feature each.

Before continuing to the detailed explanations of features below, note that none of the content features are computed using all tweets available in the tweet spreadsheets. Instead, only the 100 most recent tweets from each account have been used for these computations, with the exception to this being when an account has less than 100 tweets, in which case all available tweets are used.

Now, some of the entries in table 1 are quite straight forward, like *age of account, number of tweets*, or *number of words*. For other entries, however, a clarification might be in order.

Follower-friend ratio First off, a *friend* in Twitter lingo is someone being followed by someone else. So if account A follows account B, B is a A:s *friend*, while A is B:s follower. Thus, *follower-friend ratio* signifies the relation between

Account features	**Content features**
Age of account	Hashtags per tweet
Follower-friend ratio	Hours of day tweeting
Given likes per follower	Length of tweets *
Given likes per friend	Mentions per tweet
Has location	Normalized distribution hours tweeting
Has default profile description	Normalized distribution of tweet endings
Length of username	Normalized distribution weekdays tweeting
Likes per day	Number of words *
Number of followers	Retweets achieved per tweet
Number of friends	Retweet-tweet ratio
Number of likes given	Time between mentions *
Number of tweets	Time between retweets *
Tweets per day	Time between tweets*
	Time between urls *
	Unique hashtags per tweet
	Unique mentions per tweet
	Unique sources
	Urls per tweet
	Weekdays tweeting

Table 1: Features used in the classification (: Computing the measures mean, median, standard deviation, maximum value and minimum value)*

the number of users following an account and the number of users being followed by that account.

Has location & Has default profile description On Twitter, users can choose to write a short description about themselves that shows up when you visit their user page. If they do not write anything, they are said to have a default profile description. Users can also choose to state a location, for instance the city in which they live. These two profile attributes are captured by the features *has location* and *has default profile description*, which are binary and take the value one if true and zero if false.

Likes per day & Tweets per day Features counting something *per day* (e.g. *likes per day*) are computed simply by taking the count of the given measure and dividing it by the age of the account in days. Knowing the creation date of the account, the age can be easily computed.

Hours of day tweeting & Weekdays tweeting For each Tweet, the exact time and date of posting is recorded. Using the information about time of posting, it is possible to count the number of tweets a user has posted during each hour of the day. By doing this, 24 features can be created, each representing the number of tweets posted by the account during the respective hour. In the same way, using the posting date information, the number of tweets posted on each of the seven weekdays can be counted, yielding seven features.

Normalized distributions Here, a normalized distribution simply means that counts are transformed into percentages. For instance, *normalized distribution hours tweeting* is computed from *hours of day tweeting*, by using the counts to determine what percentage of a user's tweets were posted on any given hour of the day.

Tweet endings The text in a tweet can end in many different ways; a period, a question mark, a digit, and many more. Looking at each of an account's tweet texts, the number of different tweet endings, i.e. the last character in a tweet text, can be counted and normalized accordingly. This is what is done in the features behind *normalized distribution of tweet endings*; each tweet ending is represented by a feature, and the value of the feature is the percentage of a user's tweets which end with that particular character. Since it would not be possible to create a feature for every possible tweet ending, a number of tweet endings are selected. Tweets not ending with any of the characters in that list count toward the feature *other tweet endings.* Tweet endings selected to create features are *period, exclamation mark, question mark, digits, lower case letters* and *upper case letters.* (Note that this may not be the same tweet endings selected in [8], since these were not explicitly stated by the authors).

Retweet features Each post made by a Twitter account can either be a tweet, i.e. a post composed by the user himself, or a retweet, which basically means reposting or forwarding a post made by someone else. The number of a user's posts which are retweets relative to the number which are tweets is what the feature *retweet-tweet ratio* measures. The *retweets achieved per tweet* feature, on the other hand, measures how many times per tweet the user is retweeted by someone else on average.

Time between events As mentioned above, the date and time of posting is recorded for each tweet. This makes it possible to compute the time that passed between the posting of two tweets. Applying this computation to the 100 most recent tweets, things like mean time between tweets as well as mean time between tweets containing hashtags, mentions, or URLs (together with other statistical measures like median and standard deviation), can be calculated and used as features. (Note: a mention is when a tweet is directed at a specific user by stating their username preceded by an at sign(@) in the tweet).

Unique instances Just as the number of hastags per tweet can be counted, the number of unique hashtags can be counted. This entails counting each hashtag only once, regardless of whether it is used multiple times in multiple tweets. The same goes for mentions and sources. To clarify, the source of a tweet is the utility used to post the tweet, for instance a mobile app or a web page. Hence, if a user always posts their tweets through the same mobile app, that user will only record one unique source. Unlike most other content features, *unique sources* is not a feature averaged over the number of tweets, but simply the total number of unique sources. It is worth wile to note that since the unique hashtags and mentions are found via text searches, there is a risk that these counts are not completely accurate. For instance, if someone uses the at sign in front of a word which is not a username, it will still count toward the unique username feature. However, this can be assumed to be a minor issue with little impact on the results.

2.3 Generalizing from available data

One issue that arises immediately when trying to develop any sort of Twitter bot classification method, be it binary or multinomial, is the question of how well the classification methods developed in the study generalize to the real world. For the methods to generalize, it requires that *a)* the categories assumed in the study truly exist as disjoint classes in the real world, and subsequently *b)* that these classes are indeed well represented in the sampled data.

In other words, the viability of generalizations is an issue of *properly defining bot categories* on one hand, and *sampling procedure* on the other.

2.3.1 Defining categories

In this thesis, three different bot categories, or classes, are assumed, together with a non-bot category, genuine users, with the hope that these truly correspond to disjoint categories in the real world. Note that these bot categories do not necessarily correspond to the categories drawn up in the list in section 1.1. Instead, they are to be viewed as separate categories which simply fall under the "bot" term as defined in the end of that section.

For Twitter bots, the most interesting aspects, those that would justify separation into different categories and at the same time make these categories disjoint (or as disjoint as possible), include how the bots *behave* and what their *purpose* is. Thus, when trying to create relevant categories, these two aspects ought to be given large weight.

To begin with, the idea behind the chosen categorization into social bots and traditional bots is, as shown in [4], that from what used to be simple spambots (traditional bots), the emergence of a new type of bot has, by now, been well documented. Due to their ability to imitate humans, these new bots, called social bots in reference to this ability to act "social", are supposedly significantly harder to detect. In addition, these types of bots generally tend to be used for different purposes. While the traditional spambots often tweet links to different malicious software, social bots are mainly used for marketing purposes, for instance to influence public opinion in some political issue or to promote some product. Because of distinctive behavioral patterns as well as distinctive purposes of use, the division into social and traditional bots is deemed justifiable and reasonable.

As for fake followers, the behavioral patterns play a smaller role in putting them in a category of their own, although it is expected that they will exhibit behaviour different from that of the other two bot categories. Here it is instead mainly the purpose for which these accounts are used that justifies the separate category, and that purpose is to boost a users follower numbers. This is a wholly different purpose than for the above mentioned traditional and social bots, giving legitimacy to the assumption of this third category.

