UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS"
MSC PROGRAMME IN DATA SCIENCE

# Contemporary Machine Learning Methods For Twitter Bot Detection

by

Andreas Bouras

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

**Supervisor:** Nikolaos Kolokotronis
Professor

Athens, June 2023

Contemporary Machine Learning Methods For Twitter Bot Detection

Andreas Bouras

MSc. Thesis, MSc. Programme in Data Science

University of the Peloponnese & NCSR “Democritos”, June 2023

UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS"
MSC PROGRAMME IN DATA SCIENCE

# Contemporary Machine Learning Methods For Twitter Bot Detection

by

Andreas Bouras

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

**Supervisor:** Nikolaos Kolokotronis
Professor

Approved by the examination committee on June 21, 2023.

| (Signature) | (Signature) | (Signature) |
|---|---|---|
| ..... | ..... | ..... |
| Nikolaos Kolokotronis<br>Professor | Konstantinos Vasilakis<br>Professor | Charilaos Akasiadis<br>Researcher |

Athens, June 2023

UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS"
MSC PROGRAMME IN DATA SCIENCE

# Declaration of Authorship

(1) I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

(2) I confirm that this thesis presented for the degree of Master of Science in Data Science, has

   (i) been composed entirely by myself

   (ii) been solely the result of my own work

   (iii) not been submitted for any other degree or professional qualification

(3) I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or processional qualification except as specified.

(Signature)

. . . . . .

Andreas Bouras

Athens, June 2023

# Abstract

The aim of this thesis is to provide a comprehensive review of the current state of the art in detecting bots active on the social media platform of Twitter through the use of machine learning algorithms. Bots can be classified into a variety of categories and while some clearly identify themselves as automated accounts, many are posing as human users. This latter category is known to be used for a number of reasons like distorting online discourse and swaying political elections, manipulating stock markets, pushing conspiracy theories and spreading misinformation. Due to the nefarious purposes these bots are used for it is important to evaluate current capabilities in detecting them and identify ways in which they can be improved. For this purpose four recently published research papers are examined, in which numerous machine learning methods are used on a common dataset, with varying results in accuracy and efficiency. These methods are reimplemented, in order to confirm the original authors' findings, and in many cases enhanced by combining elements that proved effective in other research efforts. A part of the work examined attempts to utilize the sentiment features of posted tweets, while other focuses on account and tweet level metadata. The rest of them follow a language-agnostic approach or try to classify accounts from single observations. All efforts above score highly in performance metrics ($accuracy > 90\%$) and at least a couple of them achieve nearly perfect classification accuracy ($AUC > 99\%$). Methods that use sentiment features demonstrate the need for better feature engineering in order to extract more features while the rest highlight the importance of further research in sampling techniques and the social aspect of making bot detection systems public and open source.

# Table of Contents

# List of Tables

# List of Figures

# Glossary of Terms

**Accuracy** Accuracy computes how many times a model made a correct prediction across the entire dataset.

**AdaBoost (AB)** Belongs to the ensemble learning methods and sublimates the "Wisdom of the Crowds" principle: models that individually show poor performance can form a strong model when combined. It is basically a Boosting technique used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances..

**Area under curve (AUC)** AUC measures the entire two-dimensional area underneath the entire *Region of Convergence* curve from (0,0) to (1,1).

**Condensed Nearest Neighbors (C-NN)** Undersampling technique that randomly selects the samples within its k-nearest neighbors from the majority class that are to be removed.

**Convolutional Neural Network (CNN)** A Convolutional Neural Network is a class of artificial neural network most commonly applied to analyze visual imagery.

**Cresci 2015 dataset (C15)** Created by the Cresci team in 2015 to support a study on detecting Twitter fake followers. Used to test the adaptability of methods used in chapters 5 and 6, to new data..

**Cresci 2017 dataset (C17)** Twitter dataset created in 2017 by Cresci et al. The main dataset used in the papers examined by this study.

**Edited Nearest Neighbor (ENN)** An undersampling technique that removes samples from the majority class to match the minority class.

**F1 score** A machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model.

**Feed-forward Neural Network (FNN)** A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle.[1] As such, it is different from its descendant: recurrent neural networks.

**Global Vectors for Word Representation (GloVE)** An unsupervised learning algorithm developed by Stanford for generating word embeddings by aggregating global word-word co-occurrence matrix from a corpus..

**K-nearest neighbors (KNN)** IA non-parametric supervised learning method used for classification and regression. In both cases, the input consists of the k closest training examples in a data set.

**Logistic Regression (LG)** A classification algorithm. It is used to predict a binary outcome based on a set of independent variables. A data analysis technique that uses mathematics to find the relationships between two data factors. It then uses this relationship to predict the value of one of those factors based on the other.

**Long short-term memory (LSTM)** Long short-term memory is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections.

**Matthew's Correlation Coefficient (MCC)** A statistical tool used for model evaluation. Its job is to gauge or measure the difference between the predicted values and actual values and is equivalent to chi-square statistics for a 2 x 2 contingency table.

**Multi-layer Perceptron (MLP)** A multilayer perceptron (MLP) is a fully connected class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to mean any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation).

**Natural Language Processing (NLP)** Natural language processing is an interdisciplinary subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.

**Precision** Precision is the ratio between the True Positives and all the Positives.

**Random Forest (RF)** A popular machine learning algorithm that belongs to the supervised learning techniques. It can be used for both Classification and Regression problems and is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model..

**Recurrent Neural Network (RNN)** A recurrent neural network is a special type of artificial neural network adapted to work for time series data or data that involves sequences. Ordinary feedforward neural networks are only meant for data points that are independent of each other.

**Region of Convergence (ROC)** Region of Convergence is defined as the set of points in s-plane for which the Laplace transform of a function x(t) converges. It is the range of Re(s) (i.e.,$\sigma$) for which the function X(s) converges.

**Stochastic Gradient Descent (SGD)** A variant of the Gradient Descent algorithm used for optimizing machine learning models. In this variant, only one random training example is used to calculate the gradient and update the parameters at each iteration..

**Synthetic Minority Oversampling Technique (SMOTE)** An oversampling method of balancing class distribution in the dataset. It selects the minority examples that are close to the feature space. It then draws the line between the examples in the features space and draws a new sample at a point along that line..

**TextBlob** TextBlob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

**Tomek Links (TL)** An undersampling technique that is basically a modification of the Condensed Nearest Neighbors technique. Uses a properties rule to decide which of the nearest neighbors to delete.

# Chapter 1

# Introduction

In 2023, Twitter has more than 350 million monthly users. What initially was an SMS-based app for updating close friends and family has evolved to a dominant force in the global media landscape, now handling more than 500 million tweets daily [1]. And while Twitter's share in social media users globally might seem like a fraction of what other social media currently has (Facebook: 2.93 billion users, Instagram: 1.35 billion users, TikTok: 834 million users), Twitter is universally considered as the most influential among them and the public town square of the world. It is the platform of choice of 69% of U.S. based journalists and the podium for many political figures around the world [2].

As a free tool for real-time communication of massive scale, it been praised by researchers for democratizing discussions [3] and giving an outlet for people living in oppressed countries to raise awareness for social and political issues in their countries. It also considered a functional framework for disaster response, when people need easily accessible means to seek support, check on family and friends, gather information about the magnitude of a disaster and provide first-hand accounts of ground zero [4]. Case in point the earthquakes in Turkey and Syria in February 2023, where people called for help on social media, through their phones, from under the rubble [5].

Also notable is it's capacity to shape political discourse, influence the stock market, distract and direct the attention of millions of people. Following recent events such as

the 2016 US presidential election, or the still ongoing COVID-19 pandemic, many have expressed concerns about the effects of false stories ("fake news"), circulated largely through social media, including Twitter [6, 7]. However, this trend of misinformation during times of humanitarian crises propagating through social media platforms is not novel. Previous research has focused on the spread of misinformation and conspiracy theories on social media in the aftermath of the 2010 Haiti earthquake [8], the 2012 Sandy Hook Elementary School shooting [9], Hurricane Sandy in 2012 [10], the 2013 Boston Marathon bombings [11, 12], and the 2013 Ebola outbreak [13].

With the media coverage of events of global attention exploding, the proliferation of conflicting information quickly became evident. The engagement patterns of Twitter accounts posting unverified information or blatant misinformation were soon linked to automation mechanisms behind the content posted [14]. This aspect of Twitter posting activity even came under the spotlight by Elon Musk, who amidst an ongoing deal to purchase Twitter for $44 billion claimed that 33% of 'visible accounts' are 'false or spam accounts', statement that grossly contradicted Twitter's affirmation that bots comprise less than 5% of its active users [15].

## 1.1 Motivation

While it is already established that bot presence in social media is a problem, examining the scale of it can help realize just how serious of a problem it really is and whether actions should be taken on a private or government level.

Studies, done as far back as 2015, have shown than nearly 48 million of Twitter's user-base could be bots [16]. But even if Twitter's optimistic claim that bots are less than 5% of its users (roughly 17 million users) is true, that is not indicative of the reach an automated tweet can get. In 2021, Facebook stated that it blocked 4.5 billion accounts in the first nine months of the year, while in a similar way TikTok reported that they have removed 53 million fake accounts starting January through June 2022 [17, 18]. Content often spreads from one social media to the other, with

tweets being posted on Facebook, short TikTok videos posted on Instagram, Reddit threads being linked on Twitter, the reach one social media post can get can never be accurately estimated. With the latest metrics showing that more than half of the world population (59%) is on some form of social media with 4.76 billion users spending an average of 2h 31m daily [19] scrolling through their timelines, it is evident that social media, including Twitter, is a vast attack surface for malicious bots.

The bots' connection to online misinformation has caused concern among Americans, especially in the aftermath of the 2016 U.S. presidential election. Congressional hearings held in 2018 [20] along with investigations and research done by academic researchers and social media [21, 22] have suggested that social media bots are a major factor in spreading misinformation. The topic has drawn the attention of much of the public, with a survey done in 2018 showing that 8 in 10 of the US citizens that have heard of the phenomenon believe they are used maliciously. Also notable: 34% of Americans have never heard about bots [23]. For clarity in this part of the subject, R. Gorwa and D. Guilbeault have formulated a typology for the various kinds of automated, or seemingly automated, software commonly referred to as bots [24]. Present below is a part of that typology, most closely related to the bot categories referenced in this thesis and present in the used dataset.

- Crawlers and scrapers: bots used to index and archive websites
- Chatbots: used to engage in natural dialog with physical persons
- Spambots: used to post messages, URLs, images etc. en masse
- Social bots: used to generated content on social media, often posing as real humans
- Sockpuppets: accounts used by real humans with false provided credentials
- Cyborgs: accounts used by both real humans and by automation mechanisms

While Twitter is the digital town square and an accessible place for voicing an opinion, it is also a huge advertising platform. In 2021, $4.51 billion out of $5.08 of Twitter's revenue came from advertising [25]. And while diversifying Twitter's revenue stream to make the platform less reliant on advertising is a standing goal, analysts and industry experts claim that it will be difficult to grow subscription revenue in the manner outlined by current Twitter management [26]. Bots are the main tool with which ad fraud is conducted, with yearly costs estimated in the billions [23]. With advertisers looking to deepen their engagement with humans and marketers that rely too heavily on ad impressions as a measure of success, too many fake accounts could translate to decreased monetization and loss of revenue [27].

Bots can have a hurtful impact on the public discourse, the platform they operate on, and the advertisers that depend on the platform. But they can also negatively affect the platform's user-base directly in various ways: site slowdowns, comment spam, rogue reviews, skewed analytics, promotion exploitation. But they can also affect users in a way that can spill over to the rest of the user's activity, with ways that fall on the spectrum of cybersecurity. Bots can post links to malicious websites and applications and lead to loss of data, infected systems, hacked bank accounts or worse.

This whole range of problems caused by malicious bots creates the need for a comprehensive review of the current state of the art in bot detection, and the ways already available methods can be pushed further to produce more accurate results with improved efficiency.

## 1.2 Thesis Objectives

The purpose of this thesis is to provide an understanding of current methods used in detecting bots on Twitter by presenting the approach used in four recently published research papers (least recent one published in 2018), compare their performance and underline the room for improvement and the opportunities for future work. The

frame of reference for these papers is that all of them employ tools based on machine learning and their feature set is both content-based and account-based. They also use the same dataset, the one created in 2017 by Stefano Cresci and the rest of the authors of the "The paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race" research paper [28]. More details about the dataset are given in the data chapter below. The focus is on explaining the *purpose* of each study, examining and simulating the methods used, interpreting the results and summarizing the strengths and limitations of each approach.

The code used and most technical details of the methods used in these papers were not publicly available. For this reason and in the scope of this thesis, the machine learning algorithms used by the researchers was retraced and reproduced taking into account the limited information available, to gain a better understanding of the methods applied and confirm the claims on performance made by the authors.

## 1.3 Thesis Outline

This thesis is organized in a text-to-text way. In chapter 2, related work is discussed along with the datasets used across all research papers selected. Then the four papers are discussed on separate chapters with a short introduction followed by the methods used and technical details, the results in terms of the metrics used and a conclusions section.

A point-by-point comparison between the papers was avoided in order to escape the illusion of a ping-pong game and focus on the heart of the argument in the most readable way possible.

# Chapter 2

# Preliminaries

## 2.1 Related Work

Most of the work done on Twitter bot classification approaches the issue as a *binary* classification problem, meaning it focuses on telling bots apart from real users, without going into specifics about the type of bots that were recognized. This does not mean that there is a total absence of multinomial classification approaches in the field. The work done by Jan Novotny[29], closely examined later in this project, sets out to identify three different categories of bots among real users, social bots, traditional bots and fake followers. Another research team , Gianvecchio et al.[30], worked on classifying accounts as bots, humans, or sth in-between: cyborgs. C.A. Davis[31] and his team worked on distinguishing different types of bot accounts without technically using multinomial classification, but rather employing binary classification methods and K-means clustering to identify bot and real user subcategories. Kagan et al. were one of the first teams to work on sentiment extraction and analysis with the purpose of Twitter bot detection, followed by Heidari and Knauth's teams, whose work is also examined in this project [32, 33]. Most other approaches focus on recognizing automated behavior through identifying rigid posting habits and time patterns or optimizing already tried methods. Optimization that's possible through refining the set of features used, creating leaner and easier to understand models that can even identify a bot by a single data point.

What these research efforts have in common is that they all use the dataset described in section 2.2.1 and that they all involve both account-based features and content-based features. This means that they take into account available information for both the account's configuration and the tweets that were posted from it. This information is the basis on which extra features are produced, which are in turn used to train the classification models. Overall, for the papers examined, the broad categories for the rationale on which they were based are the following:

1. *Sentiment extraction*: identifying sentiment expressed by tweets on an account basis by processing tweet text

2. *Language-agnostic bot detection*: sentiment extraction by processing expressive means in tweets other than text, like emojis

3. *Bot category classification*: identifying time patterns and posting habits by analyzing tweet-metadata

4. *Model optimization and feature minimization*: a combination of some of the above methods with emphasis on creating lean and easily interpretable models

These categories are obviously not exhaustive, and the possibility of an overlap between groups is very real, however, they were considered a good starting point for the scope of this thesis. The paper discussed in chapter 3 is representative of the sentiment extraction category. Chapter 5 is more closely aligned to the third category, in that many features produced in that paper are about the time-related aspects of tweet posting and classifying bots in a multinomial way to different categories . Chapter 4 is another attempt at sentiment extraction, with emphasis on keeping the procedure language-agnostic and with universal applicability. The work described in chapter 6 is about creating simpler and easier to understand models, with basic off-the-shelf machine learning aspects that can offer great performance with a modest

amount of features, that can go as far as identifying a bot by examining a single tweet.

## 2.2 Datasets

### 2.2.1 Cresci 2017

The dataset used across all featured research papers was created by the research team of Stefano Cresci in the University of Pisa, Italy [28]. It was part of the 2017 research effort that led to the paper "The paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race". It is commonly used in research efforts because it is one of the few manually annotated datasets publicly available. It is noteworthy that for the sake of completeness the authors actually paid for fake follower services. Fake followers are simple accounts that can inflate the number of followers of another account. The data is organized into nine folders, containing spreadsheets for four account categories. Each folder contains two spreadsheets: one with Twitter account data, the other with tweet content per account. On the account data spreadsheet, each row contains information about an individual Twitter account and there are no duplicates. In more detail, when it comes to account data there are nine account spreadsheets: one for real users, one for fake followers, three for social bots and four for traditional bots. For cases where there are multiple spreadsheets for the same category, that is due to the fact that various sampling processes were utilized, so each process produced it's own data file. There is no information about tweet content in this spreadsheet - a complete list of account features can be seen in the last column of the 2.2 table. In total, there are four different data labels in the data: **genuine users**, **social bots**, **traditional bots**, **fake followers**.

The data files that are about tweet content are organized in a different way. In each tweet file, every row represents a unique tweet, and each column represents a piece of metadata regarding that tweet. In these columns there is also the Twitter account ID

of the account that posted that row's tweet - it's how the account and tweet datasets are connected. Each account can have several tweets linked to it, which is why in almost all cases in the presented papers the data is aggregated before processing. The dataset will, henceforth, be referred to as C17.

