Bot Detective: Explainable Bot Detection in Twitter

Author:
Maria Kouvela [AEM: 9]
mvkouvela@csd.auth.com

Supervisors:
Professor Athena Vakali
PhD Candidate Ilias Dimitriadis

A thesis submitted in fulfillment of the requirements for the degree of Master of Data & Web Science in the

Aristotle University of Thessaloniki
Faculty of Sciences
School of Informatics

February 2020
Acknowledgments

Attending a master’s degree program in Data and Web Science and writing this thesis must be some of the hardest things I have ever accomplished. In my point of view the completion of this thesis was a teamwork, and I finally have the opportunity to thank all the people who supported me.

I would first like to thank my thesis supervisors PhD Candidate Ilias Dimitriadis and Professor Athena Vakali for our excellent cooperation, for their guidance and the trust that they showed in me.

I would also like to thank Napoleon, my beloved partner, who was there for me throughout all my ups and downs, always willing to help me, who tolerated me, and taught me not to give up.

Finally, I must express my very profound gratitude to my family for their endless love and support, my mother Elena, who provided me with everything at any cost and made me who I am today, my father Vasilis, who is always looking after me, my sister Alexia, the funniest and most loving friend in my life, my brother Alexandros, who helps me stay positive and my sister Manto, who is always looking up to me. I also want to thank my grandmother Mary for all her homecooked meals and my grandfather Alekos for his encouragement. Of course, I could not forget Pavlos, who has taught me all sorts of things and who I am so grateful to have in my family.

Thessaloniki, February 2020,

Maria.
Bot Detective: Explainable

Bot Detection in Twitter

Abstract

Bot accounts exploit social media networks in order to fulfill their malicious purposes; phishing, influencing public opinion and spreading hate speech, to name a few. Twitter, with more than 330 million active users, is one of the social networks that suffer the most from bot attacks. The research community is getting more and more involved in this subject ever since social networks started attracting more audience. The approach using Machine Learning has proven to be the most effective, since it helps in discovering of patterns that exist in bot-like behavior. A lot of work in the literature focuses on extracting user and content information from identified Twitter accounts and then training a classification model to use in predicting the likelihood of an account being a bot. The issue that lies there is that the reasons behind how and why a Machine Learning model reaches to decisions internally are not clear.

The present thesis introduces a novel methodology in order to explain the decisions made from Machine Learning models about their results of identifying accounts in Twitter. To that end, first, a large in volume labelled dataset was created using the Twitter API, that lead to the development of an explainable regression model, able to produce accurate results. To assist the public, in the context of this thesis, a web application was also developed, which makes real-time predictions about the status of Twitter users accompanied with the reasons behind those predictions. The contribution of this work will hopefully support not only the research community, but users of Twitter as well, that can use the developed tool as their ally in identifying bots in their everyday interactions in the network.
Bot Detective: Εντοπισμός κακόβουλων και αυτοματοποιημένων λογαριασμών στο Twitter με επεξηγηματικό χαρακτήρα

Περίληψη

Οι αυτοματοποιημένοι λογαριασμοί, γνωστοί και ως bot εκμεταλλεύονται τα δίκτυα κοινωνικών μέσων ενημέρωσης προκειμένου να εκπληρώσουν τους κακόβουλους σκοπούς τους: η εξαπάτηση, ο επηρεασμός της κοινής γνώμη και η ομιλία μίσους είναι μερικές από τις πράξεις τους. Το Twitter, με περισσότερους από 330 εκατομμύρια ενεργούς χρήστες, είναι ένα από τα κοινωνικά δίκτυα που υποφέρουν περισσότερο από επιθέσεις των bot. Η ερευνητική κοινότητα συμμετέχει ολόκληρη σε αυτό το θέμα από τότε που τα κοινωνικά δίκτυα άρχισαν να προσελκύουν περισσότερο κοινό. Η προσέγγιση που χρησιμοποιεί τη μηχανική μάθηση έχει αποδειχθεί ότι είναι η πιο αποτελεσματική, καθώς βοηθά στην ανακάλυψη των μοτίβων που υπάρχουν στη συμπεριφορά των bot. Πολλές εργασίες στη βιβλιογραφία εστιάζουν στην εξαγωγή πληροφοριών σχετικά με τους χρήστες και το περιεχόμενο που αναρτούν, από αναγνωρισμένους λογαριασμούς του Twitter και στη συνέχεια στην εκπαίδευση ενός μοντέλου ταξινόμησης ώστε να προβλέψει τη πιθανότητας ένας λογαριασμό να είναι bot. Το ζήτημα που εγείρεται είναι ότι οι λόγοι πίσω από το πώς και γιατί ένα μοντέλο μηχανικής μάθησης φθάνει στις αποφάσεις του εσωτερικά δεν είναι σαφείς.