Since there is nothing that prevents fake followers to be used for other purposes than boosting someones Twitter stats, accounts in this class could potentially behave in ways which would equally well justify categorizing them as any other bot type. Although this is a weakness of the class definition, it is assumed that the behaviour of fake followers is notably more passive than that of other bot types. This assumption is supported by table 4, which shows that on average, fake followers post significantly fewer tweets per day than other account types.

Using genuine users as a category goes without saying. Although it is not completely trivial how to define this category (how much automation can an account employ before deemed a bot?), it is probably still the most well defined category, namely real humans predominantly tweeting manually. Assuming such a class in the classification is justified by the sheer fact that it is meant to include all accounts which are *not* bots.

These definitions of three bot categories and one non-bot category may not be the ones best representing the real world. There is always a risk of there being accounts for which two or more of the categories overlap, or which do not really fit into any of the categories. By creating more categories, either by further subdivision or by introducing wholly new categories, these risks could potentially be reduced. Another way of handling the issue could be to employ multi-label classification, i.e. allowing each observation to actually belong to several classes at once. However, due to limitations in time and in the scope of this thesis, as well as practical limitations posed by the available data, other categorization schemes are not explored. It is instead assumed that the categories presented above indeed exist in the real world and that they are disjoint, or at least that their intersections are marginal, and the labels given in the data set are taken as ground truth for these categories. To what extent this is reasonable of course depends on how the data has been sampled.

2.3.2 Sampling procedure

A description of the different spreadsheet pairs and their origins was provided in section 2.1. However, to what extent they truly fit into the presumed categories, and how well they represent them, still remains to be answered.

For fake followers, the sampling is very straight forward. The authors of [4] have simply bought these accounts from three different online markets for these kinds of products. There is almost no doubt that this sampling procedure will yield observations belonging to the correct category, since a bought account employed to boost follower numbers is the very definition of a fake follower. There is, however, the question of sampling bias. That is, whether these particular accounts, although members of the correct category, are truly representative of that category. Perhaps accounts bought from other websites will exhibit different properties than the ones used here? This is something which one should keep in mind during the following analyses.

Genuine users, too, have been sampled in a fairly transparent way, namely via random sampling, aided by use of natural language. As explained in section 2.1, a large amount of accounts have first been sampled randomly, and each account sampled has been asked a question in natural language. Then, the accounts that responded have been verified to indeed belong to a human users [4], although it is a bit unclear exactly how this verification has been performed. A guess is that the authors have looked at the quality of the answer, potentially together with some other parameters for the account. Despite this minor obscurity, the likeliness of a bot passing both these controls can reasonably be viewed as slim. Thus, the subset of genuine users indeed most likely contains an overwhelming majority of human, non-bot accounts. In addition, using random sampling ought to have brought a relatively unbiased sample to begin with, although there might of course have been some new bias introduced in the verification process, for instance by picking only those who answered the control question.

Traditional bots have been sampled by the authors of [27]. Here, the authors have used a graph sampling technique from [17] to initially randomly sample a nearly 500 000 Twitter accounts and their latest tweets. These tweets have been analyzed by special computer programs to see if they contain URL:s linking to malicious websites. Accounts which have posted large amounts of these links (more than 10% of their tweets containing at least one such link) are examined manually to verify that they are truly spammers. Given that the graph sampling performs well in terms of random selection, the large initial sample should have been decently unbiased. Picking out the spambots, however, includes a number of more or less arbitrary measures, which could potentially skew or contaminate the sample. For instance, any spammers not exceeding the 10% limit would have been excluded. In the same way, any real users exceeding it might have been incorrectly included. The odds for this, however, ought to be viewed as slim given the subsequent manual verification, which of course is a potential source of bias in itself. Taken together, it does not seem very likely that the subset would contain accounts which are not traditional spambots. There is however a non-negligible risk that some accounts belonging to this category were excluded due to different biases in the filtering process.

Lastly, there are the social bots, which have been collected in three separate sampling processes. It is not clear how any of these sampling processes have been conducted, since the authors of [4] simply state that they have obtained the data without mentioning how. Thus, the extent of possible sampling biases is virtually impossible to assess here. The verification of these accounts as social bots, however, is commented in [4], albeit briefly. Out of the originally suspected social bot accounts, only those passing yet another manual verification process have been added to the final data sets (i.e. the spreadsheet pairs used in this thesis). The verification has been carried out by way of account comparison between the social bots, seeing which accounts behave similarly, excluding the most distinguishable outliers. No further details about this manual annotation process are provided, making it difficult to have a well informed opinion on how likely it is to yield a correct assessment. However, the described behavior of these accounts, for instance as retweeting every single post of a specific political candidate, fits well into the behaviour one would expect to see from social bots. Thus, given that this was truly how the accounts behaved, and given that the authors of [4] choose an appropriate way of double checking it, the final, verified data should indeed mainly be made up of social bots. Still, out of all the account categories, this is the one whose sampling methods are shrouded in most uncertainty.

Even though parts of the sampled data might raise some concerns, there is not much that could be done to remedy these. The main purpose of this section is merely to make the reader aware of weaknesses in the sampling that might cause problems when trying to generalize the obtained results to a wider population of Twitter accounts. In addition, these weaknesses can also affect the classification performance for different classes in this thesis, which is good to keep in mind.

3 Methodology

In this section, performance measures, balanced classes and model training, testing and validating, are discussed. After that follows explanations of each of the four machine learning methods used in this thesis.

3.1 Measures of model performance

To evaluate and tune the models, as well as to make comparisons between the performances of different methods, some measure or measures of this performance is needed. In multinomial classification, some measures commonly used are averages of *accuracy, precision* and *F-score* [23]. These averages can be predicted either by computing the measure for each class separately and then averaging over the number of classes (macro average), or by using all the classes at once to compute the average (micro average). In this paper, macro averages have been used.

Another measure of model performance is the *Matthews Correlation Coefficient (MCC)*, also known as the phi coefficient. Although this was originally developed for binary classification problems, a generalization for multinomial cases has been developed in [10]. The value of the MCC can range between -1 and +1 depending on the direction of the correlation, and is basically a measure of how well the predicted class belongings correlate with actual class belongings.

3.2 Balanced classes

The number of observations differ quite a lot between the different classes. For social bots and fake followers, the number of observations exceed 3 000, while genuine users and traditional bots only have around 1 000 each due to missing data. Although these imbalances are not huge, they can still cause problems in classification, as the classifiers are likely to be biased in favor of the more common classes [15]. That is, the classifiers are likely to be better at classifying observations from the more common classes than from the more uncommon ones when trained on imbalanced data.