### 2.2.2 Cresci 2015

As an expansion to evaluating the methods used in these research papers, a second dataset was employed on the same models trained and used on the original dataset. The goal was to verify whether the tools and procedures used with C17 can be applied to other datasets with similar results. The dataset, henceforth called C15, was created in 2015 by the same research team that created C17 [34]. The dataset is organized into five folders, each one containing four different spreadsheets: *followers.csv, friends.csv, tweets.csv, users.csv*. The tweets and users spreadsheets are straightforward to describe, they contain account metadata and tweet information, and are linked through the user identifiers present in both files. Followers and friends spreadsheets are basically mapping tables that connect each account to its followers and to its friends (the accounts it follows). The organization to five folders has to do with the source of the data and the project they are products of, specifically *'The Fake Project - TFP'* and *'#elezioni2013 - E13'* for real user data and *'fastfollowerz - FSF', 'intertwitter - INT', 'twittertechnology - TWT'* for fake follower data.

## 2.3 Data labels - C17

While the C15 dataset was originally used in an effort to distinguish between real users and fake followers, the C17 was used to identify fake followers but also social and traditional bots. Due to them being so close in their intended usage, and also due to the fact they were both produced by the same research team, there is a huge overlap between the features of the two datasets. In this section the focus is on C17, since it is the latest dataset of the two and is also, in principle, a superset of

| | | | relationships | | |
|---|---|---|---|---|---|
| **dataset** | **accounts** | **tweets** | followers | friends | total |
| The Fake Project (TFP) | 469 | 563,693 | 258,494 | 241,710 | 500,204 |
| #elezioni2013 (E13) | 1481 | 2,068,037 | 1,526,944 | 667,225 | 2,194,169 |
| fastfollowerz (FSF) | 1169 | 22,910 | 11,893 | 253,026 | 264,919 |
| intertwitter (INT) | 1337 | 58,925 | 23,173 | 517,485 | 540,658 |
| twittertechnology (TWT) | 845 | 114,192 | 28,588 | 729,839 | 758,427 |
| real humans dataset (HUM) | 1950 | 2,631,730 | 1,785,438 | 908,935 | 2,694,373 |
| fake accounts dataset (FAK) | 1950 | 118,327 | 34,553 | 879,580 | 914,133 |
| HUM ∪ FAK | 3900 | 2,750,057 | 1,819,991 | 1,788,515 | 3,608,506 |

Table 2.1: Cresci 2015: data overview

C15. Theoretically, social and traditional bots along with genuine users fall into the macro-category of fake followers.

### 2.3.1 Generalizing from available data

A standard goal when designing and implementing a machine learning approach to solving a a classification problem, binary or multinomial, is to make sure the methods utilized can properly adapt to new, unforeseen data based on the same distribution as the one used when creating the models. The adaptability of the models is founded on two parameters: the classes considered in the study **exist** and are actual and disjointed categories in the real world, and the classes are **accurately represented** in the dataset used for training the models. These two parameters translate to a need for *realistic definition* of bot classes and a *proper sampling procedure* applied in the data gathering effort.

### 2.3.2 Bot categories

In the C17 dataset is is assumed that there are **three bot categories**, along with a category that represents real users. These assumed categories do not correspond directly to the bot categories listed in paragraph 1.1 and suggested by Gorwa and

Guilbeault [24], but they are to be considered as more generic supersets, within which some of those categories fall in.

The defining lines that make these bot categories separate but equal are each category's **behavior** and **purpose**. For example the purpose of a traditional bot is to regularly post links to software, websites and sometimes even products. This purpose is mostly supported through frequency of posting. Social bots on the other hand, utilize their ability to act in a more social manner, often instilling sentimentality into their posts with the goal of promoting products and influencing public opinion on a variety of topics. These are distinct behavioral patterns that support disparate purposes. For fake followers it is mainly the difference in goals that justifies a distinct category, not so much the ways in which they interact with other users - though it is to be expected that there are differences in behavior as well. Fake followers are characteristically more passive in the ways they engage than all other bot categories. This is supported by the findings in the *'Retweet-tweet ratio'* feature used in this project, where fake followers are shown to be retweeting much more often compared to all other bots and even genuine users (FF: 31.188, SB: 0.4919, TB: 0.0517, GU: 0.7804). Other than tweeting scarcely however, there is nothing that prevents a fake follower account from being used in any other way a bot or a real account can be used, and that can be a considered a flaw of the class definition.

The most well defined class is undoubtedly that of *genuine users*. It is the category where accounts predominantly tweet manually and it's also a 'catch-all' category to which all non-bot accounts belong in. While defining this category is of minimum complexity, there's still the question of how much automation is tolerated in an account before it is considered a bot.

It is obvious that an account belong to one bot category does not necessarily mean it will not act in ways that can be attributed to other types of automated accounts. For example a social bot can be used to promote a product or inflate one's following. In order to cover bot **category overlap** it might be useful, in future works, to

examine the possibility of multi-label classification, or adding 'super' categories that condense more than one categories in them. This prospect is further discussed in section 7.2.

**Genuine users**

The data for genuine Twitter users was collected by the authors of [28], by selecting thousands of accounts randomly and asking each one of them a question in natural language. Of the accounts that received that question, only **3747** responded, and are the ones that are included in the dataset. Of these accounts that are verified as genuine, only **1083** have available tweet data.

**Social bots**

This type of bots refers to a relatively new and sophisticated kind of bot that poses as a real person. This part of the dataset contains 6 files in total, 3 pairs of accounts-tweets files. The first pair of accounts and tweets is about content posted during the 2014 mayoral election in Rome. It lists **991** Twitter accounts that were used as amplifiers that retweeted every tweet a specific candidate posted. The second pair is about **3457** accounts that promoted a smartphone application for finding and hiring workers specialized in digital arts. The third pair of files contains content of **464** accounts that promoted products on reduced price on Amazon.

**Traditional bots**

Traditional bots are mainly used for spamming links to websites and applications for a range of purposes, like increasing a website's traffic or facilitating the spread of malicious software. For the traditional spambots category the dataset included four spreadsheet files with Twitter user data, but only one with tweet data. Therefore there is only one pair of accounts-tweets that is usable and it contains **1000** Twitter accounts. For the papers examined, only the 2nd paper in line of presentation [29] explicitly comments on the outline of the dataset and the part of the data the authors used.

**Fake followers**

Fake followers are accounts mainly used for inflating one's follower count. Access to such accounts is a publicly offered paid service, which is how the authors of [28] created this part of the dataset. **3351** such accounts are included in the dataset, of which only **3202** are paired with tweet data and therefore usable.

The merged dataset that includes account-level and tweet-level information has a total of 67 features. Figure 2.1 below demonstrates how the information is merged between the accounts and the tweets subsets.

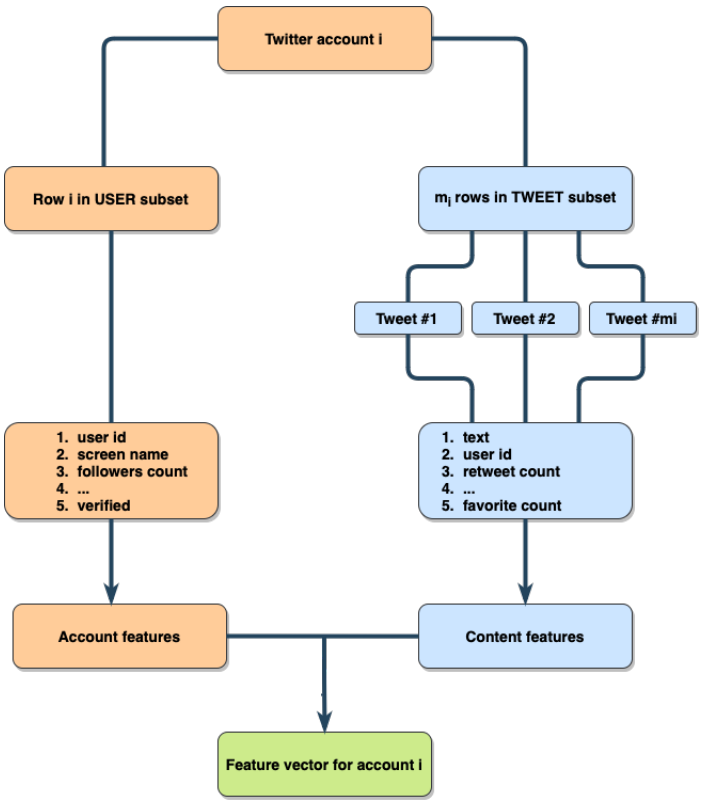


Figure 2.1: Cresci 2017 outline flowchart

### 2.3.3 Original features

The C17 dataset contains features descriptive of both the accounts involved and the tweets they posted. The account-based features are derived from account metadata and include the accounts screen name, its username, its number of followers and

friends, and others, for a total of 42 features. A friend in Twitter terminology is the account that is being followed by someone else. As an example if an account **A** follows an account **B**, A is a follower to B while B is a friend to A. The tweet, or content, based features number a total of 25 attributes and include the body of the tweet, the date posted, the number of likes(favorites) received as well as how many replies it has and the number of times it got retweeted.

All four of the papers examined use some of the features readily available in the C17 dataset and build on the available data to produce more attributes to be used in the classification effort. Computed features are about the sentiment in tweets, patterns in posting habits, the use of alternative symbols for emotional communication. Table 2.2 showcases the original set of features in the dataset used. For each research paper the original features utilized, along with the computed features and the final feature selection are different and are explained in more detail on each paper's designated section.

### 2.3.4 Sampling Procedure

This section basically aims to report on the characteristics of the sampling procedures followed and underline the potentially negative effect these could have in classification performance and generalizing to new data or entirely different datasets. The different categories of Twitter accounts present in the dataset were discussed in section 2.2.1. The degree to which the category subsets in C17 represented their supposed Twitter user categories is still pretty open-ended.

As was previously mentioned in paragraph 2.3.2, **genuine users** were randomly sampled and were then asked a question in natural language. How the answers were evaluated is not perfectly clear but it is assumed that the authors of [28] and creators of the dataset did a manual examination of the answers and also took into consideration other features of the accounts' metadata. Despite the lack of total transparency as to how exactly the evaluation was done it is pretty reasonable to assume that a

| No. | Content-level | Account-level |
|---|---|---|
| 1 | id | id |
| 2 | text | name |
| 3 | source | screen_name |
| 4 | user_id | statuses_count |
| 5 | truncated | followers_count |
| 6 | in_reply_to_status_id | friends_count |
| 7 | in_reply_to_user_id | favourites_count |
| 8 | in_reply_to_screen_name | listed_count |
| 9 | retweeted_status_id | url |
| 10 | geo | lang |
| 11 | place | time_zone |
| 12 | contributors | location |
| 13 | retweet_count | default_profile |
| 14 | reply_count | default_profile_image |
| 15 | favorite_count | geo_enabled |
| 16 | favorited | profile_image_url |
| 17 | retweeted | profile_banner_url |
| 18 | possibly_sensitive | profile_use_background_image |
| 19 | num_hashtags | profile_background_image_url_https |
| 20 | num_urls | profile_text_color |
| 21 | num_mentions | profile_image_url_https |
| 22 | created_at | profile_sidebar_border_color |
| 23 | timestamp | profile_background_tile |
| 24 | crawled_at | profile_sidebar_fill_color |
| 25 | updated | profile_background_image_url |
| 26 | - | profile_background_color |
| 27 | - | profile_link_color |
| 28 | - | utc_offset |
| 29 | - | is_translator |
| 30 | - | follow_request_sent |
| 31 | - | protected |
| 32 | - | verified |
| 33 | - | notifications |
| 34 | - | description |
| 35 | - | contributors_enabled |
| 36 | - | following |
| 37 | - | created_at |
| 38 | - | timestamp |
| 39 | - | crawled_at |
| 40 | - | updated |
| 41 | - | test_set_1 |
| 42 | - | test_set_2 |

Table 2.2: Content-level and account-level features in C17

non-genuine Twitter account would have difficulty slipping through both steps undetected. It is therefore safe to consider this subset as representative of real Twitter

users and the sample relatively unbiased due to the random sampling that was done.

The subset that contains **traditional bot** accounts was sampled as part of the work of C. Yang, R. Harkreader and G. Gu in their work on Twitter spamming [35], which was provided by request to the authors of the C17 dataset. About 500.000 accounts and over 14 million tweets are sampled using graph sampling. The content of the tweets was parsed by special software to identify URLs to malicious websites. The accounts that had a large concentration of such URLs posted were then manually examined by the authors to decide on whether they are bots or not. Due to the element of randomness in graph sampling the initial sample can be safely considered to be largely unbiased. The second step, however, of manually examining suspicious accounts depends on discretionary measures that can potentially befoul or skew the original sample. For example the threshold of 10% was selected by the authors to signify a potential bot: if more than 10% of an account's tweets included a URL to a nefarious website or app then the account is flagged as suspicious and is kept for manual examination. That would obviously exclude accounts that do not go past the threshold but could still be bots. Similarly, real accounts that do go above the threshold will be misflagged as potential bots. This parameter alone can be a potential source of bias, as is the manual examination process. Due to these aspects of the filtering process is is not unlikely that there are accounts that were falsely classified, though it is assumed that if such a problem exists, it is not extensive.

The **social bots** subset was created through a mixed effort of three different sampling processes by the authors of [28], though no specifics are given as to how these processes work. The first sampling process focused on the activities of a, back then, novel type of social bots that were observed during the 2014 Mayoral election in Rome. The second process was about accounts continually posting tweets with the #TALNTS hashtag that refers to a niche hiring app about artists. The third sampling process is about a group of bots that were observed advertising product listings on the Amazon platform. All these bot groups would mix political and promotional

tweets with posts that closely resemble genuine users' content like songs and YouTube videos. Due to the relative obscurity of the internal works of these process it is not possible to assess potential bias in this subset. Similarly to the subset of genuine users, closer examination of suspicious accounts was done manually. This examination, apparently, involves comparing tweet content and account metadata between accounts to identify the ones that have similar behavioral patterns and excluding standouts. Despite the overall vagueness in the sampling process it is not unreasonable to link the patterns of online activity observed by the authors to automated Twitter accounts and consider the subset to really consist of social bots.

**Fake followers**' sampling was done in a very simple way, as the authors bought access to such accounts by paying three different online marketplaces for the service. Paying for Twitter accounts and then using them to boost one's following is as close to the definition of fake followers as it gets, so there is no question as to whether the subset contains the claimed type of accounts. The question of sampling bias and whether fake followers from other providers will exhibit different characteristics is also sufficiently answered, since this subset is sourced from three different providers, it is safe to assume that other similar services will not be significantly different in the properties of the accounts they supply.

### 2.3.5 Class balancing

Many real-world classification cases exhibit some degree of class imbalance, which is what happens when each class does not comprise an equal amount of a dataset. C17 is no exception, where for genuine users, social bots and fake followers the samples exceed 3.000, for traditional bots the dataset only has 1.000 entries. And while this difference does not constitute a gross imbalance, it can still affect classification performance and model flexibility, as the classifiers will likely be more biased in favor of the more common classes [36].

This imbalance is not linked to real circumstances and profusion of social bots and

fake followers over traditional bots. It's most likely connected to the different ways the various parts of the dataset were sampled. To remedy that, and only in papers 2 and 4, the classes are balanced out by the means of *undersampling*. Samples are removed from the categories with a surplus, until all of them have the same number of observations. That inevitably entails the loss of, potentially useful, information but 1.000 samples per class is still considered a healthy amount of data that can support the classification methods used. This means that in the cases where undersampling is used, the dataset contains 4.000 tuples.

## 2.4 Data labels - C15

C15 is a combination of datasets created as byproducts of various research initiatives that will be briefly described below. The paper's primary goal was to investigate the, back then novel, issue of fake Twitter followers, and while the final dataset is much smaller in scale, during the whole research effort of the authors over 9 million Twitter accounts were crawled along with almost 3 million tweets.

### 2.4.1 Generalizing from available data

The principles behind training models that will be able to perform on new data are the same as described in paragraph 2.3.1, essentially correct class definition and sampling procedures. The same applies in the C15 dataset, for which the classes defined and sampling procedures are described below.

### 2.4.2 Defining classes

As was previously stated, the work that produced the C15 dataset's main focus was to automate detection of Twitter accounts that are created for the sole purpose of inflating the follower's count of a targeted account. This, effectively, boils down the classes used to the main two, **humans** and **fake** followers.

### 2.4.3 Original features

Understandably the type of effect that fake followers provide can also be provided by other types of automated accounts, like bots, spammers and even real user accounts. This means that all these types of accounts potentially fall into the macro-category of fake followers, and that the features that are already proven to be effective in detecting them could perform well in detecting fake followers as well. The original features are almost identical to the ones present in C17 (see 2.2), with only a few missing in content-level and account-level features. The missing features in C15 that are present in C17 are listed below:

- *Content-level*: contributors, favorited, retweeted, possibly_sensitive, crawled_at, updated
- *Account-level*: is_translator, follow_request_sent, notifications, contributors_enabled, following, timestamp, crawled_at, test_set_1, test_set_2

None of the missing features, however, are utilized in the examined four papers, therefore these are only listed for the sake of completeness.

### 2.4.4 Sampling Procedure

The datasets that were combined to create C15 along with the sampling approaches that were used are briefly descibed in this section below.

- *The Fake Project*: A project that started its activities on December 2012. It was based on a Twitter account with the handle @TheFakeProject and a profile description that read "Follow me only if you are NOT a fake". The account was created by researchers at the Institute of Informatics and Telematics of Italy's National Research Council (IIT-CNR) and the idea was to create an account that would only be followed by real Twitter followers. Within 12 days of the account's creation the followers count reached 574 followers. The followers'

accounts were crawled, which led to the collection of 616,193 tweets and 971,649 linked accounts (relationships). There was also a verification phase where these 574 accounts were sent a private message with a URL to a unique CAPTCHA. **469** of these accounts completed the CAPTCHA and were classified as "certified humans" and are included in the C15 dataset. The account is still present in 2023 with a follower count of 398 and appears to be inactive.