Η παρούσα εργασία εισάγει μια νέα μεθοδολογία με σκοπό να εξηγήσει τις αποφάσεις που λαμβάνονται από μοντέλα μηχανικής μάθησης σχετικά με τα αποτελέσματα τους στην αναγνώριση λογαριασμών του Twitter. Όπως έχει παρατηρηθεί μέσα από διάφορες εργασίες της βιβλιογραφίας στο θέμα, οι λογαριασμοί bots τείνουν να υποδύονται πραγματικούς χρήστες μέσα από το περιεχόμενο των tweets που αναρτούν αλλά και τις πληροφορίες του προφίλ τους, γεγονός το οποίο κάνει το πρόβλημα πιο δύσκολο. Παρατηρώντας τις συμπεριφορές τους και συγκρίνοντάς τες με αυτές πραγματικών χρηστών μπορεί να οδηγήσει σε στοιχεία διάκρισης μεταξύ τους.

Για το σκοπό αυτό, αρχικά, δημιουργήθηκε ένα μεγάλο σύνολο δεδομένων, χρησιμοποιώντας το API του Twitter. Επιλέχθηκαν διάφορα από τα διακριτικά στοιχεία των προφίλ ως κύρια χαρακτηριστικά για το σύνολο δεδομένων. Στη συνέχεια αναπτύχθηκε ένα επεξηγηματικό μοντέλο παλινδρόμησης, ικανό να παράγει ακριβή αποτελέσματα. Το τελικό σύστημα πρόβλεψης που κατασκευάστηκε δέχεται ως είσοδο το @username ενός λογαριασμού του Twitter και
συλλέγει πληροφορίες για τα τελευταία 20 tweets που αναρτήθηκαν από το λογαριασμό αυτό. Από τις πληροφορίες αυτές εξάγει τα κύρια χαρακτηριστικά και κατασκευάζει το νέο αταξινόμητο σύνολο δεδομένων. Στη συνέχεια το νέο σύνολο δεδομένων εισάγεται στο εκπαιδευμένο μοντέλο και παράγονται οι προβλέψεις για το λογαριασμό. Το σκορ που αποδίδει το μοντέλο στο λογαριασμό σε κλίμακα [0-5], όπου 0 είναι πραγματικός χρήστης και 5 είναι bot, συνοδεύεται και από μια σειρά επεξηγήσεων γύρω από την πρόβλεψη του μοντέλου, οι οποίες κατασκευάστηκαν με εξαγωγή κανόνων σε φυσική γλώσσα ώστε να είναι κατανοητές από κάθε χρήστη, ανεξαρτήτως τεχνικού υποβάθρου.

Για να εξυπηρετηθεί το κοινό, στο πλαίσιο αυτής της εργασίας, αναπτύχθηκε επίσης μια web εφαρμογή, η οποία κάνει προβλέψεις σε πραγματικό χρόνο σχετικά με την κατάσταση χρηστών του Twitter συνοδευόμενη από τους λόγους πίσω από αυτές τις προβλέψεις. Επίσης, δίνεται η δυνατότητα στους χρήστες του εργαλείου αυτού να συμβάλλουν στην αξιολόγηση και βελτίωση του, παρέχοντας τα σχόλιά τους, τα οποία συμβάλλουν στην κατάσταση χρηστών του Twitter, ενώ μπορούν να χρησιμοποιηθούν για να αναπτυχθεί εκ νέου εκπαίδευση του μοντέλου στο νέο δεδομένα.