Moreover, as explained in section 2.1, these imbalances are not due to any natural abundance of social bots and fake followers on Twitter, but rather to the fact that the class subsets have been sampled in different ways, on different occasions, and sometimes by different authors. Therefore, the classes will be balanced by means of *undersampling*. This entails removing observations at random from the classes with largest abundance until all classes have the same number of observations (1 000 observations in this case). A potential drawback from this is of course the loss of potentially useful information. However, seeing as the author does not have any prior knowledge about the true distribution of the account types, and that 1 000 observations per class is still a relatively large amount, this is deemed to be a reasonable approach. Thus, the dataset which is split into a training and a test set (see next section) will contain 4 000 observations.

3.3 Training, validating & testing

The proportions by which to split the data into training and test sets is chosen to be 4:1, that is, 80% of the data (3 200 observations) goes toward the training set and 20% (800 observations) toward the test set. No validation set is created. Instead, model tuning, i.e. selecting hyper-parameters, is performed via cross-validation (see below).

Hyper-parameters are parameters which are not estimated directly by the model but must be chosen by the model creator before the *training of the models* can begin. Examples are the number of trees in a random forest, the weight of the penalty term in regularized logistic regression, the learning rate in artificial neural networks, or the number of neighbors k in k-nearest-neighbors.

To avoid making this choice completely arbitrary, each model undergoes a *validation process* where different combinations of hyper-parameters are tried out.

The method used to try them out is *5-fold cross-validation.* This method works by randomly splitting the training data into five parts. The model with the combination of hyper-parameters in question is then fitted on four of these parts, and thereafter set to classify the observation in the part which was left out. A measure of model performance, accuracy, is calculated, after which the process of fitting and classifying is repeated, but now leaving out another part, until each part has been left out once. Finally, an average of the five different accuracy measures obtained is recorded. After this, another combination of hyper-parameters are applied to a model, which goes through the same process. When all selected combinations of hyper-parameter have been tested in this manner, the combination that yielded highest averaged accuracy is nominated as the final model to be used on the test set [12].

When each machine learning method has nominated a model through the trials of cross-validation, these models are employed to classify the observations in the test set. From these classifications, the measures discussed in section 3.1 are computed and presented as the verdict of each method's merits.

3.4 Random forests

To understand what a random forest is, one must begin with the concept of decision trees, and then move on to bootstrap aggregation (bagging) and decorrelation of predictors [14]. In this section, a brief walk through of each of these concepts is provided, together leading up to an explanation of the random forest model.

3.4.1 Decision trees

In the case of classification, a decision tree can be defined as a number of cut-off points, used to make a series of decisions, the last of which being the decision on how to classify any given observation.

Training (fitting) a classification tree is performed by creating a number of *regions in feature space.* A region is basically a set of splitting rules. An observation is said to belong to a given region if the feature vector $\boldsymbol{x}_i$ of that observation conforms to all splitting rules set up by the region. These rules take the form of binary decisions, for example $x_1 < v_1$ or $x_2 \geq v_2$, where v_j are cut-off points in feature space. Naturally, splitting rules always come in pairs of events and complementary events. This means that if the splitting rule $x_j < v_j$ is used to define one region, $x_j \geq v_j$ will be used to define another, complementary, region. Thus, new regions are created by finding cut-off points along which to split the observations belonging to one particular region into two new regions.

The cut-off points v_j along which to perform the splits are found by optimizing a measure of quality of these splits. Here, this measure is chosen to be the *Gini index*, which is defined as

$$G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}),$$

where p_{mk} is the proportion of observations from class k in region m, assuming there are K classes in total. In other words, there is one Gini index measure for each region. When a cut-off point is considered, the average Gini index for the resulting two hypothetical regions, weighted by the number of observations falling into each region, can be computed. This average can then be compared to the Gini index in the region to which the observations currently belong, yielding a measure of change in Gini index. Running such comparisons over all existing regions, for each region examining all features and their potential cut-off points, the split creating the largest reduction in Gini index can be found, and those two new regions can be created. Repeating this process of splitting until some stopping criteria is reached, for instance until all regions contain five or fewer observations, yields the final, trained model.

With this model, any new observations can be classified by first determining which region it belongs to, and subsequently applying the principle of majority vote.

3.4.2 Bootstrap aggregation (bagging)

The predictions of a single classification tree will be highly dependent on exactly which observations were selected to use as training data. Trees trained on different data partitions are likely to exhibit highly variable results in terms of performance on test data. This problem of *high variance* can be reduced however. Instead of training a single tree, multiple trees can be trained, classifying any new observations by assigning it to the class it has been assigned to by most of those trees.

However, to train multiple trees, multiple training sets are needed. These can be obtained via *bootstrapping.* The technique entails drawing observations from the training set with replacement, until the size of this newly drawn training set is equal that of the original one. This procedure can be repeated any number of times to yield B such bootstrapped training sets.

For each of the bootstrapped sets, a classification tree is trained, consequently yielding B trees in total. The predictions over all these trees are then aggregated to yield the final classification via majority vote as described above. Hence the term *bootstrap aggregation*, or *bagging.*

3.4.3 Decorrelating features

A random forest model is basically bagging with a twist. Although bagging reduces variance, the reduction might not be very large if there exists some feature which is a notably stronger predictor than the others. This is because the particularly strong feature is likely to be used in the first split by almost all the trees, leading to a *high correlation* between their predictions. In other words, many of the trees are likely to yield very similar regions in feature space, thereby producing very similar classifications. Taking the majority vote of such classifications then somewhat defeats its purpose, since virtually the same result could be achieved by using just one or a few trees, meaning variance is not reduced nearly as much as if the predictions would be uncorrelated.

To counteract this, the random forest method uses a randomly selected subset of all features when searching for each optimal split. By doing so, the method ensures that no feature is allowed to dominate the early splits of the trees, allowing the trees to be fitted in more various ways. The predictions of these trees will consequentially be *decorrelated*, and taking the majority vote of their classifications will thus yield a larger reduction in variance than in the simple bagging case.

The hyper-parameters that will be tuned as described in section 3.3 are hence the number of trees B, and the number of features searched in each split.

3.4.4 Feature importance

Due to its use of bagging, one downside of the random forest method as compared to a single classification tree is the loss of interpretability. In a single tree, the regions in feature space created can be studied, something which is not possible when there are perhaps hundreds of trees involved.

A way to make sense of the features in a random forest is instead to calculate the *feature importance*. This is done by recording the average reduction of the Gini index for each feature over all B trees. That is, the reduction in the Gini index achieved by all splits along a particular feature, averaged over all trees, and computed for all features. A large value indicates that the feature tends to create fairly qualitative splits, making it a relatively useful and important feature. In figure 3, this has been visualized for the 15 features with highest importance for the actual training data.

3.5 Logistic regression

Multinomial logistic regression builds on the transformation of linear combinations of features, a linear predictor, to produce estimates of class probabilities. To understand how this model is trained, one must first understand the underlying multinomial distribution, the logistic response function used to transform the linear predictor, and the maximum likelihood method employed for the estimation of the coefficients in this predictor.