- *#elezioni2013*: This dataset was produced in the context of a sociological study done by the University of Perugia and the Sapienza University of Rome that looked into the evolution in the Italian political discourse during a 3-year period in 2013-2015. More than 84,000 Twitter accounts that used the hashtag #elezioni2013 were examined in the span of two months leading up to the 2013 general elections in Italy. Of these accounts, more than half were classified as politicians, candidates, parties, bloggers, journalists and were excluded. The classification was done by keyword-driven queries on the username and the profile description of the accounts. The rest of the accounts were classified as *citizens* (about 40,000). This subset of 'citizen' accounts was sampled and narrowed down to a set of 1488 accounts, which were then manually verified by sociologists from the University of Perugia to ascertain the nature of their profiles and tweets. The verification process was extensive and included the analysis of profile pictures, profile descriptions and a careful examination of the accounts' timelines. URLs in profile descriptions were manually checked for validity while accounts that had no profile picture or description were dropped from the dataset. The verification process took up almost two months and resulted in a dataset of **1481** Twitter accounts.

  The above two subsets result in a total of **1950** ($= 469 + 1481$) Twitter accounts that are verified to belong to *real humans*. It is worth noting that the accounts in the 'The Fake Project' participated in the project on a voluntary basis,

come from Europe and the US and are mainly researchers, social media experts and journalists. The accounts in the #elezioni2013 dataset are mostly Italian users that come from a variety of professional backgrounds and social standings, are not politicians/journalists/bloggers/political parties and they all share an interest in politics.

- *fastfollowerz*, *intertwitter*, *twittertechnology*: These three subsets contain 3000 fake follower account from three different online marketplaces. 1000 fake accounts were bought by the research team from http://fastfollowerz.com for $19. Another 1000 accounts were bought from http://intertwitter.com for $14. Both marketplaces provided more accounts than originally paid for, respectively 1169 accounts from fastfollowerz and 1337 from intertwitter. The last millenary was bought from http://twittertechnology.com for $13, with only 845 of the accounts ending up in the dataset as 155 of them were almost immediately suspended by Twitter.

  The above subsets result in a pool of **3351** (=1169 + 1337 + 845) fake Twitter accounts. While this fake followers dataset is not exhaustive, it is the product of using services publicly available and easily searchable on commonly used search engines, therefore representative of what can easily be done by an average user. This set of 3351 accounts was undersampled down to 1950, to match the size of the 'real humans' subset and lead to a dataset of balanced distribution of fake followers and humans. An overview of the subsets that went into the final Cresci 2015 dataset can be seen on the 2.1 table.

# Chapter 3

# Sentiment Extraction

This paper's approach is based on M. Workman's 2018 study on social media exchanges [37], specifically on the aspects of **confirmation bias** and the **backfire effect**. The backfire effect is what happens when one side of an exchange provides many arguments to support their opinion and change the other side's mind but, instead, succeeds in making the second side entrench itself in their original position and show a stubborn resistance to change [38]. As an example when a Twitter user regularly receives a lot of tweets that contrast his personal opinion, that person will usually become defensive and decide to not move away from their original belief - regardless of how solid the opposing arguments are. Thus the attempt to convince the user will 'backfire' and the reverse effect is achieved. Confirmation bias is when people tend to process information by interpreting, or looking for and handpicking parts of information that are consistent with their beliefs and theories. On Twitter, as in all social media, effects like confirmation bias and the backfire effect can lead to what is commonly referred to as 'echo chambers' [39]. An echo chamber is an environment or a digital space where a person only encounters opinions and information that are aligned with their personal values and reinforce their opinions. These spaces allow for circulation of existing views without opposition, which cordons constructive discourse off and circles around to more confirmation bias. Echo chambers can dramatically increase social and political polarization and extremism. Bots can pick up

on these effects and use them to create fake trends and spread misinformation, which can then be exploited commercially by driving traffic to web news outlets or by doing product placements. The effectiveness of these bots is largely based on the emotional reaction their content has on real users, which is why they usually try to infuse strong emotions in their posts, like anger, bitterness, resentment and desperation.

In order to identify these efforts from bots to cause emotional reactions, the authors base their approach on the belief that a **very biased** opinion can be identified by the sentimental 'symmetry' of an account's tweets. In other words, if the tweets posted by an account show a weighty concentration of positive, negative, or neutral statements, this could be linked to confirmation bias and then traced back to characteristics usually attributed to accounts with automated behavior [37]. It is to be expected that a human user would voice opinion of varying sentiments, like in example one could be happy with a play they watched but be dissatisfied with a recent purchase. With bots, the variety in the number of positive/negative posts is characteristically low [37]. To establish the idea that sentiment is a discriminating feature for identifying bot accounts, the authors perform a quantitative analysis that proves that the variation of the mentality between different tweets posted per account is drastically different. The analysis is done on the C17 dataset previously described.

## 3.1 Feature selection

The authors follow a forked approach where they test a selection of machine learning methods on a subset of the original features and then repeat the process with the newly created sentiment features added to the subset. The subset of original features is comprised of *followers_count, friends_count, retweet_count, reply_count, num_hashtags, num_urls*. The enhanced subset also includes *CountNeutral, CountPositive, CountNegative, SumPositive, SumNegative, AvgPositive, AvgNegative*. This parallel approach is done to determine the effect of the new sentiment features on the machine learning models used. Details of data cleaning and the number of rows used are not

specified in the paper, so it is assumed that the dataset is used in it's entirety.

## 3.2 Produced Features

For the quantitative analysis of tweet sentiment a new set of features is created for each Twitter account present in the dataset. These new features are shown in table 3.1 below.

| **Sentiment Feature** | **Description** |
|---|---|
| Count-Neutral | Number of neutral tweets per user |
| Count-Positive | Number of positive tweets per user |
| Count-Negative | Count of negative tweets per user |
| Sum-Positive | Sum of polarity scores of positive tweets per user |
| Sum-Negative | Sum of polarity scores of negative tweets per user |
| Average-Positive | Sum-Positive / Count-Positive |
| Average-Negative | Sum-Negative / Count-Negative |

Table 3.1: Sentiment features

In order to create these attributes the polarity score of each tweet is calculated with a python library for text processing called Textblob[1]. Textblob will parse through a tweet's text and produce a polarity score which can then be used to classify a tweet as *positive, negative* or *neutral*. Polarity score has value that range from -1 to 1. A negative value means that the tweet is of negative sentiment, 0 is for neutral and a positive value designates a positive sentiment.

## 3.3 Methods and Procedure

The machine learning models employed in this paper are *Random Forest, Support Vector Machine, Logistic Regression* and a *Feed-Forward Neural Network*. The main goal is to evaluate whether the newly created sentiment features can really help identify bot accounts and how much of an improvement do they pose compared to using readily available tweet and account metadata. This will also establish whether this approach can help examine tweets in a language other than English. In this

[1]https://textblob.readthedocs.io/en/dev/

section the four different methods used are presented along with the metrics they achieved, with and without the created sentiment features.

### 3.3.1 Random Forest

The RF method is applied as seen on the work of Battur and Yaligar published in 2018 [40]. The features used are the ones mentioned in section 3.1, and while not a lot of technical details are included in the paper, it is mentioned that 10-fold cross-validation is applied and 20 trees are created. The *cross-validation* resampling method uses different portions of given data to train the model on - that helps estimate the skill of a machine learning model on previously unseen data. The number of trees used is decided through *grid search*, a method for hyper-parameter optimization. Of the metrics used, this study focuses on *accuracy and the F1 score*, since these are metrics that are used in all examined papers and offer a common basis for comparison across the table. It is worth noting that while all methods are stated to have been tested on both the original feature set and the enhanced one, only in RF and FFNN the authors provide performance metrics for both feature sets. Metrics achieved by all methods are shown in table 3.2.

### 3.3.2 SVM

Two variations of the Support Vector Machine learning method are used: SVC - Support Vector Classification and SVR - Support Vector Regression. The data is split into 70-10-20 subsets for training, validation and test. The SVM kernel was configured to use a nonlinear hyper-plane polynomial function with a degree of 7 (default is 3). Overall, both variations achieved accuracy and F1 score that reached a bit over 90% and MCC of about 85%. More details in table 3.2.

### 3.3.3 Logistic Regression

Logistic regression is applied on the enhanced feature set and achieves accuracy of over 87% and F1 score of about 89%.These metrics are confirmed by the reimplementation of the original authors' methods that was done as part of this thesis.

### 3.3.4 Feed-Forward Neural Network

A feed forward neural network is an artificial neural network wherein the information moves in only one direction - forward. From the input nodes, through the hidden ones and onward to the output, the information never goes backwards or moves in a circle: that's the main difference to recurrent neural networks. This implementation of FFNN (also called ANN: Artificial Neural Network) uses two *hidden layers*. In addition to these, a *dropout* layer is added to the model to help prevent overfitting. A dropout layer will randomly set input units to 0 with a user-specified rate during training time. The input units that do not get picked for dropout are scaled up by 1 / (1 - rate) so that the sum of overall inputs stays the same. If a dropout layer is applied to an input vector, it nullifies some of its features and leaves unmodified the rest of them. If applied to a hidden layer, it nullifies some hidden neurons.

## 3.4 Results and Discussion

All above algorithms, along with the sentiment features, were reimplemented in order to confirm the authors' findings when it comes to performance. In most cases the results found are pretty close to the claimed results and can be seen in the score table 3.2 in parentheses. The authors' claim that using sentiment features can provide a modest, but not negligible, improvement in performance metrics is confirmed. Also, their conclusion that, among the methods tried, Random Forest and the Feed-Forward Neural Network are the most suitable in utilizing sentiment features to identify Twit-

| Dataset | Algorithm | Accuracy | F1 Score | Precision | Recall | MCC |
|---|---|---|---|---|---|---|
| C17 | Random Forest (original feature set) | 0.887 (0.9151) | 0.874 (0.8413) | (0.8559) | (0.8355) | 0.843 (0.6887) |
| C17 | Random Forest (enhanced feature set) | **0.923** (0.9167) | 0.912 (0.8434) | (0.8596) | (0.8362) | 0.887 (0.6931) |
| C17 | Random Forest SMOTE+ENN (enhanced feature set) | (0.9575) | (0.9210) | (0.9304) | (0.9152) | (0.8445) |
| C17 | Random Forest Tomek Links (enhanced feature set) | **(0.9681)** | (0.9410) | (0.9488) | (0.9356) | (0.8836) |
| C17 | FFNN (enhanced feature set) | 0.910 (0.91) | 0.927 (0.937) | (0.90) | (0.91) | 0.874 (0.4746) |
| C17 | FFNN SMOTE+ENN (enhanced feature set) | (0.9361) | (0.9308) | (0.9177) | (0.9212) | (0.6712) |
| C17 | FFNN Tomek Links (enhanced feature set) | (0.9228) | (0.9213) | (0.9398) | (0.9101) | (0.7129) |
| C17 | SVM(SVC) (enhanced feature set) | 0.914 (0.8341) | 0.922 (0.775) | (0.7624) | (0.8341) | 0.889 (0.1165) |
| C17 | SVM(SVC) SMOTE+ENN (enhanced feature set) | (0.9575) | (0.9210) | (0.9304) | (0.9152) | (0.6445) |
| C17 | SVM(SVC) Tomek Links (enhanced feature set) | (0.9371) | (0.9449) | (0.8267) | (0.8957) | (0.6883) |
| C17 | Logistic Regression (enhanced feature set) | 0.874 (0.8550) | 0.888 (0.8173) | (0.8253) | (0.8550) | 0.685 (0.3544) |
| C17 | Logistic Regression SMOTE+ENN (enhanced feature set) | (0.8933) | (0.8525) | (0.8491) | (0.8637) | (0.6779) |
| C17 | Logistic Regression Tomek Links (enhanced feature set) | (0.8721) | (0.8349) | (0.8517) | (0.8821) | (0.6117) |

Table 3.2: Performance Metrics in Sentiment Extraction

ter bots, is aligned with the findings of this thesis. The only metric that there is large gap between the original paper scores and the reimplementation ones is the MCC, especially on the SVM model and the Logistic Regression model. Whether the MCC is a preferable metric to the F1 score is debatable [41].

As an addition to simulating the authors' original work, two *data balancing* tech-

niques were also used in conjunction with the four machine learning methods discussed above. The first one is a combination of the SMOTE oversampling technique, combined with the ENN undersampling method. The second one is SMOTE combined with the Tomek Links undersampling method, inspired by the work examined in section 6.2.1. This kind of intervention benefited the classification effort greatly, as can be seen in table 3.2. In many cases, like with Random Forest, the scores achieved with the balanced dataset exceeded the scores reported by the authors. Even in cases where our reimplementation results did not closely align the ones reported by the authors, this data balancing act proved to close the distance by a significant measure. That could potentially point to data preparation techniques that were applied in the original work done by the authors, but were not mentioned in the paper.

While the approach with extracting the sentiment of tweets can bring some improvements with identifying bots, it's not a technique without shortcomings. One of them is that it is heavily reliant on third party text processing libraries, like in this case Textblob. Most of these libraries currently support only a limited range of languages which means that this technique is *not language-agnostic* for the time being. There's another, less technical, aspect of it that can be problematic - the distinction between bots and *trolls* [42]. The homogeneity in sentiment of tweets can be characteristic of not only automated accounts, but also internet trolls. It is very common for trolls that engage in online discourse, especially in polarizing subjects like politics and religion, to be very linear in the sentiments they express: everything is good, or nothing is. This approach can misidentify trolls, who are real humans, as bots.

It is useful to examine feature importance in this paper, so to compare with features that were most influential in the classification effort in other papers and gain a more complete understanding of the features that are more important across the board. Of the 13 features in the enhanced subset, the 5 most important ones are shown in figure 3.1 below. It's obvious that the top positions are dominated by sentiment features,

with the top spot going to the number of positive tweets an account has produced. It is noticeable that five out of seven sentiment features heavily influenced the accuracy of this approach, with only the 'AvgPositive' and 'CountNegative' features missing from the top spots.

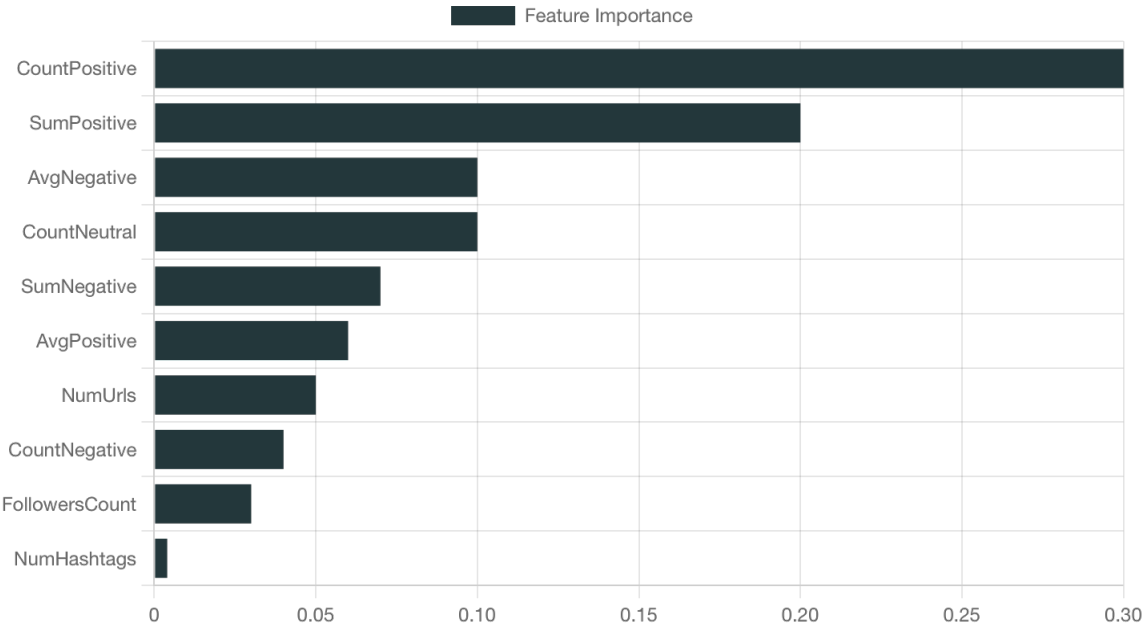


Figure 3.1: Feature importance in Sentiment Extraction

# Chapter 4

# Language-agnostic Bot Detection

The author of this paper gives another spin at identifying Twitter bots, without focusing on subcategories the bots could belong in but instead trying to adopt a **language-agnostic** approach. In this research effort the emphasis is on the prospective of detecting bots by account-level metadata only and establishing whether content-level analysis provides measurable benefits. The author also tries to explore the capability of identifying bots based only on account-level information and the ways that would extend to language-agnostic detection methods. The last research question is, with regard to how difficult is it to gather training data for bot detection, just how small can a dataset be and still provide adequate training results.

## 4.1 Feature selection

The dataset used here is once again the C17. A subset of the original features is used as the basis for producing other features that are more closely aligned to the research question this paper aims to answer. The core features, along with the ones produced, are shown in the table 4.1 below. Of a total of **68** features used here, only five are readily available in C17, the rest are derived from other features in the dataset.