Η συμβολή αυτού του έργου αναμένεται να υποστηρίζει, όχι μόνο την ερευνητική κοινότητα, αλλά και τους χρήστες του Twitter, οι οποίοι μπορούν να χρησιμοποιήσουν το αναπτυγμένο εργαλείο ως σύμμαχο τους στον εντοπισμό bots στις καθημερινές τους αλληλεπιδράσεις στο δίκτυο.
Table of Contents

Acknowledgments ........................................................................................................... 2

Abstract .......................................................................................................................... 4

Περιλήψη ......................................................................................................................... 5

Table of Contents ............................................................................................................. 4

List of Figures .................................................................................................................. 6

List of Tables ..................................................................................................................... 6

Chapter 1 Introduction ..................................................................................................... 7
  1.1 Background on Social Media Networks and Spam .................................................... 7
    1.1.1 Social Media Networks ....................................................................................... 7
    1.1.2 Twitter Platform and Its Features ..................................................................... 7
    1.1.3 Spam .................................................................................................................. 8
    1.1.4 Spam Behavior on Twitter ............................................................................... 9
    1.1.5 Malicious Spammers and Bots ........................................................................ 9
  1.2 Subject of master’s Thesis ....................................................................................... 11
    1.2.1 Problem Statement ......................................................................................... 11
    1.2.2 Twitter Rules and Policies ............................................................................. 12
    1.2.3 Penalties for Spam Behavior ......................................................................... 13
    1.2.4 Regulations for Bot Accounts ........................................................................ 14
    1.2.5 European Union General Data Protection Regulation .................................... 14
    1.2.6 Approach to the Problem of Spam ................................................................ 15
  1.3 Thesis Contribution and Challenges ....................................................................... 16
    1.3.1 Selection of the Twitter Social Media Platform .............................................. 16
    1.3.2 Explainable AI ............................................................................................... 16
    1.3.3 Contribution .................................................................................................... 17
  1.4 Document Structure ............................................................................................... 18

Chapter 2 Related Work ................................................................................................. 19
  2.1 Methods ................................................................................................................... 19
    2.1.1 Machine Learning Based Methods .................................................................. 21
    2.1.2 Graph Based Methods ................................................................................... 23
    2.1.3 Hybrid Based Methods .................................................................................. 24
    2.1.4 Categorization of Literature ......................................................................... 25
    2.1.5 Categorization of Features .......................................................................... 27
    2.1.6 Explainable Artificial Intelligence .................................................................. 32
  2.2 Comparison and Opportunities ............................................................................... 32

Chapter 3 Methodology ................................................................................................. 34
  3.1 System Requirements and Architecture .................................................................. 34
  3.2 User Interface .......................................................................................................... 35
  3.3 API Server ................................................................................................................. 36
  3.4 Machine Learning Model ....................................................................................... 36
    3.4.1 Data Collection ............................................................................................. 36
    3.4.2 Dataset Creation ............................................................................................ 36
    3.4.3 Regressor Algorithm ....................................................................................... 37
  3.5 Conversion of Results to Human Readable Explanations ..................................... 37
Chapter 4 Implementation ................................................................. 38
  4.1 Implementation ............................................................................. 38
  4.1.1 Twitter API and Data Collection ................................................. 38
  4.1.2 Botometer API ............................................................................ 39
  4.1.3 Data Storage .............................................................................. 41
  4.1.4 Programming Language and Libraries ........................................ 42
  4.1.5 Feature Extraction and Dataset ................................................... 43
  4.1.6 Model ........................................................................................ 45
  4.1.7 Explainable Model ...................................................................... 46
  4.1.8 Server Development ................................................................. 47
  4.1.9 User Interface ............................................................................ 48
  4.2 Usage ............................................................................................ 48

Chapter 5 Experimentation and discussion ........................................... 53
  5.1 Feature Selection ............................................................................ 53
  5.2 Model Selection and Hyperparameters Optimization ......................... 53
  5.3 Comparison of Explainable Models ................................................. 57

Chapter 6 Conclusions and Future Work .............................................. 59
  6.1 Conclusions ................................................................................... 59
  6.2 Future Work .................................................................................. 60

References .......................................................................................... 61
List of Figures