3.5.1 Multinomial distribution

Assume that for a given population, there exist $K+1$ disjoint classes that any individual can belong to. Sampling one observation from this population, the class belonging of that observation can be described by the vector $\boldsymbol{y}$ of length K. In this vector, each element y_k corresponds to a class and takes the value of either one or zero depending on if the observation belongs to that class or not. If the observation belongs to the $K+1$:th class, the reference class, all elements are instead zero, i.e. $\boldsymbol{y} = \mathbf{0}$. The vector denoting the probabilities of classes $1, ..., K$ is defined as $\boldsymbol{\pi} = (\pi_1, ..., \pi_K)$. Again, the probability for class $K+1$ is simply $1 - \pi_1 - ... - \pi_K$.

The distribution of $\boldsymbol{y}$ is then the multinomial distribution, and its probability mass function is thus:

$$f(\boldsymbol{y}|\boldsymbol{\pi}) = \pi_1^{y_1} \cdot ... \cdot \pi_k^{y_K} \cdot (1 - \pi_1 - ... - \pi_K)^{1-y_1-...-y_K}. \tag{1}$$

This means that the probability of an observation belonging to class $k \in \{1, ..., K\}$ is

$$P(y_k = 1|\boldsymbol{\pi}) = \pi_1^0 \cdot ...\pi_k^1 \cdot ... \cdot \pi_K^0 \cdot (1 - \pi_1 - ... - \pi_K)^{1-0-...-1...-0} = \pi_k,$$

and subsequently that the probability for class $K+1$ is:

$$P(\boldsymbol{y} = \mathbf{0}|\boldsymbol{\pi}) = \pi_1^0 \cdot ... \cdot \pi_K^0 \cdot (1 - \pi_1 - ... - \pi_K)^{1-0-...-0} = (1 - \pi_1 - ... - \pi_K).$$

Viewing the class probabilities π_k as functions of linear predictors, class probabilities for any given observations can be modeled. For this, however, a function like the logistic response function is needed [7].

3.5.2 The logistic response function

The logistic response function is a way to transform a linear predictor to ensure that the response is bounded between zero and one, which is necessary for modeling probabilities. Assuming a feature vector $\boldsymbol{x}_i$, the linear predictor for a class k is defined as $\eta_{ik} = \boldsymbol{x}_i'\boldsymbol{\beta}_k$. Then, for any observation i, the class probability for class $k \in \{1, ..., K\}$ can be modeled in the following way:

$$P(y_{ik} = 1|\boldsymbol{x}_i) = \pi_{ik} = \frac{\exp(\eta_{ik})}{1 + \sum_{l=1}^{K} \exp(\eta_{il})} \tag{2}$$

where the right hand side of the equation is the *logistic response function.* For class $K+1$, the probability is subsequently modeled as:

$$P(\boldsymbol{y}_i = \mathbf{0}|\boldsymbol{x}_i) = 1 - \pi_{i1} - ... - \pi_{iK} = \frac{1}{1 + \sum_{l=1}^{K} \exp(\eta_{il})}. \tag{3}$$

No matter what values any of the linear predictors η_{ik} take, these functions will always output a number between zero and one. Combining this way of modeling probabilities with the probability mass function of the multinomial distribution, the maximum likelihood method can be employed to estimate the regression coefficients $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K)$.

3.5.3 Maximum likelihood estimation

If $\boldsymbol{y}_i$ follows a multinomial distribution with its parameters $\boldsymbol{\pi}_i$ modeled as a functions of linear predictors, the likelihood function for a sample of size n from this population is defined as:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} f(\boldsymbol{y}_i|\boldsymbol{\pi}_i)$$

with $f(\boldsymbol{y}_i|\boldsymbol{\pi}_i)$ being the probability mass function from eq. 1. Since the functions in eq. 2 and 3 are inserted into the probability mass function, it is clear how the likelihood indeed becomes a function of parameters $\boldsymbol{\beta}$.

Developing this further, the log-likelihood becomes

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_{i1} \log\left(\frac{\exp(\eta_{i1})}{S_i}\right) + ... + (1 - y_{i1} - ... - y_{iK}) \log\left(\frac{1}{S_i}\right)$$

where $S_i = 1 + \sum_{l=1}^{K} \exp(\eta_{il})$. With that, the estimates $\hat{\boldsymbol{\beta}}$ can be computed via the equation $\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbf{0}$, solving for $\boldsymbol{\beta}$ by an iterative method [7].

3.5.4 Regularization

To optimize the coefficient estimation for model performance on new data, regularization using an L_2 penalty term will be employed. This is done to constrain the regression coefficients in order to avoid overfitting. The tuning parameter used to determine the size of the penalty term will be selected via cross-validation (see section 3.3). For regularization, standardized features, i.e. mean zero and variance one, are necessary to ensure all features are treated equally [12].

3.6 Artificial Neural Networks

The artificial neural networks (ANN) derive their name from originally being used to model the connection between synapses in human brains. Seeing as there are many different kinds of ANN:s, constraints on the number of models tested is needed. Therefore, this thesis will only be exploring the multiple hidden layer back-propagation network.

3.6.1 Inputs, outputs & hidden layers

An ANN has three types of layers, each layer having a certain number of units, or *neurons*. The *input layer* is where the feature values of an observation are inserted and fed to the next layer. If there are p features, there are $p+1$ neurons in this layer, the added neuron constantly outputting a one to introduce a bias term, much like the intercept in a regression.

The *output layer* produces estimated probabilities for each class. Naturally, if there are K classes, this layer has K neurons. Each neuron k in this layer is fed a linear combination of the outputs from the previous layer. It then transforms this linear combination using the *softmax* function (see below), yielding an estimated probability of the given observation belonging to class k.

Between the input and output layers are the *hidden layers*. A hidden layer transforms linear combinations of the output from the previous layer using an *activation function* σ. These transformations are then fed forward to the next layer. Just like with the input layer, each hidden layer introduces a bias term by having a neuron which always outputs a one. The number of hidden layers, as well the number of neurons in each hidden layer, are hyper-parameters, and will be chosen via cross-validation.

To clarify how classification is actually performed using this method, assume there is a feature vector $\boldsymbol{x}_i$, and that the network in question has L hidden layers l. Now, each neuron j in the *first* hidden layer takes in a linear combination $\boldsymbol{w}_j\boldsymbol{x}_i$, in turn outputting $z_j = \sigma(\boldsymbol{w}_j\boldsymbol{x}_i)$. The activation function σ is chosen to be the *sigmoid function* $\sigma(v) = 1/(1+e^{-v})$, which is a common choice [12]. The same sort of transformation is produced by all the hidden layers, each hidden layer l using linear combinations of the output vector from the previous hidden layer $\boldsymbol{z}_{l-1}$ to compute its output.

Finally, the output vector of the last hidden layer L is computed and fed to the output layer. Each neuron k in the layer is thus given a linear combination $\boldsymbol{w}_k\boldsymbol{z}_L$, which is transformed via the softmax function to yield the estimated probability p_k for class k:

$$p_k = \frac{\exp(\boldsymbol{w}_k\boldsymbol{z}_L)}{\sum_{l=1}^{K}\exp(\boldsymbol{w}_l\boldsymbol{z}_L)}.$$

Observation i will then be classified as belonging to the class for which the estimated probability is highest.