## 4.2 Produced Features

The features used and namely the ones produced fall under two main categories, the *account-based* and the *content-based*, not dissimilar to other approaches we've seen so far. The account-based features are then categorised to two subcategories: the features that specifically target aspects of an account's *profile name*, and all others.

**Account-level features**

Drawn from account metadata present in the C17 dataset, some of these features are used in the same form they are found in, others are derived with simple calculations from available data. They basically fall under two categories, *generic profile features* and *profile name features*.

- *Profile name features:* The profile name a user chooses is one of the more personal aspects of having a Twitter account. It can be selected with varying purposes in mind, stretching from personalization to satire, even impersonation [43]. The rationale behind focusing on this specific aspect of a user profile is that, empirically, real users tend to be more creative in matters of online identity, compared to bot accounts that have shown to be more rigid and bland in their online representation. Some of these features are:

  (i) *screen_name_length*: A Twitter account has a screen name (e.g. Joe Doe) and a username, also called 'handle' (e.g. doe). This attribute refers to the length (meaning the number of characters) of the first one. Paradoxically, in C17 the screen name attribute is called 'name' and the username attribute is called 'screen_name'.

  (ii) *user_name_length*: The number of characters in the Twitter account's 'handle'.

  (iii) *screen_name_digits*: Number of (non-unique) digits in the screen name.

  (iv) *user_name_unicode_group*: This feature group refers to 105 features, one

for each Unicode code group, with binary values that work as flags and provide information as to whether a character that belongs in that Unicode group exists in the account's username. The reasoning behind this is that, the more Unicode groups an account's username has characters from the more creative it is, therefore it can be assumed that it is less likely to be a bot. The author does not go into much detail as to how they achieved that, nor the resources they used. In our reimplementation, since we could not find more information on the Unicode groups the author refers to, nor ways to identify them in a string of characters, we used a library called *sequence* from the *unicodeblock* package that can count the number of unique Unicode blocks in a string of characters.

(v) *screen_name_unicode_group*: See point above, applied to an account's screen name.

(vi) *levenshtein_user_name_screen_name*: It's been observed that bot accounts usually have screen names and usernames that are pretty similar, in an attempt to appear more legitimate. This feature tries to model this observation by calculating the Levenshtein distance between screen name and username [44]. The Levenshtein distance is a string metric for measuring the difference between two character sequences.

- *Generic profile features*: The more generic profile features are basic Twitter account metadata and can be seen in rows one to 13 in the first column of table 4.1. Some of these features are:

(i) *default_profile*: If a user has not filled in the 'bio' field in their profile, which is like a short profile description with maximum length of 160 characters, they are considered to have a *default* profile.

(ii) *geo_enabled*: Another Twitter profile aspect, this feature is about whether

a user has specified a geographic location in their profile. Location accuracy cannot be established and is therefore irrelevant.

(iii) *protected*: This flag, when true, means that the user has used the setting that hides one's tweets from non-followers. In order for someone to see protected tweets they have to make a *follow request* to the account whose tweets they want to see, and can see the tweets only in the case the account in question accepts the follow request.

(iv) *is_verified*: This feature refers to the so-called blue checkmark/badge that an account could get through a verification process done by Twitter. At the time the C17 dataset was created, the verification process and the blue badge were only considered for highly influential accounts like political figures, online influencers, corporate accounts of large companies etc. As of April 2023 the blue checkmark and the verification process are done through a Twitter subscription service and are entirely different to the legacy procedures that led to the verified status this feature is about [45].

(v) *friends_count*: The number of Twitter accounts being followed by an account.

(vi) *followers_count*: The number of Twitter accounts that follow an account.

(vii) *favorites_count*: The number of tweets an account has liked/marked as a 'favorite'.

(viii) *listed_count*: The number of lists an account is a member of. This feature would only count the number of *public* lists an account is in.

(ix) *statuses_count*: The total number of tweets and account has posted.

(x) *profile_use_background_image*: This flag, if true, means that the user has set a background image on their profile.

**Content-level features**

These features are about the content Twitter accounts actually produce, which is mainly the tweets they post. Most of these features are derived from the text body of the tweets and a few of them are based on readily available tweet metadata. Content-level features fall under two subcategories, *behavioral* and *core content*. Behavioral features are basically features that are based on time patterns and how often an account tweets. The core content features are produced in an attempt to derive intention and sentiment from tweets in a *language-agnostic* way.

- *Core content features*: Tweet text is tokenized by splitting it into character units that are separated by commas, colons, exclamation marks, brackets etc. In our reimplementation the tweets are split on white-space, which makes sure that characters that are part of an emoji set are not being left out. For distributional features there is an asterisk (*) right beside the feature's name, and it includes calculating a feature's mean, median, standard deviation, minimum and maximum values, skewness and kurtosis.

  (i) *number_of_tokens**: Distributional values of the number of tokens on an account-level.

  (ii) *number_of_hashtags**: Distributional values of the number of hashtags on an account-level. While the author does not specifically weigh in on this, in our reimplementation it was the number of *unique* hashtags per tweet that was used in the aggregation to account-level distribution. The hashtags were extracted by identifying the tokens that start with a '#' symbol, with the ones that exist in multitude getting filtered out so only one instance per hashtag remains. The unique hashtags extracted are then counted.

  (iii) *n_of_tokens_wo_hashtags_urls_symbols**: Distribution values of the number of *plaintext* tokens on an account-level. Excluded from this count are tokens of hashtags, URLs and all non-alphanumeric sequences. The author does not go into details as to how they did that, in the reimplementation

the tweet text was stripped of non-plaintext tokens by the use of the *tweet-preprocessor* library and its *clean()* utility[1]. Then the process followed is identical to the one in the 'number_of_hashtags' feature.

(iv) *number_of_urls**: Distributional values of the number of URLs per tweet. It is based on the 'num_urls' feature that is readily available in the C17 dataset and contains the number of (non-unique)URLs per tweet.

(v) *special_char_repeats_rate**: This feature is intended to detect sequences of special characters, specifically question and exclamation marks. It is not clear how the author modeled this feature but in the reimplementation the sequences of special characters were extracted, counted, and then used in producing distributional values.

(vi) *emojis_classic,kaomji_faces,line_art,other*: These features refer to different types of emojis detected and then used to extract sentiment out of a tweet's text. The reasoning behind targeting emojis is that it serves the purpose of achieving a language-agnostic approach of identifying bots. The author, again, provides no technical details as to how they were able to detect these specific types of emojis. Due to that, in our reimplementation we only focused on the Unicode emojis the *emoji*[2] python library can detect through the *list_emojis()* utility. The extracted emojis were then fed to the *get_emoji_sentiment_rank()* utility of the *emosent*[3] library that will then produce a sentiment score per tweet. These values are then aggregated on an account-level and distribution values are calculated, based on individual occurrences of emojis in single tweets.

- *Behavioral features*: These features are produced to support the theory that precise scheduling and rigid posting habits can point to automated behaviors and

[1]https://pypi.org/project/tweet-preprocessor/
[2]https://pypi.org/project/emoji/
[3]https://pypi.org/project/emosent-py/

ultimately bots. In an effort to model such behaviors this small set of features is created based on statistical properties of tweet **time** metadata. Distributional features come with an asterisk (*).

(i) *time_between_tweets**: Models time between consecutive tweets and calculates mean, median, standard deviation, min and max values, skewness and kurtosis. The implementation of this was pretty straightforward since every tweet comes with a timestamp of the moment it was posted.

(ii) *time_between_retweets**: Similar to the previous feature but this is one is only about retweets. In the reimplementation done to simulate the author's effort, we extracted all retweets per account. The retweets can be found in the dataset by the capitalized 'RT' characters at the beginning of a tweet's text. Retweets per account are then sorted and the time between consecutive retweets is recorded. The first retweet in the list will have a time difference of zero. With the time between retweets now found the distributional values, mentioned above, are calculated per account.

(iii) *tweet_rate*(avg): The average number of tweets posted daily. Calculated by dividing the total number of tweets posted by an account by the account's age in days. The accounts' age was calculated at a set time of June 1st 2019, the approximate date this paper was published.

## 4.3 Methods and Procedure

In this paper the measures of model performance are *accuracy, precision, recall, F1 score* and *AUC_ROC - area under the ROC curve*. In order to have results more directly comparable to performance metrics in other papers, our reimplementation focuses on accuracy, precision and F1 score. The authors report that they experimented with the typical machine learning methods suitable for the task like *Logistic*

| No. | Account-level | Content-level |
|---|---|---|
| 1 | default_profile | time_between_retweets* |
| 2 | geo_enabled ratio | time_between_tweets* |
| 3 | protected | tweet_rate(avg) |
| 4 | is_verified | emojis_classic* |
| 5 | friends_count | emojis_kaomji_faces* |
| 6 | followers_count | emojis_line_art* |
| 7 | favorites_count | emojis_other* |
| 8 | listed_count | number_of_tokens* |
| 9 | statuses_count | number_of_hashtags* |
| 10 | profile_use_background_image | n_of_tokens_wo_hashtags_urls_-symbols* |
| 11 | Number of likes given | number_of_urls* |
| 12 | Number of tweets | special_char_repeats_rate |
| 13 | Tweets per day | - |
| 14 | screen_name_length | - |
| 15 | user_name_length | - |
| 16 | screen_name_digits | - |
| 17 | user_name_unicode_group | - |
| 18 | screen_name_unicode_group | - |
| 19 | levenshtein_user_name_screen_name | - |

Table 4.1: Language-agnostic Bot Detection Features

*Regression, Support Vector Machine, Random Forest, Multi-layer Perceptrons* and *AdaBoost*, with AdaBoost emerging as the method having the most high valued metrics. Only the performance values achieved by AdaBoost are listed in the paper.

In our reimplementation, all mentioned methods are implemented and their results are shown in table 4.2 below. Results shown in parentheses are the ones produced by the reimplementation. Since the authors only provided results from the AdaBoost method, only those numbers are seen in the paper. It is worth noting that the authors tested the listed methods in a variety of subsets of the features shown in table 4.1, before finally testing them on the whole feature set. Our comparison was done with the results of the methods applied on the whole feature set, which is what brought the best results in the original paper.

| Dataset | Algorithm | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|---|
| C17 | AdaBoost | 0.9881 (0.8167) | 0.9958 (0.8377) | 0.9835 (0.8167) | 0.9896 (0.8115) | 0.9959 (0.9505) |
| C17 | Logistic Regression | (0.9287) | (0.9429) | (0.9355) | (0.9447) | (0.9879) |
| C17 | SVM | (0.9266) | (0.9317) | (0.9241) | (0.9354) | (0.9648) |
| C17 | Random Forest | (0.9554) | (0.9512) | (0.9554) | (0.9525) | (0.9388) |
| C17 | Multi-layer Percep-trons | (0.9287) | (0.9455) | (0.9355) | (0.9447) | (0.9879) |

Table 4.2: Language-agnostic Bot Detection Performance Metrics

### 4.3.1 AdaBoost

The AdaBoost classifier is a machine learning technique that is used as an *ensembling* method. Ensembling is a *meta* approach (meaning it's a machine learning algorithm that learns from the output of other machine learning algorithms) to machine learning, that can achieve better performance metrics by combining predictions from multiple models, often weak learners. With AdaBoost the weak learners are decision trees with a single split that are called *decision stumps*. The way AdaBoost works is it progressively puts more weight onto difficult to classify instances, and less weight on those that are already handled well. It initially fits a classifier on the original dataset and then fits extra copies of the classifier on the same dataset, but with modified weights for incorrectly classified instances. These weight modifications will shift the focus on the cases that are more difficult to classify. AdaBoost algorithms can be used for classification and regression problems.

The authors in this paper use AdaBoost after resampling the training data using *SMOTE-ENN*. SMOTE-ENN is a method developed by Gustavo Batista et al. in 2004, and it combines the SMOTE ability to generate synthetic examples for the minority class of a dataset and the ENN capacity to delete observations from both classes identified as being of different class to the observation's class and its K-nearest neighbor majority class. The authors resampled the training subset which was then used with the AdaBoost classifier. Reported accuracy for AdaBoost is 0.9896 and ROC-

AUC is 0.9959. In our reimplementation we could not confirm these metrics, possibly because applying SMOTE-ENN to the training dataset was very time-consuming and had to be abandoned. Our implementation of AdaBoost produced accuracy of 0.8167 and ROC-AUC of 0.9505.

### 4.3.2 Logistic Regression

There are no technical details of the authors' implementation of Logistic Regression or the performance metrics it achieved in the research paper. Our reimplementation of Logistic Regression achieved metrics that start from 0.9287 and stretch up to 0.9879. More details can be seen at the second row of the 4.2 table.

### 4.3.3 SVM

No technical details were provided as to how SVM was implemented in this paper, and no performance metrics to compare to. In our reimplementation the SVC - Support Vector Classification variation of SVM was used, with the dataset split in an 80-20 analogy for training-testing and a polynomial function of degree 7. Results achieved are similar to the ones brought by Logistic Regression and they are in the range of 0.9266 (accuracy), up to 0.9648 (ROC-AUC).

### 4.3.4 Random Forest

The Random Forest algorithm used in our reimplementation had 20 trees, used the *Gini* impurity function to measure the quality of the splits and employed 5-fold cross-validation to evaluate scores. This configuration was decided based on previous efforts that involved the Random Forest algorithm and performed well. The metrics RF achieved came up on top of all methods used in this paper, including AdaBoost, which is something that contradicts the authors' inferences. Accuracy achieved is 0.9554, with other metrics hovering a bit over 0.93.

### 4.3.5 Multi-layer Perceptrons

A multilayer perceptron is a type of fully connected FFNNs - Feed Forward Neural Networks. A fully connected neural network is a class of artificial neural networks with architecture where *all* neurons, or nodes, of a layer are connected to the neurons of the next layer. An MLP is made of, at least, three layers of nodes. These typically are an *input* layer, a *hidden* layer and an *output* layer. All nodes in the MLP, except the input nodes, use a nonlinear *activation function*. It is these multiple layers and the activation function that distinguish an MLP from a linear perceptron.

In our reimplementation the MLP consisted of two hidden layers and achieved accuracy of 0.9287 and AUC of 0.9879, with the rest of the metrics in the 0.9+ range.

## 4.4 Results and Discussion

All performance metrics achieved by the reimplementation of the machine learning methods that this paper utilized can be seen in table 4.2. Scores in parentheses for the AdaBoost algorithm show the reimplementation's performance, the values without parentheses are the performance claimed by the authors. AdaBoost is the only method for which the authors provided performance scores and is the method that achieved the best metrics in their implementation. Our results were not the same, with Random Forest coming up on top of our metrics, which can be attributed to technical difficulties in implementing SMOTE-ENN resampling. The top scores, however, between the authors' AdaBoost and the reimplementation's RF are not wildly different, with AdaBoost achieving metrics around 0.98 and RF performing at the 0.95+ range.

As with the previous two papers examined, it is useful to go over the most important features in the dataset that are the most influential in the classification's performance. By using the *RandomForestRegressor*[4] library of sklearn's *ensemble*

[4]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

package, we extracted the top 17 most important features for the Random Forest algorithm that achieved the best performance in our reimplementation, which can be seen in figure 4.1 below.

It is noticeable that the top spot in feature importance is shared by features that are basically the number of words in a tweet and the number of URLs in a tweet. Followed by the *TimeBetweenRetweets_Kurtosis* and the *TimeBetweenRetweets_Skewness* features that are a good indication of the consistency and how often an account retweets other tweets. It is useful to note here that a retweet does not necessarily mean that the tweet 'shared' belongs to another account. An account can retweet their own tweets, meaning a traditional bot can post a URL with a tweet and then retweet that same tweet through the day just to refresh its exposure and the attention it gets. We can also see two Unicode-related features in prominent positions in the list, the *ScreenNameUnicodeGroups* and the *UsernameUnicodeGroups*. The presence of these two gives ground to the theory that real users can be more creative when it comes to choosing their screen and user names. There is a also a feature that exploits the emojis that are sometimes present in tweets, the *EmojisSentiment_Median*. This emoji feature however cannot strongly support the viability of a language-agnostic approach in bot detection, as many of the other top features are content-level.