FIGURE 1: TWITTER LOGO ........................................................................................................... 8
FIGURE 2: EXAMPLE OF A TWEET ............................................................................................. 8
FIGURE 3: EXAMPLE OF SPAM EMAIL [3] .................................................................................. 10
FIGURE 4: WARNING FOR FLAGGED URLs ................................................................................. 11
FIGURE 5: POLITICAL BOTS ON TWITTER [6] .......................................................................... 12
FIGURE 6: TWEET FROM @TWITTERSAFETY .......................................................................... 15
FIGURE 7: ANNOUNCEMENT OF @TWITTERSAFETY ON UPDATING THE FOLLOWING LIMIT .... 15
FIGURE 8: MACHINE LEARNING AS A BLACK BOX [14] ......................................................... 17
FIGURE 9: SYSTEM STRUCTURE ............................................................................................... 34
FIGURE 10: PREDICTION PROCESS .......................................................................................... 35
FIGURE 11: EXAMPLE OF TWEET IN JSON FORMAT ............................................................... 39
FIGURE 12: EXAMPLE OF BOTOMETER API RESPONSE ......................................................... 40
FIGURE 13: LANDING SCREEN OF BOT DETECTIVE .................................................................. 49
FIGURE 14: LOGGING IN WITH TWITTER ................................................................................. 49
FIGURE 15: EXAMPLE OF USAGE ............................................................................................. 50
FIGURE 16: FEEDBACK SCREEN ............................................................................................... 51
FIGURE 17: ACCOUNT DETAILS ................................................................................................. 52
FIGURE 18: RANDOM FOREST PREDICTION RESULTS ............................................................ 55
FIGURE 19: GRADIENT BOOSTING PREDICTION RESULTS ....................................................... 56
FIGURE 20: STACKING PREDICTION RESULTS ........................................................................ 57

List of Tables

TABLE 1: CATEGORIZATION OF LITERATURE ......................................................................... 27
TABLE 2: CATEGORIZATION OF FEATURES .............................................................................. 31
Chapter 1
Introduction

This Chapter attempts to familiarize the readers with the issue of spam in social media networks, which was the motivation for the current thesis. Firstly, an introduction to social media networks and spam is provided. Continuing, the main aspects of the issue are addressed, and the research approach is described. Lastly, the contribution and novelty of this work are presented at the end of the Chapter.

1.1 Background on Social Media Networks and Spam

1.1.1 Social Media Networks

Social media networks have quickly become from just a hobby to a big and necessary part of our lives, allowing us to interact with each other and share media across our connections. According to Wikipedia, a social media network is an online platform which people use to build social networks or social relationships with other people who share similar personal or career interests, activities, backgrounds or real-life connections [1]. Such platforms are usually free to join and easy to use, available through their websites and their mobile/tablet applications. Social networking services vary in format and number of features. One of the most popular social media platforms with millions of active users is of course Twitter.

1.1.2 Twitter Platform and Its Features

Twitter is an online microblogging platform. Users can create their profile, a personalized feed, and post tweets, texts up to 280 characters long that can optionally contain media such as images or videos, as in the case of a blog. There is no restriction in the subject one can tweet about, as long as the tweet follows Twitter’s policies, which we will review below. Users establish connections with other users in the Twitter network when they follow or get followed by other profiles. Followers are the accounts that follow a particular account and followees, or sometimes encountered as “followings”, are the accounts that are followed by a particular account. By following a profile, the user “subscribes” to view the content that this profile posts on demand. Unlike other social media platforms, the following functionality is not bidirectional, so the concept of friendship does not really exist. However, if two users follow each other we tend to use the term friends to refer to the connected accounts for convenience.

1 [https://twitter.com/](https://twitter.com/)
Other than the concept of tweet, Twitter provides some extra features. A **retweet** is the result of a tweet owned by an account that another account can repost. Additionally, accounts can **reply** or **mention** other accounts in their tweets by adding the @username of that account inside the text of the tweet. Those types of tweets are known as replies and mentions respectively. Users can **like** or **favorite** a tweet, to show appreciation for a tweet. A user might use **URLs** (Uniform Resource Locator), **links to external web pages**, or **hashtags**, the “#” symbol followed by a word or phrase that usually matches the content of the tweet. For example, if someone tweets about a football game it could contain the hashtag “#sports” or “#football”. Twitter uses hashtags that users include in their tweets to create and rank trending topics.