If the class of observation i is known, the predicted class probabilities can be compared to the actual target vector $\boldsymbol{y}_i$. This vector is of length K, each element corresponding to one of the K classes. If observation i belongs to class k, the k:th element will be one in this vector, and all other will be zero. Using the discrepancies between this target vector and the predicted probabilities, the parameters $\boldsymbol{w}$ can be adjusted to improve the accuracy of the model. This is is the main idea behind fitting the ANN.

3.6.2 Fitting the ANN

Assume that there is an error function $R(\boldsymbol{w}) \equiv \sum_{i=1}^{n} R_i(\boldsymbol{w})$ that measures how much the predicted probabilities differ from the actual target vector, i.e. how well the model fits the data. Here, $\boldsymbol{w}$ can be seen as the vector containing all parameters w, often called *weights*, in the network. Note that n is the sample size, meaning an error can be computed for each individual observation. For R_i, the *cross-entropy* function

$$R_i(\boldsymbol{w}) = -\sum_{k=1}^{K} y_{ik} \log(p_{ik})$$

will be used in this thesis.

Furthermore, taking the derivative of the error function with regard to each weight w and evaluating it at the current weight value, predicted class probabilities $\boldsymbol{p}_i$, and target vector $\boldsymbol{y}_i$, yields the gradient ∇R_i. This specifies the slope of the error function with regard to the weights, and will be needed to fit the model.

Fitting the ANN then begins by randomly assigning values to all weights in $\boldsymbol{w}$, usually from a distribution producing values near zero. The first observation i from the training set is then fed into the model, producing a vector of predicted class probabilities $\boldsymbol{p}_i$. Since the target values are also known, the gradient of the error function can be computed. The weights can then be updated via *gradient descent* as follows:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \gamma \nabla R_i$$

where γ is the learning rate adjusting the size of each update. This procedure of updating the weights via gradient descent is then repeated for all observations in the training set. When all observations in the training set have been fed to the model once, one *epoch* is said to have passed. After a selected number of epochs, the process stops, and the model is considered to be fitted.

The way the gradient is actually computed is by a technique called *back- propagation.* Compared to forward propagation, back-propagation is computationally faster, and is therefore used here [6].

3.6.3 Hyper-parameters & model framework

Many choices need to be made when fitting an ANN. Some of these have already been presented in the previous sections, for instance using the sigmoid function for activation, or the back-propagation technique for computing the gradient. Due to limitations in time and in the software used [19], only certain selected hyper-parameters will be tuned via cross-validation (see section 3.3). These parameters and the values chosen for them will be presented in the results.

Finally, the features will be standardized before used, i.e. given mean zero and variance one. This is to avoid that features with higher absolute values are given greater importance by the regularization process [12]. This process entails adding a penalty term to the error function. By doing this, the weights are shrunken toward zero, which decreases the risk of overfitting.

3.7 k-Nearest-Neighbors

When compared to the other methods used in this thesis, the k-nearest-neighbors method (k-NN) stands out in terms of simplicity. The way it performs classification on new data is by calculating the *Euclidean distance* between the feature vector of a new observation and all observations in the training set. It then sorts the distances from lowest to highest, and assigns the given observation to the class most common among the top k entries on that list.

For the Euclidean distance to make sense as a measure of distance, the features must be measured in the same units. Therefore, all features are standardized to have mean zero and variance one [12].

The number of neighbors k to use for the classification decision will be decided via cross-validation as described in section 3.3, and is presented in the results. Values tested for k have been 1, 2, 3, 4, 5, 10, 20 and 50.

4 Results

In this section, the results are presented. These include performance measures for the different models, together with a test for whether the models differ significantly in regard to overall accuracy. Some descriptive statistics are also presented. It is with the help of the content in this section that the research questions will be answered. Additionally, the results will be the topic of discussion and interpretation in section 5.

4.1 Model performance

The main result of this thesis can be seen table 2 and figure 2. Table 2 of model performance on test data indicates that the *random forest* model actually beats the other models regardless of which performance measure is studied. As a matter of fact, there is no case in which a model outperforms another in terms of one measure, but is outperformed by the same in terms of another measure.

After random forest, the second best model is the *ANN*, which beats the *logistic regression* in all measures, in turn beating the *k-NN* in all measures.

	Accuracy	*Precision*	*F-score*	*MCC*
Random Forest	0.99125	0.991347	0.991259	0.988360
Logistic Regression	0.9750	0.975077	0.975005	0.966689
ANN	0.98000	0.980119	0.979973	0.973390
***k*-NN**	0.92625	0.926668	0.925845	0.902067

Table 2: Model performance measures (accuracy, precision, F-score and MCC) for the different machine learning methods used (random forest, logistic regression, artificial neural network and k-nearest-neighbors)

For the top three models, however, the race is fairly tight, and for the logistic regression and the ANN, the performances seem to be virtually indistinguishable. To test whether any of the differences in accuracy actually indicate a significant difference between the methods, a bootstrapping technique is employed. The result of this can be seen in figure 2.

What has been done here is that 50 bootstrapped training sets have been sampled from the original one. For each of these bootstrapped sets, models from all four methods have been fitted. Every fitted model has then been used to classify the test set, for which accuracy has been recorded. Out of the 50 accuracy estimates obtained for each method, a mean and standard deviation of the estimates has been computed. The standard deviation has then been used to create lower and upper bounds for the accuracy estimates via the margin of error, using $\approx 95\%$ confidence. These bounds are visualized as black error bars

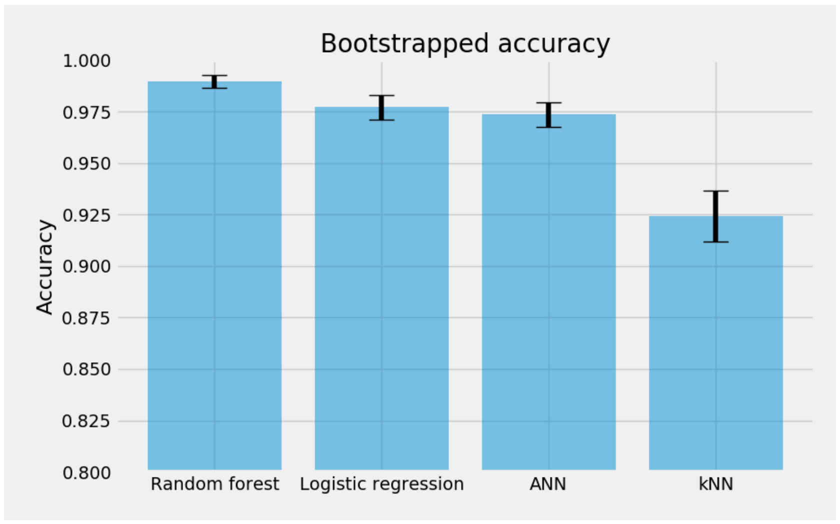

Figure 2: Accuracy for the four different models averaged over 50 bootstrapped training sets. Error bars indicating the upper and lower bound of an $\approx 95\%$ confidence interval. Note broken y-axis.

in figure 2, in which the mean accuracy for each method is represented by the blue bars.