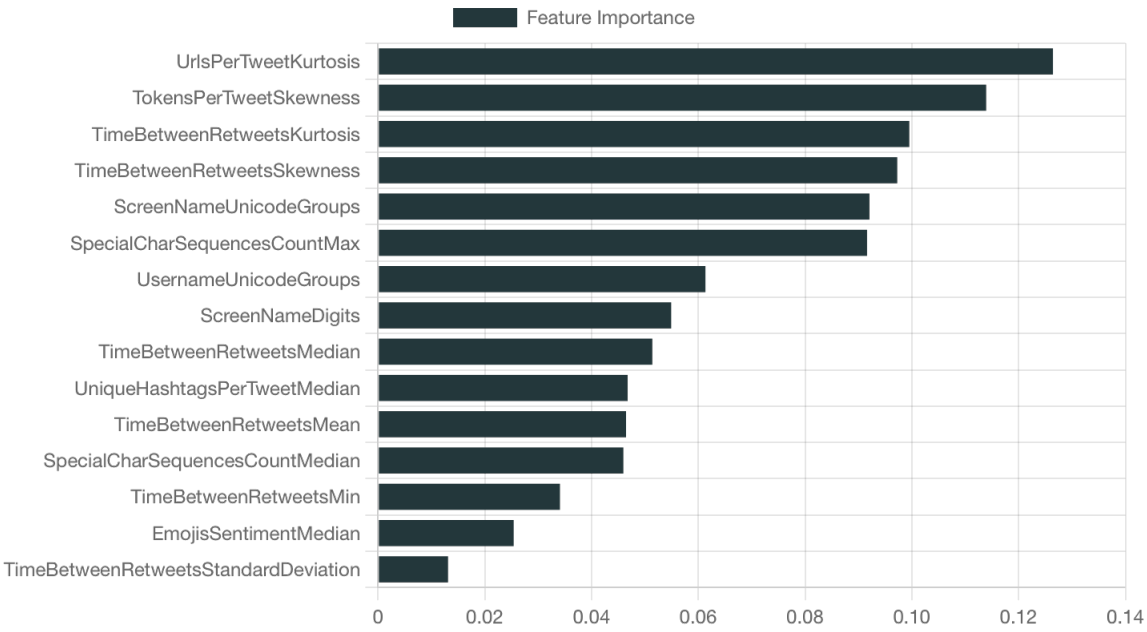


Figure 4.1: Language-agnostic Bot Detection Feature Importance

# Chapter 5

# Bot category classification

In this study the author takes the Twitter bot identification effort one step further, by not only detecting a bot but also classifying it to a **bot category**. With today's diversified landscape of automated accounts, there are many bots that are not only safe for the platform to keep online, they are actually useful for the user-base. For example the @$RemindMe_OfThis$ bot can be used as a reminder to revisit specific tweets later in time. @*threadreaderapp* will unroll a long tweet thread and provide a link to a page with all the thread's content in an easy to read format. @$pikaso_me$ can produce screenshots of tweets when mentioned in a comment under the tweet. These bots, along with other sophisticated social bots and sockpuppets give ground to the argument that not all types of bots should be lumped together into the same, unified group. And this paper's research goal is to identify the machine learning method that would perform best in this multinomial classification task.

## 5.1 Feature selection

Due to the large number of features available in the C17 dataset, a feature selection must be done. The most straightforward way to do this, and the one used by the author, is selecting the features used in similar works that have shown to be working well. In this case the features selected were used in a similar manner by J. Fernquiest and his team in their work about political bots in the Swedish general election [46].

In Fernquiest's work a total of 140 features were used, many of them based on the features of the C17 dataset. About 20 of those features were produced by fetching additional Twitter data through Twitter's API, including some features that it is not perfectly clear how they were calculated. So in this paper, 120 features were used, the core of which can be seen in table 5.1.

| No. | Account-level | Content-level |
|---|---|---|
| 1 | Age of account | Hashtags per tweet |
| 2 | Follower-friend ratio | Hours of day tweeting |
| 3 | Given likes per follower | Length of tweets* |
| 4 | Given likes per friend | Mentions per tweet |
| 5 | Has location | Normalized distribution hours tweeting |
| 6 | Has default profile description | Normalized distribution of tweet endings |
| 7 | Length of username | Normalized distribution weekdays tweeting |
| 8 | Likes per day | Number of words* |
| 9 | Number of followers | Retweets achieved per tweet |
| 10 | Number of friends | Retweet-tweet ratio |
| 11 | Number of likes given | Time between mentions* |
| 12 | Number of tweets | Time between retweets* |
| 13 | Tweets per day | Time between tweets* |
| 14 | - | Time between URLs* |
| 15 | - | Unique hashtags per tweet |
| 16 | - | Unique mentions per tweet |
| 17 | - | Unique sources |
| 18 | - | URLs per tweet |
| 19 | - | Weekdays tweeting |

Table 5.1: Bot Category Classification Core Features

## 5.2 Produced Features

It is important to point out that some entries in 5.1 actually enclose multiple features. And while some of them like 'Age of account' seem quite straightforward, for others it's better to go into a bit more detail and clarify what they actually entail. In addition to that, features marked with an asterisk (*) represent five different features: *mean, median, standard deviation, min* and *max* values. Most other features, unless noted otherwise, correspond to the *mean* value of whatever tweet metadata the feature

designates. For account-level features all entries correspond to one feature each.

**Follower-friend ratio** A 'follower' in Twitter terminology is an account that follows another account, meaning the follower will see the follower's tweets in their timeline. The follower is the follower's 'friend'. The follower-friend ratio is basically the relation between the number of users following an account and the number of accounts that account follows itself (*followers_count**friends_count*).

**Has location & Has default profile description** Each Twitter profile can exhibit a 160 character profile description called 'Bio', along with a location and a website the user would like to have on display. Providing this information is not obligatory and if a user has opted to leave these fields empty it is considered that they have have a *default* profile. These binary, account-level features are shown on lines 5 and 6 of table 5.1. Value of 1 means true and 0 means false.

**Tweets per day & Likes per day** These features are calculated by dividing the count of each measure by the age of the account in days. The age of the account can be seen at the *created_at* feature of the users subset in the C17 dataset. In the reimplementation of this paper's work the date of reference selected was June 1st 2019, in order to approximate the time period within which the paper was submitted.

**Weekdays tweeting** This feature actually refers to 7 features, each feature being the count of tweets an account has posted on a specific day of the week. So in the final dataset used to train the machine learning algorithms this feature can be seen as seven distinct features (e.g. Tweets_Monday, Tweets_Tuesday) with integer values.

**Hours of day tweeting** For each Tweet posted there is a timestamp of when that was done at the C17 *created_at* feature of the tweets subset (not to be confused with the feature with the same name in the *users* subset which is the account's creation date). Using the hour part of the timestamp it is easy to count the number of tweets posted at a specific hour of the day throughout the account's lifetime. This leads to the creation of 24 features, each one containing the number of tweets posted by the account at each respective hour.

**Normalized distribution hours tweeting/tweet endings/weekdays tweeting** These features are basically the *'Weekdays tweeting'* and *'Hours of day tweeting'* features in the form of percentages. They basically show what percentage of an account's tweets are posted on each day and on each hour. This means that another 7 features are created for the distribution of tweets per day and twenty-four for the distribution per hour.

**Tweet endings** The ways a tweet can end are as many as any letter, number or symbol available. It would obviously be impossible to create a designated feature for each possible ending, therefore features are created for each one of the following endings, with ending being the last character in the tweet: *period, question mark, exclamation mark, lower case letter, upper case letter, digit*. There is also a catch-all feature for all other possible endings called *'other tweet endings'*. For each one of these cases the number of tweets, per account, is counted and then normalized, in the same way the hours/weekdays tweeting features are handled. With these features created it is visible at glance what percentage of tweets an account posts ends in a similar manner.

**Retweet-tweet ratio** This feature measures the analogy of the number of original tweets an account posts versus the number of tweets it 'retweets' or forwards. A retweet can be done by an account on its own tweets or on the tweets posted by other accounts. The C17 dataset includes a feature called 'retweet_count' that is the count of tweets an account has retweeted. The values in this feature were confirmed in the reimplementation of this paper's work by counting the number of tweets per account that started with capitalized RT letters. For the retweet-tweet ratio the number of retweets per account is divided by the original tweets posted by the account ($NumberOfRetweets \backslash NumberOfTweets$). On a side note, the *Retweets achieved per tweet* feature is a content-level tweet that's basically the number of times a tweet was retweeted.

**Time between mentions\tweets\retweets\URLs** These features measure the

time between tweets posted by an account that share similar characteristics. The intention was to be able to see how much time passed between, for example, two tweets that include mentions or URLs, or are retweets. Mentions are tweets that are directed to specific users in a thread by stating their username preceded by the symbol. Only the most recent 100 tweets per account are taken into consideration for these features. Not much detail is given in this paper as to how this was implemented so in the reimplementation the first step was to extract three different subsets of the C17 dataset where all tweets either include mentions or URLs or are retweets. Then a maximum of 100 most recent tweets were kept for each account. With the time a tweet was posted available in the original features it was easy to calculate the difference, in hours, between two consecutive tweets. With the time difference between tweets now calculated the mean, median, standard deviation and minimum and maximum values are now computed per account. This exercise is done four times, one for the tweets that include mentions, one for the retweets, one for tweets that include at least one URL and one for the rest of the tweets. This results to a total of 20 new features which are then merged with the original dataset.

**Unique hashtags, mentions, sources per tweet** Similar to the *'Hashtags per tweet'* and *'Mentions per tweet'* features, but different in that they include the number of unique instances of a hashtag,mention or source. *Source* of a tweet is the platform through which the tweet was posted, and it can be the Twitter web app, the official iOS and Android Twitter apps or any 3rd party app available like Tweetbot or Aviary. It should be noted that as of January 20 2023, Twitter has blocked all API access to 3rd party apps effectively banning all non-official clients from using Twitter services [47]. For example if an account is being accessed and used to post from Twitter's web app and from the Android mobile app, the 'Unique sources' feature's value would be 2. In the simulation of the author's implementation, both hashtags and mentions were extracted to different lists per tweet. These lists were then traversed to count both total instances and unique instances per tweet. This approach is not without

flaws as it can potentially take into account words with preceding hashtags or at signs (@), though that is considered to have nominal impact.

## 5.3 Methods and Procedure

The measures of model performance in this paper are *accuracy, precision, F1 score* and *MCC - Matthews Correlation Coefficient*. The first three are commonly used in multinomial classification and MCC, while originally developed for binary classification, has been generalized into a version that can handle multinomial classification as well. *Macro average* was used with these metrics, which is computing the metric for each class separately and then averaging over the number of classes (micro average would be to use all the class combined to compute the average). MCC is a more reliable statistical rate that produces high scores only if its predictions achieved good results in all four confusion matrix categories (true positives, false negatives, true negatives, false positives), comparatively to the size of positive elements and the size of negative elements in the dataset.

As previously mentioned in paragraph 2.3.5, the way the imbalance between class subsets is handled is through *undersampling*. That leads to a dataset of 4.000 tuples, of which 80% (3200 rows) is used as a **training** set and 20% (800 rows) as a **test** set. No validation set is used and hyper-parameter tuning is done through cross-validation. Hyper-parameters have to be chosen by the model creator and include, for example, the weight of penalty in logistic regression, the number of trees in random forest, the rate of learning in neural networks, the number of neighbors in k-nearest-neighbors, and others.

One way to make parameter tuning more targeted and less arbitrary is through *cross-validation*. In this paper a 5-fold cross-validation process is used to do that, and the way it works is that it splits the training dataset into five different parts, uses a combination of hyper-parameters to train the model on four of these parts and tests it on the fifth part. The model is measured on its accuracy and then the process is

repeated with a different combination of training sets and testing set. The average of accuracy measurements is recorded for this cycle. When all five combinations are performed on the model, another cycle starts with another set of hyper-parameter values. When all selected combinations of hyper-parameters are used on the model, the combination that performed the best is nominated to be used on the test set [48].

### 5.3.1 Random Forest

For the RF algorithm the hyper-parameters that can be tuned are the *number of trees* used and the *number of features* searched in each split. Through the cross-validation process described above, the number of tress is set to **200** and the number of features that are used for each split is set to the square root of the total number of features ($\sqrt{120} \approx \mathbf{11}$). In creating the RF model, the function to measure the quality of a split was set to criteria value of *"gini"* for the Gini impurity measure. The final outcome of a decision tree, like the RF, is heavily influenced by its hierarchical structure. To decide on this structure it is necessary to answer a couple of questions, like which feature should be placed at the root node, or which features should act as internal nodes and which as leaf nodes. The Gini index is a splitting measure that helps answer these questions. Also called Gini coefficient or impurity, it computes the probability of a specific variable to be erroneously classified when chosen randomly. When building a decision tree it is preferable to choose the feature with the smallest value in Gini index as the root node. The author does not explain the reasoning behind choosing Gini index but it is likely because of it being more computationally efficient to alternatives like 'Information Gain'. Metric results are discussed in section 5.4.

### 5.3.2 Logistic Regression

Logistic regression is a parametric classification model, despite having the word 're-gression' in its name. LR models have a fixed number of parameters that depend on

the number of input features and they output categorical results - binary classification. In general, multinomial logistic regression builds a linear predictor based on the transformation of linear combinations of features, in order to build class probabilities. In this paper's application of LR and in order to escape overfitting, regularization with a $L_2$ penalty term is used. The tuning parameter for regularization is set to be *one*, value found through cross-validation. In the reimplementation done to confirm the author's findings, a maximum number of 500 iterations was arbitrarily chosen as a parameter in training the model. The maximum number of iterations is the maximum number of iterations taken for the solvers to converge.

### 5.3.3 ANN

Artificial Neural Networks are computational models that are inspired by the biological human networks that constitute the human brain. An ANN is technically a bunch of artificial neurons that are interconnected to each other. The information coming through is processed by filtering done at the densely connected artificial neutrons. Each connection between neurons can transmit the signal from one neutron to another. The neurons are structured into several sequential layers, with each neuron in a layer being connected to all neurons in the previous layer and the next. Each layer receives input from the previous layer, processes it and passes it on to the next layer. The first layer in the layout is called **input** layer, and it mainly receives the input and feeds it to the next layer. If an input layer receives $p$ features, it'll have $p + 1$ neurons. The extra neuron constantly outputs a signal of one value to introduce a bias term, similarly to the intercept term in regression models. Right after the input layer are the **hidden** layers, where the processing takes place. The number of hidden layers that can be stacked is up to the model creator, though with adding hidden layers comes a trade-off in computational speed. Each hidden layer transforms linear combinations of the output of the previous layer, whether input or another hidden layer, which are then pushed forward to the **output** layer. Each hidden layer intro-

duces a bias term with an extra neuron in the same way that is done in the input layer. The output layer produces estimated probabilities for each class. If the output layer has $p$ features, it'll also have $p$ neurons. Each neuron in the output layer will receive a linear combination of outputs from the hidden layer and transform it to estimated probabilities for each observation to belong to a specific class. In this implementation the *softmax* function is used for the transformation described above, and the *sigmoid function* is the activation function used by the hidden layer. By applying cross-validation it is decided to only have one hidden layer with nine neurons in each one of them. Fitting the ANN starts by randomly assigning values to all weights, most often by a distribution with values close to zero. Then an observation from the training set is fed into the model which then produces a vector of predicated class probabilities. The target values are already known, so the gradient of the error function can be computed. The gradient is computed through a process called *back-propagation*, which is the technique that is used in this paper, because it is considered to be faster than *forward-propagation* [49]. The weights are then updated by using *gradient descent*. This procedure is repeated for every tuple in the dataset. When all tuples of a dataset are fed into the model, one cycle called *epoch* has passed. The model is considered to be *fitted* when the designated number of epochs has passed.

The features present in the dataset are *standardized*, meaning they are given mean value of zero and variance value of one. This will help avoid having features with higher absolute values being credited with higher importance by the regularization process [48]. Standardization process adds a penalty term to the error function, which will in turn shrink the weights towards zero and decrease the risk of overfitting.

### 5.3.4 k-Nearest Neighbors

The simplest among the methods used in this paper, k-NN classifies new data by measuring the *Euclidean distance* between the feature vector of a new observation and every observation in the training subset. The algorithm will then sort the found

distances in a descending order and will appoint the observation in question to the class most commonly found in the top $k$ entries of the sorted list (k = number of neighbors). The author of this paper tested a range of values through cross-validation (1, 2, 3, 4, 5, 10, 20, 50) but does not point out the value that brought the highest performance. In the reimplementation done as part of this project the number of neighbors used is **2**.

Before training the model, all features are standardized to have mean zero and variance one - because the only way for Euclidean distance to make sense as a distance measure is to have everything measured in the same unit system.

## 5.4 Results and Discussion

The results of this paper can be seen in table 5.2. Values in parentheses are the results produced by the reimplementation of the methods described in the paper. It obvious that in many cases the results brought by our reimplementation are very close to the ones claimed by the author.

With the C17 dataset, the author's data indicates that the Random Forest model followed by the ANN are the best performing by most measures. And while our reimplementation was able to replicate the scores achieved by RF and ANN, it is Logistic Regression and k-NN that bear the best results among methods employed.

| **Dataset** | **Algorithm** | **Accuracy** | **Precision** | **F1 Score** | **MCC** |
|---|---|---|---|---|---|
| C17 | Random Forest | 0.9912 (0.992) | 0.9913 (0.993) | 0.991 (0.992) | 0.988 (0.990) |
| C17 | Logistic Regression | 0.975 (0.998) | 0.975 (0.998) | 0.975 (0.998) | 0.966 (0.997) |
| C17 | ANN | 0.9800 (0.979) | 0.98019 (0.976) | 0.979 (0.981) | 0.973 (0.972) |
| C17 | k-NN | 0.9262 (0.998) | 0.9266 (0.999) | 0.925 (0.999) | 0.902 (0.999) |
| C15 | Random Forest | (0.971) | (0.976) | (0.966) | (0.938) |
| C15 | Logistic Regression | (0.974) | (0.972) | (0.974) | (0.944) |
| C15 | ANN | (0.968) | (0.973) | (0.973) | (0.952) |
| C15 | k-NN | (0.974) | (0.999) | (0.999) | (0.999) |

Table 5.2: Bot Category Classification Performance Metrics

The models used on the C17 dataset were also applied on the C15 dataset, with results visible on the last four lines of table 5.2. It is noteworthy that all models achieve accuracy safely above 95%, which given the results seen in chapter 3 and other works in the field seem relatively good. It also goes to show that the methods applied in this paper can indeed handle new, previously unseen data and provide good classification accuracy. Another point regarding metrics is that precision, F1 score and MCC do not really provide any useful information regarding the models' performances. The goal of these metrics is to identify patterns in classification efforts that accuracy does not succeed in finding. With precision, F1 score and MCC so close to accuracy results, no novel aspect of the models is revealed.