1.1.3 Spam

A common issue in the online world is spam. **Spam** could generally be described as **a form of intentional attack to users**, whether its type is malicious or not. Spam is not a new phenomenon, it was first observed in the early days of the web, in the form of email spam where unsolicited messages were sent in bulk to users’ inboxes. Undeniably spam of any kind is useless and annoying to their receivers, but in many cases, it is harmful too. Most spam emails aim in promoting advertisements, while others intend to harm their victims, by embedding malicious software or links to unsafe websites inside their sent content.
1.1.4 Spam Behavior on Twitter

As social media started growing, it did not take long for spammers, the users who initiate spam, to find their way in and expand their activities. The registration process for someone to join Twitter is fairly easy and does not require many input fields from the user. As such, a user can create multiple Twitter accounts by providing different inputs like emails. Spammers usually leverage this to manage multiple accounts in order to expand their network and increase the possibility of being noticed by other users in the platform.

Inside the Twitter platform exist different types of spammers. Some of them are theoretically harmless. Some spammers’ main purpose is to promote a product or a service to the public by featuring advertisements to increase their profits from potential shoppers. Some will just post the same tweets over and over for no particular reason. Others wish to expand their number of followers and gain popularity by randomly and constantly sending following requests to other Twitter accounts, expecting that they will follow them back. They might even include @usernames in their tweets to attract their victims’ attention.

1.1.5 Malicious Spammers and Bots

Additionally, there exist spammers of the malicious type and behavior. A characteristic of this type of spammer is that they usually include URLs inside their tweets, to transfer users to their websites and acquire more audience. These URLs often link to harmful websites both for the users and their hardware. A few years ago, a user was able to identify the content of the attached URLs to external websites, without having to click on them, but just by viewing the structure of the URL strings. Spammers would attempt to disguise their URLs to appear as URLs that are commonly known to users and that the users trust. A very simple example could be replacing the lowercase character “l” (“L”) into the capital character “I” (“i”) to use in a URL, for instance “www.google.com” and make it appear like “www.googIe.com” in users’ eyes.

Spammers imitating legitimate companies would send out fake emails to conceal their real identity. The nickname for this type of hacker attack is a "phishing email" because the email is actually bait [2]. A very common pattern was sending fake emails from PayPal, a company that operates online payments. With the majority of scam emails the tell-tale signs are the odd characters or random numbers in the sender’s email address, the “Dear Customer” opening and the sense of urgency it creates. Fraudsters prey on customers acting in panic. The fear that someone has hacked your account based on what the email is saying will cause people to click that fake link without thinking [3]. An example of such an email can be viewed in Figure 3 below:
However, over time users started getting aware of this kind of attack from spammers and got more vigilant. On the other hand, spammers have the tendency of finding ways to overcome such obstacles as soon as they notice that their actions are not effective. Spammers got, soon enough, aware of a service that was developed to help shorten URLs to fit inside the tweet limit of 280 characters. By using a URL shortener a URL can be made substantially shorter and still direct to the original page. One of the first well known URL shortening services, is TinyURL\(^2\), that was launched in 2002. A URL produced by a URL shortening service does not contain any information about the structure of the website, like the path or the hostname, so users cannot identify the content of the website that it links to. Later on, Twitter realized that spammers where abusing the platform by posting shortened URLs linking to dangerous websites for its users, and incorporated its own shortening service, t.co\(^3\), to help users minimize the characters of their attached URLs in tweets and direct messages, whilst protecting users from visiting malicious sites that engage in spreading malware, phishing attacks, and other harmful activity. A link converted by

\[^2\]https://tinyurl.com/
Twitter’s link service is checked against a list of potentially dangerous sites, and users are warned with a warning message when clicking on potentially harmful URLs.

![Warning: this link may be unsafe](image)

**Figure 4: Warning for flagged URLs**

Twitter provides users with some automation tools allowing them to automatically post tweets through programming scripts, that can be customized, by providing beforehand content to be tweeted, to fill the needs of the users. Along with others, another functionality that is provided from Twitter is auto-searching and auto-following users. These options have resulted in the creation of another kind of users which are called bot users or bots. It is rather common for spammers to adopt this kind of behavior to make their work much easier, by taking advantage of this auto-posting functionality to schedule their potential tweets and by randomly searching and following other users to eventually expand their network.

### 1.2 Subject of master’s Thesis

#### 1.2.1 Problem Statement

Spam in general is not desirable by the users. Email spam unexpectedly and without any consent made its way into users’ inboxes, forcing them to unwillingly open annoying and harmful messages. Likewise, in Twitter, users can not enjoy the experience of Twitter, since they are constantly interrupted from their everyday activities. Influencing public opinion, amplifying a message, catfishing and hate speech are just a few ways spam accounts can be used for malicious purposes [4].