It is clear from this plot that the error bars for logistic regression and ANN overlap, indicating that the methods indeed seem to perform equally well. If anything, the logistic regression even has a slightly higher accuracy than the ANN when looking at the bootstrapped sets. Furthermore, although it is hard to spot, none of the error bars overlaps with the error bar of the random forest. This would seem to indicate a significantly higher accuracy than for the other methods. In contrast, since the error bar of the k-NN does not overlap any of the others, this indicates that the method has significantly lower accuracy than the rest.

The hyper-parameters used for each method above have not been changed over the bootstrapped training sets. Instead, they have been selected via 5-fold cross-validation performed on the original training set, and then kept constant during the bootstrapping.

For the random forest method, the number of trees is set to 200 and the number of features used for each split is set to the square root of the total number of features, i.e. $\sqrt{120} \approx 11$ features. In the logistic regression, the tuning parameter for the regularization is chossen to be 1. Next, the ANN method finds *one hidden layer* with 50 neurons, a learning rate of 0.01, and 200 epochs, to yield the optimal model. Finally, cross-validation for k-NN results in using $k = 2$ neighbors. For the rest of the thesis, the models trained on the using the hyper-parameters stated above will be called the *final models*.

To see in greater detail how the final models classify the observations in the test set, table 3 presents the confusion matrices obtained for all these models. The rows show the actual number of observations in the corresponding class (200 for each one) and what they have been classified as. The columns instead show the number of observations that have been classified as belonging to a certain class, and what class those observations actually belonged to. This means that the diagonals show the number of observations for each account type that have been classified correctly.

Note that although the columns have not been labeled, they have the same order as the rows, so the first column is for observations classified as *genuine users*, the second for those classified as *social bots*, and so on.

Random forest

GU	199	0	1	0
SB	2	198	0	0
TB	3	1	196	0
FF	0	0	0	200

Logistic regression

GU	194	4	2	0
SB	2	197	1	0
TB	3	2	193	2
FF	0	1	3	196

ANN

GU	198	2	0	0
SB	2	198	0	0
TB	2	2	192	4
FF	0	2	2	196

k-NN

GU	191	5	2	2
SB	2	197	1	0
TB	13	4	175	8
FF	7	1	14	178

Table 3: Confusion matrices for the four final models. Actual class belongings are given in rows, predicted class belongings are given in columns. Rows and columns correspond, so for instance the first row (titled GU) shows number of actual genuine users, and first column shows number of predicted genuine users (GU: Genuine users, SB: Social bots, TB: Traditional bots, FF: Fake followers).

4.2 Examining the features

To get an at least somewhat better understanding of how the features might be utilized in the models, this section presents a figure of feature importance as obtained from the random forest. A table of means for some selected features is also presented to give an idea of how and in what regard the different account types might differ the most.

4.2.1 Which features are most relevant?

In figure 3, the 15 features with highest feature importance, as described in section 3.4.4, are displayed. The names of the features are the ones used by the program running all the computations. Except for using "md" as an abbreviation for "median" and "btw" for "between", there should not be many

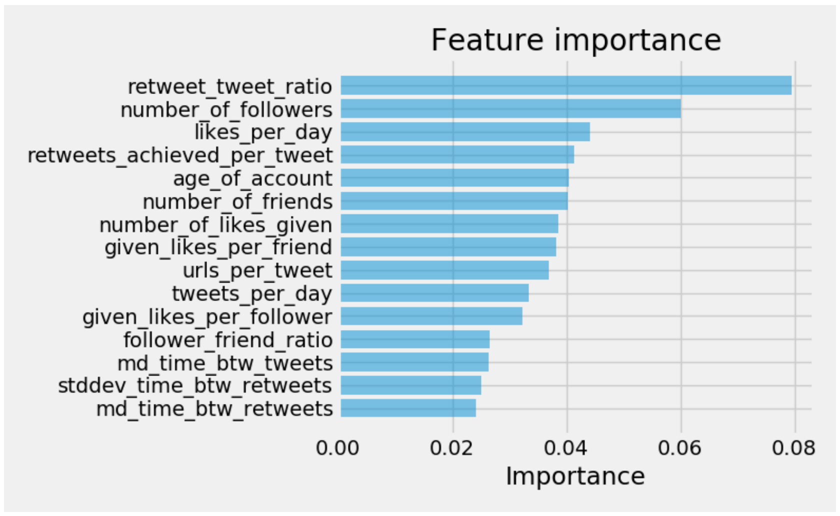

Figure 3: Feature importance for the 15 most important features as obtained by the random forest model. The bars represent the average reduction in Gini index achieved by each feature as a percentage of the total reduction in Gini index over all features.

differences between these names and what the features have been called elsewhere in the thesis.

The kinds of features that show highest importance vary quite a lot, with both account and content features reaching the highest spots (account features having a slight majority over all), and not any specific types of feature completely dominating the list. However, none of the features concerned with the text of the tweets, like *length of tweets, number of words* or *tweet endings*, show up on the list. Neither do the features dealing with hour of the day or weekday specific information. On the other hand, a couple of features targeting the time between different kind of tweets take up the bottom placements, and four features related to retweets enter the list at somewhat varying positions.

4.2.2 Class characteristics

It is not feasible to analyze every model in the level of detail required to learn what features have what effect on the classification. However, to at least get some picture of the roles played by the features in the models, table 4 presents the mean values for 20 selected features. Some have been selected because they where given high importance by the random forest model (see previous section), but in general the selection has been made so as to represent a wide variety of feature types.

Features	**GU**	**SB**	**TB**	**FF**
Retweet-tweet ratio	0.7804	0.4919	0.0517	31.188
Retweets achieved per tweet	728.40	3.66	0.0	403.45
Number of followers	2271.07	611.68	637.30	18.20
Number of friends	676.94	589.79	1326.54	363.30
Likes per day	4.8556	0.0658	0.0051	0.0087
Age of account	1419.64	296.19	412.22	320.93
Number of tweets	19951.97	1500.43	220.83	74.73
Tweets per day	15.7634	2.3061	0.4929	0.2637
Length of username	10.7959	12.3009	11.26	11.9069
Length of tweets (mean)	72.2173	72.0041	75.6161	62.0722
Length of tweets (std.dev)	34.8352	35.912	20.1683	34.3911
Number of words (mean)	12.0591	12.5193	13.3363	10.6049
Time between tweets (mean)	19.68	21.16	138.06	272.55
Tweet endings, "."	0.1638	0.1770	0.1186	0.2013
Tweet endings, "!"	0.0577	0.0546	0.0793	0.1232
Weekdays, Mondays (norm.)	0.1152	0.0827	0.1183	0.1407
Weekdays, Fridays (norm.)	0.1998	0.0996	0.1745	0.1571
Hours of day, 12 p.m. (norm.)	0.0287	0.0321	0.0337	0.0408
Hashtags per tweet	0.2986	0.0763	0.1558	0.1822
Unique hashtags per tweet	0.1798	0.0658	0.0617	0.1599

Table 4: Mean values for 20 selected features and all account types (GU: Genuine users, SB: Social bots, TB: Traditional bots, FF: Fake followers).