It is also useful to examine the feature importance (here obtained from the RF algorithm), in order to gain a better understanding of which features are most critical to classification performance, and how many of them are in the original set of features. The ten most important features of the ones used in this paper are shown in the figure 5.1 below. Note that the feature importance metrics were taken by the reimplementation of this paper's methods.

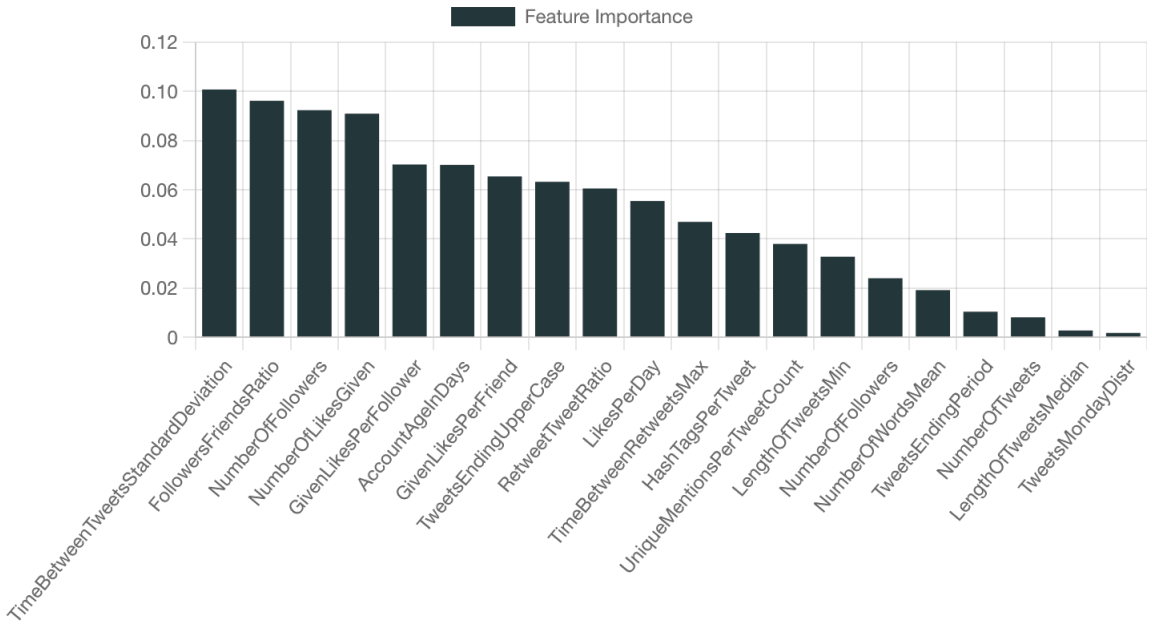


Figure 5.1: Bot Category Classification feature importance

The top spot in importance is taken by a content-based feature, and it seriously outweighs all other features. The features that follow are a mix of content-based and account-based features, meaning that there is no specific type of features that complete dominates the list. Only one of the *text-oriented* features like length of tweets, number of words etc has made it to the list, ('TweetsEnding_UpperCase'). None of the generic *time-oriented* features, like time or day of the week tweeting, have made it to the list, however, it is the feature about the standard deviation in time between two consecutive tweets that takes the top spot.

# Chapter 6

# Model optimization and feature minimization

At this point it's been established that social media, Twitter included, are justifiably praised for being able to democratize opinion sharing and empowering online discourse. But it's also apparent that they can facilitate the spread of political agendas, misinformation, fate and extremism [3]. These types of malign effects have been, for the most part, been linked to software-controlled social media accounts.

The biggest challenge in detecting this type of accounts, commonly known as bots, is the lack of **extensive** enough datasets that will support the development of machine learning models that are based on supervised learning. In the previous papers we examined, most methods we encountered focus on detecting bots on a combined account-level and content-level basis. That means that an algorithm would process the activity record of an account, a couple dozens of tweets, along with the account profile's metadata and try to determine whether the account in question is a bot or not. And while these methods are reasonable successful in that regard, the ones that really excelled, like the one featured in chapter [5], required a lot of data and plenty of produced features.

It is this paper's focus to try and remedy that by providing an approach that will: *a*) be able to identify a bot account by a single tweet, *b*) be able to enhance existing datasets by producing more samples of human and automated accounts,

while avoiding the strenuous cost in time, effort and resources, of data collection and manual annotation. The authors use two methodological approaches to achieve these goals, which are outlined in sections 6.2 and 6.3.

## 6.1 Feature selection

In this work the authors use the, by now familiar, C17 dataset [28]. The dataset is not used in its entirety, but only the subsets of *genuine accounts* and *social spambots*. This part of the dataset makes for a total of 8,386 twitter accounts and almost 12 million tweets.

Though many approaches in the field utilize large numbers of features, memorably the work of Allen Davis et al. utilizes north of 1,500 features [50], studies have shown that using a modest set of features can lead to similarly high performance in detecting bots as their maximalist counterparts [51, 52]. Smaller number of features will lead to more **efficient** models that can be trained in shorter time intervals. It can also help produce models that avoid *overfitting* and are more easily *interpretable*. In this paper a total of **16** features are used, 10 for account-level data and 6 for content-level. Features used are shown in table 6.1 below.

| No. | Account-level | Content-level |
|---|---|---|
| 1 | default_profile | retweet_count |
| 2 | geo_enabled | reply_count |
| 3 | protected | favorite_count |
| 4 | is_verified | number_of_hashtags |
| 5 | friends_count | number_of_URLs |
| 6 | followers_count | number_of_mentions |
| 7 | favorites_count | - |
| 8 | listed_count | - |
| 9 | statuses_count | - |
| 10 | profile_use_background_image | - |

Table 6.1: Model Optimization and Feature Minimization Features Used

## 6.2 Approach 1: Account-level classification

This approach focuses on proving that bot detection can be done on an account-level, meaning there is no tweet data involved, with a small number of highly interpretable features that require little to **no preprocessing** and machine learning architectures of **low complexity**. This prospective is supported by similar work that underlines the importance of Twitter accounts' metadata as a strong factor in identifying automated accounts [52].

### 6.2.1 Methods and Procedure

Within the context of this approach a number of familiar machine learning methods are tested, specifically *Logistic Regression, Stochastic Gradient Descent, Random Forest* and the *AdaBoost* classifier. These techniques are evaluated in three different iterations, one with the dataset in its original form and two with *oversampling techniques* used in order to balance the dataset. The oversampling methods used are basically different variations of the SMOTE (Synthetic Minority Oversampling Technique).

The way SMOTE works is that it produces samples based on the feature set of the minority examples, which are the classes that have the smallest representation in the dataset. Usage of SMOTE is combined with two different *undersampling* techniques that will help balance out the potential *bias* introduced by oversampling. Two undersampling techniques are used, in combination with SMOTE: *ENN - Edited Nearest Neighbors* and *Tomek Links*.

The **ENN** rule for undersampling was introduced by Dennis Wilson in 1972 [53]. The way it works is it uses the $K = 3$ nearest neighbors to identify datapoints that are misclassified and then proceeds to remove them, before applying a $K = 1$ classification rule. The combination of SMOTE and ENN is reported by the authors to bring a significant improvement in classification accuracy, compared to when applied to the

dataset in its original form. In our implementation we also encountered improvement in performance metrics, though not as drastic as the one claimed by the authors. More details can be seen in table 6.2.

**Tomek Links** is another undesampling technique proposed by Ivan Tomek in 1976 [54]. It is considered an improvement on the CNN - Condensed Nearest Neighbors undersampling technique, that randomly selects samples with their $k$ nearest neighbors that need to be removed from the majority class. The Tomek Links methods builds upon the CNN by selecting pairs of observations that meet a couple of requirements, specifically for two observations **a** and **b**:

1. Observations **a**'s nearest neighbor is observation **b**.
2. Observations **b**'s nearest neighbor is observation **a**.
3. Observations **a** and **b** are not in the same class. One of them belongs to the *majority* class and the other to the *minority* class.

The authors report little improvement in performance when combining SMOTE with Tomek Links, and that is also observed in our reimplementation. Performance metrics chosen are *accuracy, precision, recall, F1-Score* and *AUC_ROC*.

### 6.2.2 Logistic Regression

No technical details are given as to how Logistic Regression was implemented. What's known is that it was ran with three different configurations: on the dataset without applying balancing techniques, then with SMOTE+ENN and finally with SMOTE+Tomek. In the first case the scores are in the range of 0.88 - 0.94, in the second case scores point to near-perfect accuracy with numbers in the range of 0.98 - 0.99 and in the third case scores are 0.90 to 0.92. These scores point to the SMOTE+ENN configuration as the most effective in bot classification, combined with Logistic Regression.

In our reimplementation the results are mostly similar. Logistic Regression without data balancing achieved scores in the 0.90 - 0.91 range. The SMOTE+ENN configura-

tion had scores in the 0.91 - 0.97 spectrum and the SMOTE+Tomek brought accuracy within 0.90 - 0.92. These configurations were also tested on the C15 dataset with similar results: No balancing: 0.89 - 0.91, SMOTE+ENN: 0.97 - 0.98, SMOTE+Tomek: 0.89 - 0.91.

### 6.2.3 Stochastic Gradient Descent

SGD was tested in three iterations as well. The metrics with the original configuration were the lowest among the three, with performance in the 0.86 - 0.87 range. SMOTE+ENN brings vast improvements, with performance in the 0.94-0.95 range. SMOTE+Tomek comes second with scores a bit over 0.90.

SGD in the reimplementation brought scores in the 0.91 - 0.92 range for the original configuration. SMOTE+ENN was the top of the three with performance between 0.92 - 0.93. SMOTE+Tomek was not far off with scores in 0.88 - 0.90. Also tested on the C15 dataset: No balancing: 0.91 - 0.92, SMOTE+ENN: 0.93 - 0.95, SMOTE+Tomek: 0.89 - 0.90.

### 6.2.4 Random Forest

Random Forest has been a top performer throughout all papers we've examined so far and this is no exception. RF with the original dataset configuration achieves scores of over 0.98 in all metrics. With SMOTE+ENN scores are near perfect, with values over 0.99. SMOTE+Tomek has similarly high metrics in the 0.98 - 0.99 range.

In the reimplementation Random Forest also performed well: with the original dataset metrics hovered above the near-perfect 0.98 point. SMOTE+ENN was between 0.98 and 0.99 and SMOTE+Tomek around the 0.98 point. With the C15 dataset: No balancing: 0.96 - 0.98, SMOTE+ENN: 0.97 - 0.99, SMOTE+Tomek: 0.97 - 0.98.

### 6.2.5 AdaBoost

Adaboost had the best performance in two of the three configurations noted above. With the original dataset it achieved metrics of over 0.98, with SMOTE+ENN it went even higher at 0.9981 of accuracy and AUC and with SMOTE+Tomek it hovered over 0.98.

In the reimplementation the scores were around 0.98 for the original configuration, 0.99 - 1.00 range for the SMOTE+ENN and 0.95 - 0.98 for the SMOTE+Tomek. For the C15 dataset: No balancing: 0.96 - 0.97, SMOTE+ENN: 0.95 - 0.97, SMOTE+Tomek: 0.94 - 0.96.

Overall, with no retouching on the dataset the **Random Forest** classifier came up on top with accuracy of 0.9839 and AUC of 0.9845. With balancing treatment on the dataset **AdaBoost** was the top performer with near perfect accuracy and AUC of 0.9981 for SMOTE+ENN, and 0.9865 for SMOTE+Tomek. Table 6.2 contains in detail the scores achieved by the authors, as well as the scores in the reimplementation and the use of the models on the C15 dataset.

| Dataset | Algorithm | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|---|
| C17 | Logistic Regression | 0.9066 (0.9100) | 0.9400 (0.9174) | 0.9300 (0.9037) | 0.9300 (0.9092) | 0.8891 (0.9037) |
| C17 | Stochastic Gradient Descent | 0.8726 (0.9100) | 0.8700 (0.9288) | 0.8700 (0.9193) | 0.8700 (0.9236) | 0.8680 (0.9193) |
| C17 | Random Forest | **0.9839** (0.9669) | 0.9800 (0.9663) | 0.9800 (0.9686) | 0.9800 (0.9667) | **0.9845** (0.9349) |
| C17 | AdaBoost | 0.9823 (0.9695) | 0.9800 (0.9697) | 0.9800 (0.9695) | 0.9800 (0.9695) | 0.9823 (0.9961) |
| C17 | SMOTE+ENN - Logistic Regression | 0.9859 (0.9138) | 0.9900 (0.9164) | 0.9900 (0.9138) | 0.9900 (0.9132) | 0.9862 (0.9749) |
| C17 | SMOTE+ENN - Stochastic Gradient Descent | 0.9433 (0.9215) | 0.9500 (0.9240) | 0.9400 (0.9306) | 0.9400 (0.9374) | 0.9443 (0.9359) |
| C17 | SMOTE+ENN - Random Forest | 0.9937 (0.9953) | 0.9900 (0.9944) | 0.9900 (0.9972) | 0.99 (0.9888) | 0.9938 (0.9988) |
| C17 | SMOTE+ENN - AdaBoost | **0.9981** (0.9695) | 1.0000 (0.9697) | 1.0000 (0.9695) | 1.0000 (0.9695) | **0.9981** (0.9961) |
| C17 | SMOTE+Tomek - Logistic Regression | 0.9094 (0.9133) | 0.9200 (0.9165) | 0.9100 (0.9178) | 0.9100 (0.9182) | 0.9098 (0.9169) |
| C17 | SMOTE+Tomek - Stochastic Gradient Descent | 0.9039 (0.8918) | 0.9000 (0.8842) | 0.9000 (0.8842) | 0.9000 (0.8875) | 0.9031 (0.9050) |
| C17 | SMOTE+Tomek - Random Forest | 0.9859 (0.9888) | 0.9900 (0.9891) | 0.9900 (0.9899) | 0.9900 (0.9891) | 0.9859 (0.9846) |
| C17 | SMOTE+Tomek - AdaBoost | **0.9865** (0.9793) | 0.9512 (0.9525) | 0.9554 (0.9603) | 0.9525 (0.9603) | **0.9865** (0.9699) |
| C15 | Logistic Regression | (0.9130) | (0.8926) | (0.8988) | (0.8993) | (0.9074) |
| C15 | Stochastic Gradient Descent | (0.9202) | (0.9148) | (0.9220) | (0.9220) | (0.9280) |
| C15 | Random Forest | **(0.9849)** | (0.9841) | (0.9785) | (0.9682) | **(0.9836)** |
| C15 | AdaBoost | (0.9778) | (0.9762) | (0.9718) | (0.9669) | (0.9711) |
| C15 | SMOTE+ENN - Logistic Regression | (0.9836) | (0.9709) | (0.9737) | (0.9708) | (0.9899) |
| C15 | SMOTE+ENN - Stochastic Gradient Descent | (0.9388) | (0.9384) | (0.9438) | (0.9429) | (0.9542) |
| C15 | SMOTE+ENN - Random Forest | **(0.9911)** | (0.9723) | (0.9791) | (0.9737) | **(0.9902)** |
| C15 | SMOTE+ENN - AdaBoost | (0.9738) | (0.9638) | (0.9578) | (0.9610) | (0.9683) |
| C15 | SMOTE+Tomek - Logistic Regression | (0.8988) | (0.8914) | (0.8958) | (0.8912) | (0.9150) |
| C15 | SMOTE+Tomek - Stochastic Gradient Descent | (0.9099) | (0.8830) | (0.8981) | (0.8973) | (0.9003) |
| C15 | SMOTE+Tomek - Random Forest | **(0.9846)** | (0.9837) | (0.9783) | (0.9886) | **(0.9884)** |
| C15 | SMOTE+Tomek - AdaBoost | (0.9641) | (0.9576) | (0.9438) | (0.9429) | (0.9658) |

Table 6.2: Approach 1 Performance Metrics, Account-level features

## 6.3 Approach 2: Content-level classification

With this approach the focus is on the ability to be able to identify bots from a single piece of data, a tweet. With the limitations in creating datasets with reliably annotated bots, being able to identify this type of accounts by a single observation will be a turning point in tackling the adverse effects they have on online discourse.

While there are various, reasonably successful, works on bot detection based on tweet content, there are none that perform well and are based on using single observations. Using basic features such as the ones listed in the content-level column in table 6.1, with no data preparation such as balancing, brings accuracy that does not exceed the threshold of 80%.

### 6.3.1 Methods and Procedure

Many of the high performing techniques in Twitter bot detection utilize NLP - Natural Language Processing tools to identify patters and habitual elements in tweeting style. These techniques, however, are shown to be ineffective against advanced social bots [28] that more closely resemble human behavior. This lack of adaptability will also translate to reduced readiness in identifying bots from single data points. To bypass these limitations of conventional methods, an LSTM (Long Short Term Memory) model is used, which is a refined variant of RNNs - Recurrent Neural Networks. LSTMs are capable of learning long-term dependencies, especially in sequence prediction problems. They are designed with feedback connections that are capable of processing entire sequences of data. RNNs, and subsequently LSTMs, have been shown to be suitable for NLP work since their design allows recognizing relationships in sequential data [55]. In order to use an LSTM model it is necessary to convert tweet data to a suitable form.