What is an alerting example, is that Twitter bots may have altered the outcome of two of the world’s most consequential elections in recent years, according to an economic study. Automated tweeting played a small but potentially decisive role in the 2016 Brexit vote and Donald Trump’s presidential victory, the National Bureau of Economic Research working paper showed. Their rough calculations suggest bots
added 1.76 percentage point to the pro-“leave” vote share as Britain weighed whether to remain in the European Union and may explain 3.23 percentage points of the actual vote for Trump in the U.S. presidential race [5].

![Figure 5: Political Bots on Twitter [6]](image)

The main issue is that there is no official way for users to know if an account is spam or not, and thus they might follow spam accounts and invite them to act uninterruptedly in their Twitter timeline. That is because spammers attempt to impersonate normal non-spam accounts, and they have learned how to quickly adapt their behavior to avoid being detected. This camouflage has allowed spammers to stay active on Twitter for long enough, some even for years. The automation that Twitter provides to facilitate its users, has been taken advantage of by companies, offering tools that allow their customers to automatically follow a large number of users with little effort. This works as a growth tactic because some people will follow back out of courtesy, without realizing they have followed a bot. The companies also offer tools to mass unfollow Twitter accounts of those who did not return the favor by following the bot back. Other automated tools were often provided, as well, like ones for creating auto-DMs (Direct Messages), for example [7].

1.2.2 Twitter Rules and Policies

Twitter provides extended guidelines about what is considered as spam in the platform [8]. Platform manipulation includes:

- commercially motivated spam, that typically aims to drive traffic or attention from a conversation on Twitter to accounts, websites, products, services, or initiatives
- inauthentic engagements, that attempt to make accounts or content appear more popular or active than they are
- coordinated activity, that attempts to artificially influence conversations through the use of multiple accounts, fake accounts, automation and/or scripting.
1.2.3 Penalties for Spam Behavior

Twitter wants the platform to be a place where people can make human connections, find reliable information, and express themselves freely and safely as stated in their Rules and Policies. To make that possible, Twitter relies on advanced technology to fight spam, malicious automation, and other forms of platform manipulation. Using custom-built tools, they attempt to identify such behaviors. They improved phone verification process and introduced new challenges, such as reCAPTCHAs, to help ensure that a human is in control of an account. They also enforce against simultaneous actions such as likes, retweets, or follows from multiple accounts [9]. The consequences for violating policies depend on the severity of the violation as well as any previous history of violations. Twitter takes actions according to the type of spam activity that is identified, that may include the following:

- **Anti-spam challenges**
  When Twitter detects suspicious levels of activity, accounts may be locked and prompted to provide additional information (e.g., a phone number) or to solve a reCAPTCHA.

- **Blacklisting URLs**
  Twitter blacklists or provides warnings about URLs that it believes to be unsafe.

- **Tweet deletion and temporary account locks**
  o If the platform manipulation or spam offense is an isolated incident or first offense, the actions are ranging from requiring deletion of one of more Tweets to temporarily locking account(s). Any subsequent platform manipulation offenses will result in permanent suspension.
  o In the case of a violation centering around the use of multiple accounts, the account may be asked to choose one account to keep. The remaining accounts will be permanently suspended.
  o If Twitter believes an account may be in violation of its fake accounts policy, it may require from the account to provide government-issued identification (such as a driver’s license or passport) in order to reinstate the account.

- **Permanent suspension**
  For severe violations, accounts will be permanently suspended at first detection. Examples of severe violations include:
  o operating accounts where the majority of behavior is in violation of the policies described above
  o using any of the tactics to undermine the integrity of elections
  o buying/selling accounts
  o creating accounts to replace or mimic a suspended account
  o operating accounts that Twitter is able to reliably attribute to entities known to violate the Twitter Rules.
1.2.4 Regulations for Bot Accounts

Recently on July 1st, 2019, California became the first state to require bot accounts in social media platforms to openly identify as automated online accounts. This new bot-disclosure law requires all bots that attempt to influence California residents’ voting or purchasing behaviors to conspicuously declare themselves. That means that the owner or creator of any bot account is responsible for prominently designating the account as automated [10]. However, this has raised a series of questions, as in “How is a bot defined?” and “Who is responsible for identifying and reporting bots?”.