The top six features in table 4 are also the top six in figure 3. All these features show rather vast differences in mean values between the different classes. This is particularly notable for *retweets achieved per tweet*, where the values range between 0 and more than 700. In addition, there does not appear to exist any clear pattern of similarity between the classes. For instance, for *number of followers*, social bots and traditional bots are closest to each other, while for *number of friends*, social bots are more similar to genuine users.

Another feature where there exists a drastic difference between the account types is *tweets per day.* Here, the value is around 60 times higher for genuine users than for fake followers. Large differences between highest and lowest value also exist for features like *number of tweets* and *time between tweets (mean).*

In contrast, features like *length of username* and *number of words (mean)* do not show too much variation between the different classes, although these features are subject to certain limitations since usernames and tweets cannot exceed a certain length. Some interesting differences can be spotted in for instance *tweet endings, "!"*, which seems to indicate that fake followers use exclamation marks to end their tweets almost twice as often as any other class. However, this might not be enough to impact the models in any decisive way, as was seen for instance in figure 3 where this feature did not show up.

Some other features, like *length of tweets (mean)* and *length of tweets (std.dev)* are similar for all but one class, which diverges notably from the rest. These could then potentially be relevant features for distinguishing that particular class.

5 Discussion

The purpose of this section is mainly to bring up the most remarkable aspects of the results and analyze what they mean, both for the conclusions in this thesis and for future research in the field.

5.1 Interpreting results

Before any discussions on which method performed best, it is reasonable to point out that all final models achieved accuracy above 90% on the test set, which, given previous work in the field (see the works cited in section 1.3), is a relatively good result.

It is also worth pointing out that the precision, F-score and MCC do not contribute any relevant additional information about the models' performances. The idea behind these measures is to capture patterns in the classifications which the overall accuracy fails to capture. However, since both the precision, F-score and MCC are all so close to the accuracy, no new aspects of the models are revealed by inspecting these measures.

As has been shown in section 4.1, there seems to be a tied race between the *logistic regression* and the *ANN* methods. Although the ANN outperforms the logistic regression on the test set, the bootstrapping of accuracy indicates that this might as well just be due to the randomness inherent in the estimation. *Random forest*, on the other hand, is by using the same bootstrap approach deemed to significantly outperform all methods. Worst performance is attributed to the *k-NN* method, showing a significantly lower accuracy than all other models.

The random forest performing best would be very much in line with results from binary classification of Twitter accounts (as mentioned in section 1.2), where the random forest has also been shown to give best performance when compared to other machine learning methods.

One thing to note here is that only accuracy has been bootstrapped. The other performance measures might show other results, which could mean that for those measures, the random forest would not be significantly better than all other methods.

More importantly, however: since the differences between at least the top three methods are so strikingly small, it is probably more interesting to analyze which features are most important and what issues might exist with the data, than which particular model to use for the classification.

Looking at table 3, for all models, observations from the traditional bot class seem to be the most difficult to classify. This is counter intuitive, since the traditional bot class should encompass bots which are supposedly unsophisticated and thus easier to detect. Perhaps this then says something about the nature of the data set (as hinted at in section 2), namely that the sampling procedures used to obtain the different accounts likely has at least some unwanted impact on test accuracy. That is, the accounts do not only differ because of their different class belongings, but also purely because they have been sampled differently. Or in yet other words, there seems to exist some *sampling bias*. This means that some members of the intended population have not been sampled with the same probability as others. Since the members with higher sampling probability most likely also have characteristics distinguishing them from the rest of the population, this skews the data, making any conclusions drawn from it less dependable.

In light of the feature means shown in table 4, one could have thought that traditional bots would differ enough from the other classes for the models to pick it up. For multiple features, some of which are among the highest in importance in the random forest model, the means differ drastically from the other account types. Apparently, however, the models sometimes seem to mistake the traditional bots for other account types, and in particular surprisingly often for genuine users. This gives further support to the notion that sampling procedures have impacted the data to some extent, at least if we assume that what has been said in previous works about traditional bots being easier to classify is true. Now, the discrepancies between previous works and the results here could also be due to the fact that this thesis is trying to solve a multinomial problem. Furthermore, seeing that the differences are relatively small between most models, this is not something that should be taken to show that data is detrimentally skewed. It is merely indicating that the suspicion about some sampling bias being present in the data seems to have been accurate.

On the theme of sampling bias, another thing worth pointing out is that there are some suspicious values in table 4. For instance, the feature *tweets per day* shows that genuine users in the data set post more than 15 tweets per day on average. Not only is this multiple times higher than for any other account type, it also goes against what probably most people would view as normal human twitter behaviour. Common sense tells us that 15 tweets a day on average is probably not the most representative number for most regular (human) twitter users. Similarly, having more than 700 *retweets achieved per tweet* must be considered highly uncommon for most human users. Compared to the previous discussion on traditional bots, this is actually even more alarming. Users who show this much activity are not unlikely to differ from most genuine users in other ways too. It is hard to say exactly how serious this discrepancy in behaviour is, but it is surely not an optimal situation.

If doubts concerning the quality of data are suspended for a moment, assuming that the data sample is at least in most aspects representative of the wider population, one conclusion that could be drawn from the results is that the multinomial classification actually seems justified. For instance, it could have been the case that the models would not have been able to distinguish between the different bot types, or between genuine users and social bots. However, all models cope fairly well with this task, which gives at least some reason to believe that it could actually be a useful approach to classifying Twitter accounts. Seen somewhat optimistically then, this thesis would seem to suggest that using multiple bot types for Twitter account classification is not only sometimes theoretically preferable, but also in fact fully doable in practice.

5.2 Future research

There are a number of question marks regarding how the data has been sampled, with some results indicating sampling bias having a detectable impact on the classifications. In light of this, sizable parts of future research in the field ought to be devoted to collecting qualitative data. Without this, generalizations made from the sample about the wider population may never be fully reliable, and could potentially turn out to be misleading.

Developing unbiased and yet efficient sampling procedures for Twitter accounts is no easy task, however. Before this can be done properly, perhaps there even needs to be more research on the definitions and demarcations of actual bot classes. Not only would this serve to reduce bias and make the sampling more precise, it would also help increase the understanding of what actually constitutes a social bot, for instance.

For the more immediate development of the multinomial classification models produced in this thesis, the next step would be to try them out on completely new datasets. This could bring clarity to the actual impact sampling bias has had on the models, since performance measures notably different from those obtained here would indicate that the observations in test set have indeed not been representative of the full population. Of course, it might also be the case that the data in any other dataset is biased, and that performance measures would differ for that reason. However, using several other datasets to try the models should still give a hint about their validity.