### 6.3.2 Data preprocessing

To prepare tweet data for usage in an LSTM model, the first step is to tokenize the tweets and create a string of tokens for each one of them. These tokens are then parsed through, and special elements are replaced with **tags**. These special elements include *URLs, hashtags, numbers, usernames*. For example an identified URL is replaced by the tag $< url >$, a user mention is replaced by the tag $< user >$. This will effectively help the LSTM to learn the substructure of the language used by training on element/word co-occurrence. In our reimplementation this was done by utilizing a python package for regular expressions [1] and distinct patterns for hashtags, numbers, URLs and user mentions.

Similarly, the most common emojis encountered are replaced by tags. The emojis targeted are *smile, heart, lolface, angryface, neutralface*, which are converted to $< smile >$, $< heart >$, $< lolface >$, $< angryface >$, $< neutralface >$. In the reimplementation this was done with regular expressions and patterns of the Unicode representation of the emojis.

A more interesting step in data preprocessing is done by identifying two patterns in the tokens parsed: words written in *all caps* and words that contain more than two *repeated letters*. When one or both of these patterns are recognised, relevant tags are appended to the token. In example, the word 'POOL' will be replaced by two tokens, 'POOL' and $< allcaps >$. The misspelled word 'BRITTTLE' will be replaced by three tokens, 'BRITTTLE', $< allcaps >$ and $< consecutivechars >$. In our reimplementation this was done by creating two processes, one checks a token for consecutive characters, the second checks for fully capitalized tokens, references the first process for finding consecutive characters and then appends the appropriate tags to the tokens. All tokens are then converted to lower case.

[1] https://docs.python.org/3/library/re.html

### 6.3.3 Long Short Term Memory

These tokens are then transformed to an *embedding layer* by using a set of pre-trained GloVE - Global Vectors for Word Representation[2]. GloVe is an unsupervised learning algorithm for obtaining vector representations of words. The embedding layer in an LSTM is used to create word vectors for incoming words and it sits between the input layer and the LSTM layer. The embedding layer uses weights that can be initialized with random values, or can be initialized by using third-party word embeddings, like GloVE. Note that word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. Using this kind of third party embeddings is a form of *transfer learning*, since semantic information that was learned during the embedding process is transferred between words.

Using an LSTM does not come without shortcomings. Since a Twitter account's metadata are not sequential, meaning they are independent data points that make sense without informational context(unlike tweet text), they cannot be utilized by a conventional LSTM architecture. This means that if we want to push the performance of an LSTM a bit further by combining tweet data with account metada, that will not be possible with a traditional LSTM. To combat that, the authors have proposed a *contextual* LSTM architecture that can utilize account metadata as well. This architecture can be seen in figure 6.1 below. Notice the elements in *blue* color, which are the additions compared to a conventional LSTM.

The way this architecture works is by giving a supplementary input to the output layer. This supplementary/auxiliary input will contain the account metadata mentioned earlier. The **main input** is the tweet text, tokenized and processed into GloVE vectors, which are then fed into the LSTM model as described above. The 32-dimensions output vector produced by the LSTM is concatenated with the supplementary input, concatenation that is then given as an input to a two-layers(sized

[2]https://nlp.stanford.edu/projects/glove/

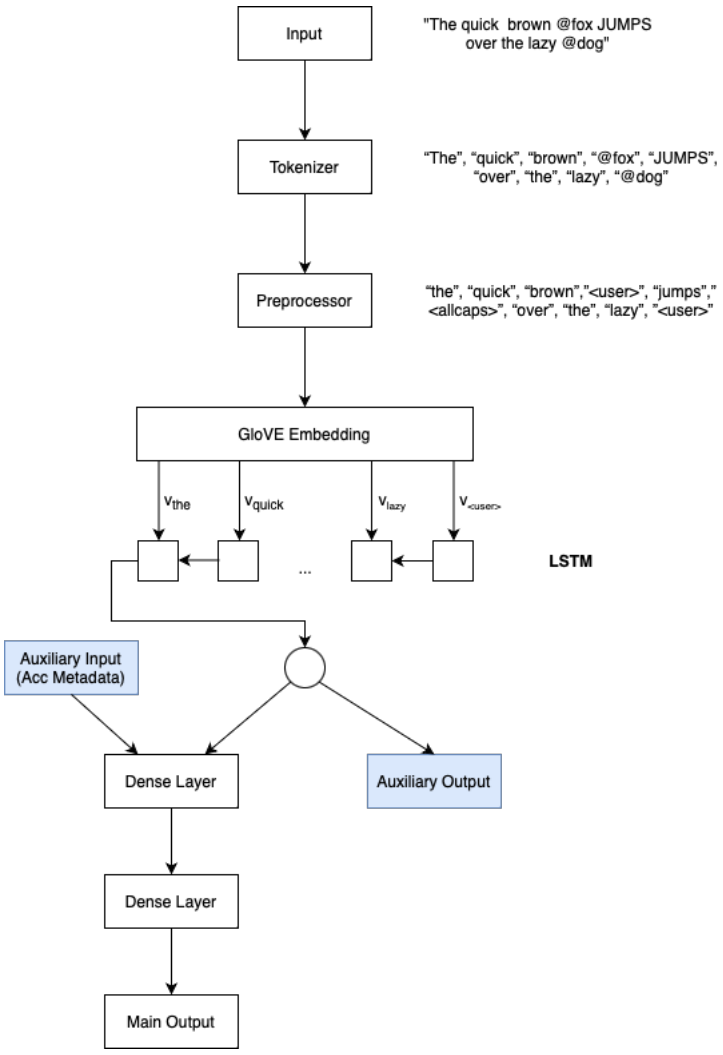


Figure 6.1: Contextual LSTM architecture

128 and 64 nodes) neural network with ReLU activation functions that produce the main output. An auxiliary output is introduced as a regularization mechanism whose target is the classification label as well.

This approach is tested in four batches: *a)* the simple, off-the-shelf classifiers, used on tweet metadata (see content-level column in 6.1) without data balancing, *b)* same classifiers combined with SMOTE+ENN, *c)* same classifiers combined with SMOTE+Tomek, *d)* the LSTM architecture with different configurations.

## 6.4 Results and Discussion

The first batch that used only a tweet's metadata, performed rather underwhelmingly, with accuracy that barely exceeds the 0.80 threshold. The second batch that utilized SMOTE combined with ENN sees a leap in performance with all models scoring in the 0.80 - 0.90 range. The third batch that utilized SMOTE combined with Tomek saw no such improvement, surprisingly it even had a small decline in performance with scores in the 0.76 - 0.77 range. Batch four, the LSTM, achieved the best scores overall, with accuracy and AUC around 0.95.

Our reimplementation mostly confirms the authors' findings on the performance improvements the LSTM brings and its capacity to identify bots based on single data points. Performance that can extend to other datasets as well, as our testing on the C15 dataset showed. All scores achieved in the reimplementation can be seen in table 6.3 along with the original scores achieved in the paper.

| Dataset | Algorithm | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|---|
| C17 | Logistic Regression | 0.8008 (0.8000) | 0.8000 (0.7686) | 0.8000 (0.7892) | 0.7900 (0.7962) | 0.7633 (0.7974) |
| C17 | Stochastic Gradient Descent | 0.7625 (0.7448) | 0.7600 (0.7322) | 0.7600 (0.7478) | 0.75 (0.7437) | 0.7191 (0.7496) |
| C17 | Random Forest | **0.8042** (0.7765) | 0.80 (0.7765) | 0.80 (0.7975) | 0.80 (0.7725) | **0.7765** (0.7975) |
| C17 | AdaBoost | 0.7991 (0.8099) | 0.80 (0.7694) | 0.80 (0.7651) | 0.79 (0.7683) | 0.7618 (0.7946) |
| C17 | SMOTE+ENN - Logistic Regression | 0.9188 (0.9161) | 0.92 (0.8849) | 0.92 (0.8715) | 0.92 (0.8859) | 0.8820 (0.8979) |
| C17 | SMOTE+ENN - Stochastic Gradient Descent | 0.8992 (0.8979) | 0.91 (0.8788) | 0.90 (0.8821) | 0.90 (0.8738) | 0.8860 (0.8816) |
| C17 | SMOTE+ENN - Random Forest | 0.9233 (0.9325 ) | 0.99 (0.9761) | 0.99 (0.9745) | 0.99 (0.9745) | 0.8806 (0.8736) |
| C17 | SMOTE+ENN - AdaBoost | **0.9234** (0.9151) | 0.93 (0.9244) | 0.92 (0.9108) | 0.93 (0.9138) | **0.9065** (0.8937) |
| C17 | SMOTE+Tomek - Logistic Regression | 0.7666 (0.7779) | 0.79 ( 0.7765) | 0.77 (0.7732) | 0.66 (0.6448) | 0.7667 (0.7697) |
| C17 | SMOTE+Tomek - Stochastic Gradient Descent | 0.7664 (0.7791) | 0.78 (0.7803) | 0.77 (0.7855) | 0.76 (0.7672) | 0.7664 (0.7727) |
| C17 | SMOTE+Tomek - Random Forest | **0.7747** (0.7675) | 0.79 (0.7637) | 0.77 (0.7716) | 0.77 (0.7693) | **0.7748** (0.7696) |
| C17 | SMOTE+Tomek - AdaBoost | 0.7715 (0.7893) | 0.79 (0.7732) | 0.77 (0.7753) | 0.77 (0.7754) | 0.7716 (0.7821) |
| C17 | LSTM + GloVE | **0.9553** (0.9600) | 0.9600 (0.9541) | 0.9600 (0.9553) | 0.9600 (0.9529) | **0.9567** (0.9646) |
| C15 | Logistic Regression | (0.7695) | (0.7966) | (0.7637) | (0.7795) | (0.7693) |
| C15 | Stochastic Gradient Descent | (0.7367) | (0.7549) | (0.7338) | (0.7419) | (0.7569) |
| C15 | Random Forest | **(0.7844)** | (0.7761) | (0.7878) | (0.7735) | **(0.7830)** |
| C15 | AdaBoost | (0.7661) | (0.7745) | (0.7761) | (0.7987) | (0.7600) |
| C15 | SMOTE+ENN - Logistic Regression | (0.8800) | (0.9151) | (0.8989) | (0.8894) | (0.9000) |
| C15 | SMOTE+ENN - Stochastic Gradient Descent | (0.9096) | (0.8947) | (0.8831) | (0.8804) | (0.9079) |
| C15 | SMOTE+ENN - Random Forest | (0.9148) | (0.9763) | (0.9733) | (0.9811) | (0.8924) |
| C15 | SMOTE+ENN - AdaBoost | **(0.9227)** | ( 0.9060) | (0.9147) | (0.9251) | **(0.9182)** |
| C15 | SMOTE+Tomek - Logistic Regression | (0.7823) | (0.7667) | (0.7819) | (0.6254) | (0.7705) |
| C15 | SMOTE+Tomek - Stochastic Gradient Descent | (0.7748) | (0.7694) | (0.7745) | ( 0.7693) | (0.7787) |
| C15 | SMOTE+Tomek - Random Forest | **(0.7839)** | (0.7885) | (0.7832) | (0.7864) | **(0.7897)** |
| C15 | SMOTE+Tomek - AdaBoost | (0.7776) | (0.7827) | (0.7878) | (0.7747) | (0.7790) |
| C15 | LSTM + GloVE | **(0.9447)** | (0.9581) | (0.9562) | (0.9357) | **(0.9722)** |

Table 6.3: Approach 2 Performance Metrics, Tweets' metadata

Overall, as can be seen in tables 6.2 and 6.3, in almost all cases the Random Forest

and the AdaBoost algorithms come up on top, performance-wise. Even in the second approach with LSTM dominating in all metrics, the next best scores are attributed to Random Forest and AdaBoost.

In terms of *minority oversampling* the combination of SMOTE and ENN is clearly the better choice for this task. Both in the original scores and in our reimplementation, and also in our application of the models on the C15 dataset, this combination clearly helped push classification performance higher, especially on the account-level bot detection task.

With tweet-level bot detection that performs well by examining a single data point, the LSTM works great with a GloVE embedding, which goes to show that his architecture can improve performance in the order of 5%, compared to AdaBoost in the SMOTE+ENN configuration, even by using a single tweet's text body.

As with previous papers examined we tried to identify the most important features in the datasets involved. Results can be seen in images 6.2 and 6.3. It appears the most important feature for account-level classification is the *favorites_count* feature, meaning the number of likes an account has given, followed by *friends_count* and *followers_count* - the number of accounts and account follows and the number of followers it has. For the content-level approach the number of mentions (*num_mentions*) seems to be the most important feature in the classification effort. The number of mentions feature contains the number of accounts mentioned in a tweet. It is followed by the number of hashtags(*num_hashtags*) used in tweet and the number of retweets a tweet has received(*retweet_count*).

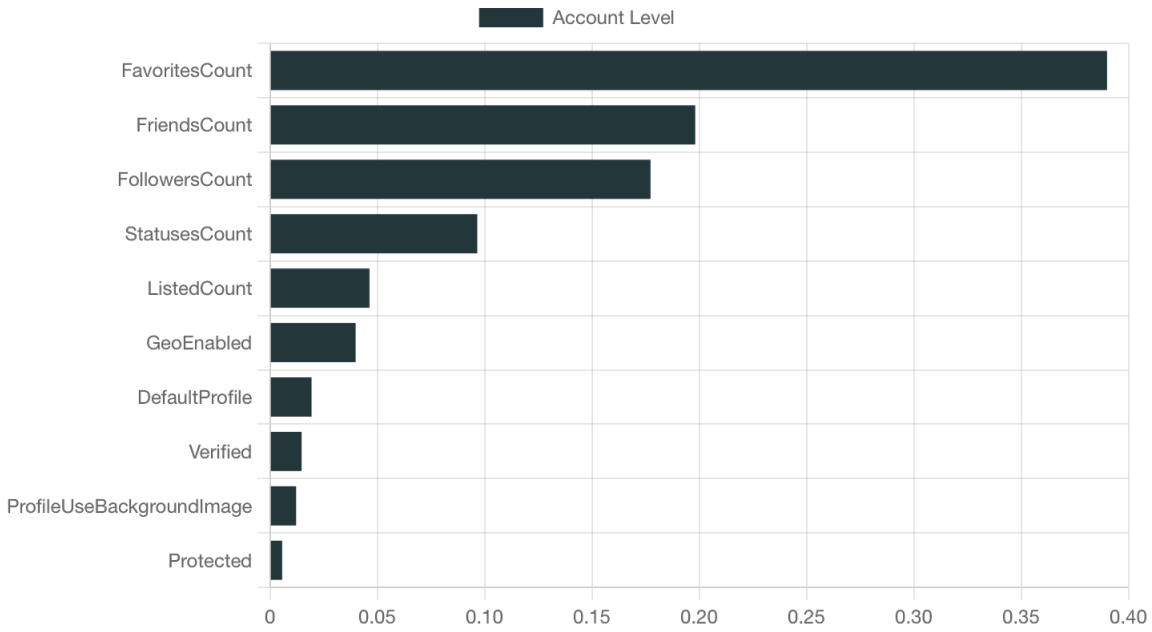


Figure 6.2: Model Optimization and Feature Minimization Feature Importance - Account level

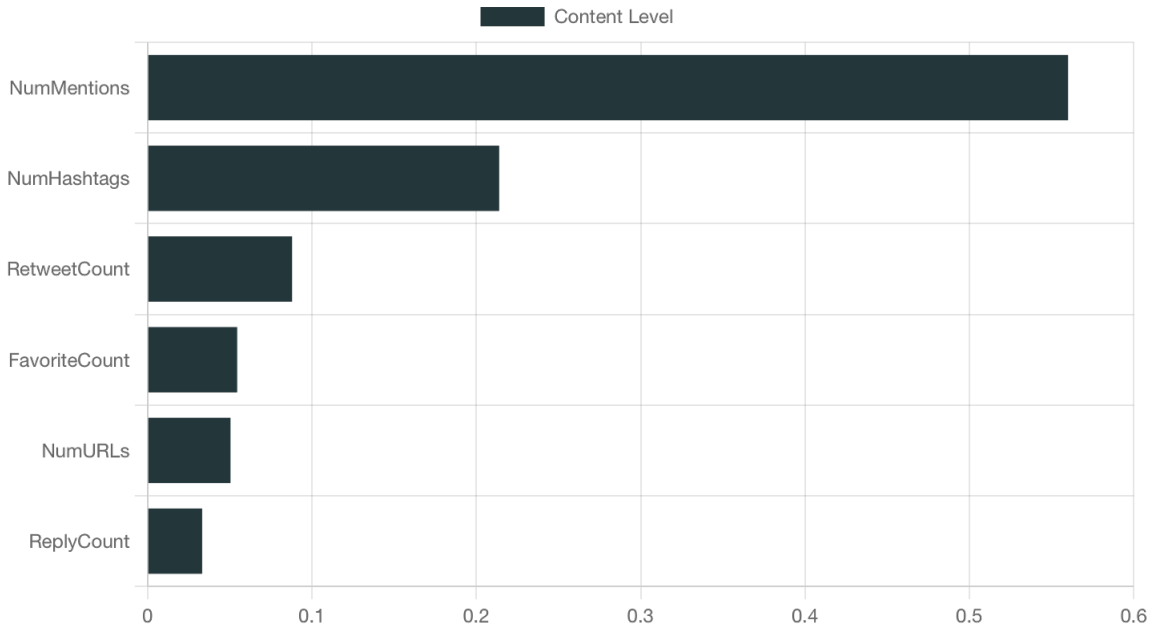


Figure 6.3: Model Optimization and Feature Minimization Feature Importance - Content level

# Chapter 7

# Conclusions, Recommendations, & Future Work

## 7.1 Conclusions

The purpose of this work was to find and analyze studies in the Twitter bot detection field, explain their purposes, examine the dataset they use, explain their methods and confirm their results and summarize the strengths and limitations of the studies. For this purpose we selected and examined four papers - the criteria of selection being how recently they were published, their claimed performance scores, how interesting their approaches seemed and also the practical benefit that they all used the same dataset.