1.2.5 European Union General Data Protection Regulation

The regulation in Article 22 about “Automated individual decision-making, including profiling” [11] states:

1. The data subject shall have the right not to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her.

2. Paragraph 1 shall not apply if the decision:
   (a) is necessary for entering into, or performance of, a contract between the data subject and a data controller;
   (b) is authorized by Union or Member State law to which the controller is subject, and which also lays down suitable measures to safeguard the data subject's rights and freedoms and legitimate interests; or
   (c) is based on the data subject's explicit consent.

3. In the cases referred to in points (a) and (c) of paragraph 2, the data controller shall implement suitable measures to safeguard the data subject's rights and freedoms and legitimate interests, at least the right to obtain human intervention on the part of the controller, to express his or her point of view and to contest the decision.

4. Decisions referred to in paragraph 2 shall not be based on special categories of personal data referred to in Article 9(1), unless point (a) or (g) of Article 9(2) applies and suitable measures to safeguard the data subject's rights and freedoms and legitimate interests are in place.

According to Article 4(4) profiling is “Any form of automated processing of personal data consisting of the use of personal data to evaluate certain personal aspects relating to a natural person, in particular to analyze or predict aspects concerning that natural person’s performance at work, economic situation, health, personal preferences, interests, reliability, behavior, location or movements”. Based
on Article 22, it is made clear that the GDPR restricts anyone from making solely automated decisions, including those based on profiling, that have a legal or similarly significant effect on individuals. For something to be solely automated there must be no human involvement in the decision-making process. The GDPR also [12]:

- requires giving individuals specific information about the processing
- obliges taking steps to prevent errors, bias and discrimination
- obliges giving individuals rights to challenge and request a review of the decision.

1.2.6 Approach to the Problem of Spam

Unfortunately, there is no guarantee solution to the problem of Spam. For two decades, researchers and developers approach this issue by observing spam behaviors and arming their systems appropriately in order to prevent their reoccurrence. Contrariwise, spammers perpetually discover new and improved ways to permeate into the systems. The most effective means of defeating spam is to find a way of immediately detecting spam accounts and report them to be suspended from Social Media. This of course, is not an easy task for users as individuals, since there are thousands of spam accounts, and most importantly accounts that have managed to remain active and undetected for long on the platform. Of course, Twitter constantly attempts to figure out ways to detect and restrict spammers.

With a tweet as viewed below, on April 8th, 2019, Twitter announced that the allowed following limit per day would decrease from 1000 accounts to 400. However, not even Twitter itself has come up with a concrete way of distinguishing spammers and bots from actual users, and its algorithms have been criticized a lot from its users for being inaccurate and biased.
A user-friendly system that would be able to identify accounts as spam/bots or non-spam/non-bots is in fact necessary. The response of such an intelligent tool would help speed up the process of reporting and eventually suspending spam accounts.

1.3 Thesis Contribution and Challenges

The purpose of this thesis can be divided into three parts. The first part includes reviewing the literature on bot/spam detection on social media and comparing techniques and methods applied in the field of Machine Learning. Since the creation of social media networks, spamming as a phenomenon attracts more and more researchers, and thus a lot of work is available to consult and provide some guidelines. The second part includes the attempt of constructing a Machine Learning model on bot detection able to output explanatory results about the decisions it makes. As it was previously mentioned, it is important having a way of “interpreting” a model’s predictions. The third part includes the development of an API, also known as Application Programming Interface, and the deployment of a web application - tool that uses the aforementioned model in order to provide its users with information about why an account is characterized as a bot or not.

1.3.1 Selection of the Twitter Social Media Platform

Twitter is, along with Facebook and Instagram, on the top three most popular social media platforms worldwide with more than 330 million active users. Because of its wide use and the automation that the Twitter API provides, spam and bot accounts are a lot more common than on any other platform. Also, a great variety of research and previous work on the subject exists in the literature and a few labeled datasets are publicly available for use. In addition, Twitter is one of the few social media platforms that provide an API. An API, according to Wikipedia, is an interface or communication protocol that allows computer programs to communicate so that they can request and deliver information [13]. The Twitter API allows access to Twitter data derived from public accounts, which is necessary for the implementation of this work.