Another thing to try that could possibly improve accuracy for new data would be to train models using only content features. It has been shown in figure 3 that many of the most important features, at least for the random forest model, seem to be account features. However, account features are more dependent on the time of sampling than content features, and may be less suitable for classification of new Twitter accounts. As an example, a new traditional bot account will have a very low account age, and most likely few tweets, friends, followers, and so on. Using these features might therefore nudge the classifier in the wrong direction. Content features, on the other hand, ought at least theoretically be more stable over time, since they are supposed to capture the actual *behaviour* of the accounts. This is something which should not change as much over time as certain meta stats are likely to do. Therefore, it might be a good idea to do similar tests as have been performed here, but excluding the account features and use only content features instead.

Finally, this selection of four machine learning methods is by no means exhaustive, and there is of course also reason to test other classifiers in future research. An example could be the unsupervised technique mentioned in section 1.3, which has been shown to work very well for binary classifications.

6 Conclusion

This thesis set out to answer how accurately different machine learning methods could classify Twitter accounts, and whether any of the methods would outperform the others. It has been shown that among *random forest, logistic regression, artificial neural networks* and *k-nearest-neighbors*, the *random forest* stands as the winner (by an ever so slight margin). It reached a test accuracy of above 99%, which is remarkably high by almost any standard.

It has however also been shown that the data likely suffers from sampling bias, making the above results somewhat less dependable. The extent of this bias is difficult to assess merely by inspecting the current dataset. Therefore, perhaps one of the most interesting conclusions of this work is that the models obtained here ought to be tested on several different datasets, preferably ones collected via new, improved sampling procedures, before their validity can be confirmed. Moreover, this thesis suggests that it is exactly the development of reliable and efficient sampling techniques that should be prioritized in the field of bot classification. Without notable increases of data quality, results like the ones obtained here will remain shrouded in uncertainty.

7 Acknowledgments

First and foremost, the author would like to thank Krzysztof Podgórski for supervising this thesis and for providing much valuable advice during the progress of this work. Krzysztofs insights and comments have greatly contributed to improving the quality of the thesis, and his guidance on how to organize the work has been paramount for its success.

The author also wishes to thank Stefano Cresci and the other authors of *The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race* [4]. They were kind enough to supply the dataset used in their paper, without which this work would not have been possible. Stefano has also been very helpful in answering questions regarding the dataset, for which the author is truly grateful.

Finally, the author owes a debt of gratitude to Johan Fernquist at the Swedish Defence Research Agency (Totalförsvarets forskningsinstitut, FOI). Johan and his colleagues shared the unpublished version of *Political bots and the Swedish general election* [8], which helped a lot in the writing of this thesis. In addition to that, Johan has on multiple occasions answered questions about their work and the data they have used. This thesis would not be what it is, were it not for these inputs.

References

[1] N. Carlson. The real history of twitter. `https://www.businessinsider.com/how-twitter-was-founded-2011-4?r=US&IR=T&IR=T`, 2011. Accessed: 2018-10-18.

[2] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia. Who is tweeting on twitter: human, bot, or cyborg? In *26th Annual Computer Security Applications Conference (ACSAC)*, pages 21–30, 2010.

[3] S. Cresci, R. Di Petro, M. Petrocchi, A. Spognardi, and M. Tesconi. DNA-inspired online behavioral modeling and its application to spambot detection. *IEEE Intelligent Systems*, 31:58–64, 2016.

[4] S. Cresci, R. Di Petro, M. Petrocchi, A. Spognardi, and M. Tesconi. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *26th International Conference on World Wide Web Companion*, 2017.

[5] J.P. Dickerson, V. Kagan, and V.S. Subrahmanian. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2014.

[6] G. Dreyfus. Modeling with neural networks: Principles and model design methodology. In G. Dreyfus, editor, *Neural Networks - Methodology and Applications*, chapter 2, pages 85–199. Springer, 2005.

[7] L. Fahrmeir, T. Kneib, S. Lang, and B. Marx. *Regression - Models, Methods and Applications*. Springer, Berlin, 2013.

[8] J. Fernquist, L. Kaati, and R. Schroeder. Political bots and the swedish general election. In *IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 124–129, 2018. DOI: 10.1109/ISI.2018.8587347.

[9] E. Ferrara. Disinformation and social bot operations in the run up to the 2017 french presidential election. *First Monday*, 22, 2017. DOI: 10.5210/fm.v22i8.8005.

[10] J. Gorodkin. Comparing two k-category assignments by a k-category correlation coefficient. *Computational Biology and Chemistry*, 28:367–374, 2004.

[11] R. Gorwa and D. Guilbeault. Unpacking the social media bot: A typology to guide research and policy. *Policy & Internet*, 2018. DOI:10.1002/poi3.184.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2nd edition, 2009.

[13] S. Hegelich and D. Janetzko. Are social bots on twitter political actors? empirical evidence from a ukrainian social botnet. In *Tenth International AAAI Conference on Web and Social Media (ICWSM)*, 2016.

[14] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning*. Springer, New York, 2013.

[15] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30:25–36, 2005.

[16] K. Lee, B.D. Eoff, and J. Caverlee. Seven months with the devils: A long-term study of content polluters on twitter. In *Fifth International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 185–192, 2011.

[17] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 631–636, 2006.

[18] P. Metaxas and E. Mustafaraj. Social media and the elections. *Science*, 338:472–3, 2012.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] J. Ratkiewicz, M. D. Conover, M. Meiss, B. Gonçalves, A. Flammini, and F. Menczer. Detecting and tracking political abuse in social media. 2011.

[21] C. Shao, G. Luca Ciampaglia, O. Varol, K. Yang, A. Flammini, and Filippo Menczer. The spread of low-credibility content by social bots. *Nature Communications*, 9(4787), 2018. DOI: 10.1038/s41467-018-06930-7.

[22] M. Singh, D. Bansal, and S. Sofat. Who is who on twitter–spammer, fake or compromised account? a tool to reveal true identity in real-time. *Cybernetics and Systems*, 49:1–25, 2018.

[23] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45:427–437, 2009.

[24] Statista. Number of monthly active twitter users worldwide from 1st quarter 2010 to 2nd quarter 2018 (in millions). `https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/`, 2018. Accessed: 2018-10-18.

[25] O. Varol, E. Ferrara, C.A. Davis, F. Menczer, and A. Flammini. Online human-bot interactions: Detection, estimation, and characterization. In *Eleventh International AAAI Conference on Web and Social Media (ICWSM)*, pages 280–289, 2017.

[26] A.H. Wang. Detecting spam bots in online social networking sites: A machine learning approach. In *24th IFIP Annual Conference on Data and Applications Security and Privacy*, pages 335–342, 2010.

[27] C. Yang, R. Harkreader, and G. Gu. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8:1280–1293, 2013.