The first paper, authored by researchers of the George Mason University [32] aimed to identify bots by using *sentiment* features that can be extracted from the text body of the tweets. The idea is based on the concepts of *confirmation bias* and the *backfire effect* and its purpose is to see whether these sentiment features can indeed benefit the classification effort and improve the accuracy of the models used, and also to examine whether this approach can be used to non-English tweets. The dataset used is C17. Five different methods are utilized, Random Forest, Feedforward Neural Network, Support Vector Machine, Logistic Regression. Of these methods, **Random Forest** and **FFNN** come up on top, on both the authors' implementation and our reimplementation. It is confirmed that there is an improvement when using the

enhanced feature set with the sentiment features, as can be seen at the first two rows of the table 3.2. Overall, this approach was proven to be effective in detecting bots with accuracy metrics hovering over 0.90. It was also proven that it can handle non-English tweets, since the same methods were tested with tweets in Dutch and produced similar results. It has to be noted thought that this aspect of the approach is heavily reliant on the libraries that can extract sentiment from tweet text, like *Textblob*. If the library does not support a specific language, the process comes to a halt. It is also worth noting that in the subset of 13 (7 produced + 6 readily available in C17) features that were used for training and testing the models, the five most important features were all sentiment-related 3.1.

The second paper is similar in nature to the first paper, in that the author aims for a *language-agnostic* approach to bot detection. Research goals include exploring the possibility of detecting on a, purely, account-level basis (without having to examine the content the accounts produce), whether that would facilitate detecting bots by completely avoiding language-specific features and finally to survey the quantity boundaries of an effective training dataset - meaning how large does a dataset need to be in order to achieve reasonable high performance scores in detecting bots. The dataset used in both the original author's implementation and our reimplementation is the C17. The methods used are for the most part the same ones used in other papers examined: Random Forest, Multi-layer Perceptron, Support Vector Machine, Adaboost and Logistic Regression. A total of 68 features are used, of which only 5 are readily found in C17, the rest are produced/calculated. The main mass of features produced revolve around interpreting the sentiment of *emojis* present in tweets. Of the methods used **AdaBoost**, combined with SMOTE+ENN for data balancing brought the best performance, with all metrics climbing over 0.98, and **Random Forest** with scores around 0.95. Our reimplementation of AdaBoost performed similarly to the model created by the author with scores in the impressive 0.99 - 1.0 range. For the rest of the methods there were no scores mentioned in the paper so

there is no room for comparison there. Overall, the research questions the author set out to answer can be considered answered. A dataset of about eight thousand manually annotated Twitter accounts has proven to be able to support high-performing machine learning models. Tweet content can indeed benefit the classification process, compared to only using account-level features. Account-level detection is indeed feasible, with methods employing only account-level features performing around 0.90 with almost all methods, which supports the argument that this kind of approach can benefit language-agnostic efforts in account-level bot detection. The aspect of identifying the sentiment of a tweet through the emojis present in the text can bring some benefits, as emoji sentiment features are amongst the most important features seen in figure 4.1. This cannot, however, be considered a robust aspect of this methodology, as many tweets do not employ emojis as a means of expression.

The third paper takes a more fine-grained approach to bot classification, by treating traditional spambots, social bots and fake followers as three separate categories. The reason for this are the recent advancements in AI and the automation landscape and how diversified the bots are nowadays, it might be immaterial to consider bots a unified group. So the author's goal is to find the best performing machine learning method for the task of bot detection and categorization. The dataset used for training and testing is the C17, and in our reimplementation we also used the C15 to test how well the models used will expand to new data. The methods used are Random Forest, Logistic Regression, an Artificial Neural Network and k-Nearest-Neighbors. The last two methods are mainly tested for diversity purposes by the author, there does not seem to be many studies around that have utilized them. 120 features are produced, and only the 100 most recent tweets per account were used. That will help balance the dataset out, since the *TweetsPerDay* feature showed that genuine users post on average 15 tweets per day - which is multiple times higher than any other account type. Of the methods used the **Random Forest** came up on top with accuracy of over 0.99, with Logistic Regression and the ANN being close seconds. Our reimplementation

produced results very similar to the ones claimed by the author, with all methods performing over the 0.97 threshold and Random Forest exceeding 0.99 in all metrics. Testing these models on the C15 brought equally great results, with metrics around 0.94 and in most cases over 0.97. So the goal the author set out to do is achieved, however it is also shown that the data used may be suffering from sampling bias. That would mean that some members of the intended population were not sampled with the same frequency as others. It is considered highly irregular that a real user would post 15 tweets on average daily, or that they would have tweets with more than 700 retweets. It is hard to say though if such cases are merely outliers or a fault in the sampling process, and the extent of the sampling bias - if it exists. Using the models on the C15 dataset indicates that the approach described in this paper can indeed be used on other datasets and bear impressive results, which does help suspend the concerns about the quality of the data. The point still stands though, that emphasis should be put on developing reliable and efficient **sampling procedures** in order to improve available data in both quality and quantity.

The final paper examined focuses on the tweet-level aspect of bot detection by proposing a methodology that will classify an account based on a single data point. They also suggest ways to overcome the lack of extensive labeled datasets by utilizing synthetic minority oversampling techniques to generate large labeled datasets. The dataset used is the C17, with a total of 16 features, all readily available in the dataset. The authors follow a forked approach: account-level classification with basic, low-complexity, off-the-shelf algorithms used on a small set of ten features that perform in the range of 0.94 to 0.98. They then use the same algorithms enhanced by using SMOTE minority oversampling, combined with ENN and Tomek Links undersampling techniques. These combinations push the performance to the 0.98 - 0.99 range, with **AdaBoost** and **Random Forest** being the top performers and proving that near-perfect scores can be achieved without complex machine learning architectures. The same round of tests is done on a set of six content-level features, which

will bring scores in the range of 0.77 - 0.92, with **AdaBoost** and **Random Forest** again at the top positions of performance. For the content-level approach, and after performing pre-processing on the data, they also employ an LSTM architecture, combined with a pre-trained GloVE model, with performance that is in the range of 0.9553 - 0.9643, for the various configurations that were tested. Similar results were achieved by the our reimplementation that was tested on both the C17 and the C15 datasets, and achieved scores in the 0.93 - 0.97 range. This goes to show that tweet-level bot detection can be performed with very high levels of accuracy, small number of features and datasets that are limited in size. Models that are based on small number of features have the extra benefit of being efficient and highly interpretable.

## 7.2 Future Work

Throughout all the papers examined the most prevalent difficulty in the research efforts was the scarcity of extensive datasets that will support future studies in bot detection and **sampling** procedures that will be able to keep up with the increasing levels of sophistication and complexity the modern bots continue to demonstrate.

Before looking into developing more robust sampling techniques, it would be useful to do more research in the bot classes that actually exist. The bot classes used in this project and the studies it examined are by no means exhaustive. Looking more into what demarcates one bot category from the other, as well as **quantifying** habitual elements per category would not only promote our understanding of what constitutes a specific type of bot, but will also help reduce bias in the sampling process. In that direction it would also be helpful to examine cases of category **overlap**, where one bot category's elements leak into another's. Establishing 'super' categories that contain more than one sub-categories, or looking into multi-label classification could help in that regard.

Another aspect of bot detection would be the **open-sourcing** of available tools and the interconnection with APIs/web services that will make identifying bots accessible

to less tech-savvy individuals. On a larger scale, open source tools like this could be used in mapping out a Twitter account's body of followers, or analyze a tweet's real impact by purging its stats from bot-sourced interactions, like retweets and likes.

# Bibliography


[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.

[2] J. G. Mark Jurkwitz, "Twitter is the go-to social media site for u.s. journalists, but not for the public," *https://www.pewresearch.org/fact-tank/2022/06/27/twitter-is-the-go-to-social-media-site-for-u-s-journalists-but-not-for-the-public/*, 2022.

[3] B. D. Loader and D. Mercea, "Networking democracy? social media innovations and participatory politics," *Information, Communication & Society*, 2011.

[4] J. B. Houston, J. Hawthorne, and e. a. Stanford A. Griffith, "Social media and disasters: A functional framework for social media use in disaster planning, response, and research," *Disasters*, 2014.

[5] U. Uras, "Turkey victims buried under rubble plead for help on social media," *https://www.aljazeera.com/news/2023/2/7/quake-victims-stuck-under-rubble-take-to-social-media-for-help*, 2023.

[6] B. Curtis, "Bots and misinformation spread on social media: Implications for covid-19," *Journal of Medicinal Internet Research*, 2021.

[7] M. T. Shahi G Dirkson A, "An exploratory study of covid-19 misinformation on twitter," *Online Social Networks and Media*, 2021.

[8] R. H. Oh O Kwon KH, "An exploration of social media in extreme events: Rumor theory and twitter during the haiti earthquake 2010," *International Conference on Information Systems 2020*, 2020.

[9] W. E, "How alex jones and infowars helped a florida man torment sandy hook families," *The New York Times*, 2019.

[10] Z. J. Wang B, "Rumor response, debunking response, and decision makings of misinformed twitter users during disasters," *Nat Hazards*, 2018.

[11] K. P. Gupta A Lamba H, "$1.00 per rt #bostonmarathon #prayforboston: Analyzing fake content on twitter," *APWG eCrime Researchers Summit*, 2013.

[12] S. K, M. J, O. M, A. P, and M. RM, "Rumors, false flags, and digital vigilantes: Misinformation on twitter after the 2013 boston marathon bombing," *iConference 2014 Proceedings*, 2014.

[13] J. F *et al.*, "Misinformation propagation in the age of twitter," *IEEE Annals of the History of Computing*, 2014.

[14] M. G. Hunt Allcott, "Social media and fake news in the 2016 election," *Journal of Economic Perspectives*, 2017.

[15] K. Wagner, "Musk's dispute with twitter over bots continues to dog deal," *https://www.bloomberg.com/news/articles/2022-07-07/twitter-reiterates-that-spam-bots-are-well-under-5-of-users?leadSource=uverify%20wall*, 2022.

[16] M. Newberg, "As many as 48 million twitter accounts aren't people, says study," *https://www.cnbc.com/2017/03/10/nearly-48-million-twitter-accounts-could-be-bots-says-study.html*, 2017.

[17] J. Nicas, "Why can't the social networks stop fake accounts?" *https://www.nytimes.com/2020/12/08/technology/why-cant-the-social-networks-stop-fake-accounts.html*, 2020.

[18] T. C. Guidelines, "Community guidelines enforcing report," *https://www.tiktok.com/transparency/en/community-guidelines-enforcement-2022-2/*, 2022.

[19] D. Chaffey, "Global social media statistics research summary 2023," *https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/*, 2023.

[20] C. T. Tony Romm, "Facebook and twitter testified before congress. conservative conspiracy theorists lurked behind them.," *https://www.washingtonpost.com/technology/2018/09/05/facebook-twitter-sandberg-dorsey-congress-tech-hearings/*, 2018.

[21] A. S. Gabe O'Connor, "How russian twitter bots pumped out fake news during the 2016 election," *https://www.npr.org/sections/alltechconsidered/2017/04/03/522503844/how-russian-twitter-bots-pumped-out-fake-news-during-the-2016-election*, 2017.

[22] D. H. Yoel Roth, "How twitter is fighting spam and malicious automation," *https://blog.twitter.com/official/en_us/topics/company/2018/how-twitter-is-fighting-spam-and-malicious-automation.html*, 2018.

[23] N. S. Galen Stocking, "Social media bots draw public's attention and concern," *https://www.pewresearch.org/journalism/2018/10/15/social-media-bots-draw-publics-attention-and-concern/*, 2018.

[24] D. G. Robert Gorwa, "Unpacking the social media bot: A typology to guide research and policy," *https://arxiv.org/abs/1801.06863*, 2018.

[25] S. M. DiMolfetta, "Twitter's struggles narrow path to revenue growth as musk charts new direction," *https://www.spglobal.com/marketintelligence/en/news-insights/latest-news-headlines/twitter-s-struggles-narrow-path-to-revenue-growth-as-musk-charts-new-direction-73040368*, 2022.

[26] J. Jammot, "Twitter aims to diversify beyond advertising, but can it be done?" *https://techxplore.com/news/2022-11-twitter-aims-diversify-advertising.html*, 2022.

[27] A.-M. Alcántara, "Elon musk's alarm over twitter bots isn't matched by advertisers," *https://www.wsj.com/articles/elon-musks-alarm-over-twitter-bots-isnt-matched-by-advertisers-11652977221*, 2022.

[28] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," *Proceedings of the 26th International Conference on World Wide Web Companion, https://arxiv.org/abs/1701.03017*, 2017.

[29] J. Novotny, "Twitter bot detection & categorization - a comparative study of machine learning methods," *Master's Thesis in Statistics*, 2019.

[30] 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)*, 2010.

[31] O. Varol, E. Ferrara, C. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," *11th International AAAI Conference on Web and Social Media (ICWSM)*, 2017.

[32] O. U. Maryam Heidari James H Jr Jones, "An empirical study of machine learning algorithms for social media bot detection," *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2021.

[33] J. Knauth, "Language-agnostic twitter-bot detection," *International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, 2019.

[34] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Fame for sale: Efficient detection of fake twitter followers," *Decision Support Systems, 80, 56-71, 2015, https://arxiv.org/abs/1509.04098*, 2015.

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

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

[37] M. Workman, "An empirical study of social media exchanges about a controversial topic: Confirmation bias and participant characteristics," *Social Media in Society, pp. 381-400*, 2018.

[38] T. Wood and E. Porter, "The elusive backfire effect: Mass attitudes' steadfast factual adherence," *Springer ScienceBusiness Media, LLC, part of Springer Nature 2018*, 2018.

[39] M. Cinelli, G. D. F. Morales, A. Galeazzi, and M. Starnini, "The echo chamber effect on social media," *PNAS, Vol. 118 — No. 9*, 2020.

[40] R. Battur and N. Yaligar, "Twitter bot detection using machine learning algorithms," *International Journal of Science and Research (IJSR)*, 2018.

[41] "Matthews correlation coefficient: When to use it and when to avoid it," *https://towardsdatascience.com/matthews-correlation-coefficient-when-to-use-it-and-when-to-avoid-it-310b3c923f7e*, 2020.

[42] E. March, "Psychopathy, sadism, empathy, and the motivation to cause harm: New evidence confirms malevolent nature of the internet troll," *Official Journal of the International Society for the Study of Individual Differences (ISSID)*, 2019.

[43] S. Ray, "Musk issues new rule for parody twitter accounts after 'verified' impersonators cause chaos," *https://www.forbes.com/sites/siladityaray/2022/11/11/musk-issues-new-rule-for-parody-twitter-accounts-after-verified-impersonators-cause-chaos/?sh=5188718e7173*, 2022.

[44] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady 10:707*, 1966.

[45] T. H. Center, "How to get the blue checkmark on twitter," *https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts*, 2023.

[46] J. Fernquist, L. Kaati, and R. Schroeder, "Political bots and the swedish general election," *In IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2018.

[47] J. Clover, "Twitter officially bans all third-party apps," *https://www.macrumors.com/2023/01/20/twitter-bans-third-party-apps/*, 2023.

[48] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning," *Springer, New York, 2nd edition*, 2009.

[49] G. Dreyfus, "Modeling with neural networks: Principles and model design methodology," *Neural Networks - Methodology and Applications*, 2005.

[50] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "Botornot: A system to evaluate social bots," *Proceedings of the 25th International Conference Companion on World Wide Web. International World Wide Web Conferences Steering Committee,*, 2016.

[51] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *ACM Journal, vol.59*, 2016.

[52] E. Ferrara, "Disinformation and social bot operations in the run up to the 2017 french presidential election," 2017.

[53] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics 2, 3*, 1972.

[54] I. Tomek, "Two modifications of cnn," *IEEE Trans. Systems, Man and Cybernetics 6*, 1976.

[55] Y. Goldberg, "A primer on neural network models for natural language processing," *JAIR - Journal of Artificial Intelligence Research*, 2016.

[56] S. V. Sinan Aral Deb Roy, "On twitter, false news travels faster than true stories," *https://news.mit.edu/2018/study-twitter-false-news-travels-faster-true-stories-0308*, 2018.

[57] L. Gorosch, "Prof. cavazos's latest report: Digital ad fraud costs will rise to $35b globally this year," *https://cheq.ai/blog/the-massive-indirect-cost-of-digital-ad-fraud/*, 2020.

[58] S. Bennet, "67% of taylor swift's twitter followers are bots, says study," *https://www.adweek.com/performance-marketing/twitter-bots-problem/*, 2015.

[59] IIT-CNR, "The fake project twitter account," *https://twitter.com/thefakeproject*, 2012.

[60] G. E. Batista, R. C. Prati, and M.-C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter 6(1):20-29*, 2004.

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

[62] E. F. Sneha Kudugunta, "Deep neural networks for bot detection," *Information Sciences, Volume 467, October 2018, Pages 312-322, 2018*, 2018.