1.3.2 Explainable AI

Machine Learning, and Artificial Intelligence in general, is not by its nature meant to be explainable towards humans, but as its name suggests, its main purpose is to create algorithms that “learn” input data in order to build mathematical models able to perform on specific tasks. It is in essence the effort to predict with as much accuracy as possible the mathematical curve that best fits the data and, as it’s widely known, the mapping of mathematical curves into physical phenomena is inherently a very difficult problem to solve. A Machine Learning system is a black box, which means
that humans do not understand how it works internally and why it reaches to what
decisions, as it can be seen in Figure 8 below:

![Figure 8: Machine Learning as a black box](image)

The latest trend in research is leaning towards an effort of finding ways of “translating” Machine Learning models, and that lead to the creation of a new field in Data Science called Explainable Artificial Intelligence, also known as XAI. The main issue is that traditional Machine Learning methods will provide no answer why the model predicted a particular label for a single instance and what features were most influential for that particular [15]. In a nutshell, the development of explainable models could [16]:

- Improve human readability
- Determine the justifiability of the decision made by the machine
- Help in deciding accountability, liability leading to good policymaking
- Avoid discrimination
- Reduce societal bias
- Help in verifying and improving the system

1.3.3 Contribution

Taking into consideration all that was mentioned in this Section, it becomes obvious that it is important for Twitter users to be able to recognize whether an account is a bot or not, and more specifically how and why an account might be classified as a bot, through a user friendly environment that will also comply to the GDPR fundamentals. To approach this issue, the focus of this thesis is developing a method using Explainable Artificial Intelligence. To the author’s knowledge no such work exists in the literature, and this gives the opportunity for further research and experimentation. In order to proceed with this effort, the creation of a dataset according to the issue of bot detection was necessary, which is also available for public use. For the purposes of this work, an API was also developed and deployed along with a User Interface (UI) to help with the testing process, and to facilitate users who wish to identify suspicious accounts as well.
1.4 Document Structure

This document is comprised by 6 chapters in total, including the introductory chapter.

In Chapter 2, the readers are presented with a brief introduction to Machine Learning and Artificial Intelligence basics, along with definitions about the issue of bots on social media platforms and a selection of methods found in the literature on approaching bot/spam detection. A categorization of the literature on the subject follows, as well as a categorization of features used in the reviewed works. Lastly, a comparison of the reviewed works is been made, and opportunities for further research and experimentation that arise are presented.

In Chapter 3, the methodology followed on the developed system’s architecture is analyzed and broken down into its core pieces. The capabilities and requirements of each component are also included.

In Chapter 4, the necessary technological knowledge is provided in order for the readers to understand the development’s flow and the choices made during the process of the overall system’s implementation. Additionally, examples of usage of the developed UI system are presented.

In Chapter 5, experiments that were conducted with different algorithms and parameters are presented and the difficulties encountered during the implementation process in general are addressed.

In Chapter 6, conclusions of this thesis are made, and ideas are suggested for further research and future work.
Chapter 2
Related Work

This chapter includes the literature review on bot/spam detection along with some basic background knowledge on Machine Learning and Artificial Intelligence.

As mentioned in the previous Section, spammers exploit the Twitter network by abusing the services it provides, to lure unsuspecting users into following their intentions. In the literature there is plenty of work in bot/spam detection on social media, where the focus is on classifying accounts as bots/spammers or non bots/non spammers. At this point, taking into account all the related work which we cover at the next Section, we propose a definition for bot and spam users, according to which:

**Bot user/Bot:** is a computer programmed account, that follows an automated behavior as auto-posting or auto-replying, that may or may not be malicious.

**Spam user/Spammer:** is a user or account that abuses the network in order to fulfill its malicious purposes, whether that is promoting a service or product, spreading fake news, stealing personal information, impersonating humans, or even polluting the network in general with useless spam data.

2.1 Methods

We could safely group the literature into three core categories of methods according to the nature of the features that each one uses:

- User/Content based features
- Graph based
- Hybrid

**Features:** are the fields of interest of the dataset in study (e.g. age).

**Dataset:** is a structured collection of data.

**User/Content based features** can be obtained by the Twitter API and provide information about the user’s profile and tweets. In some works, you might encounter these features as Profile based. Such features could be tweets, the number of followers or even the location of the user. These are very important features since this huge amount of data can be interpreted to provide intelligence on a user’s behavior on Twitter.
Graph: is a structure that consists of nodes and edges. A node can represent any unique entity (e.g. a person) and the edges can represent any kind of connection between the nodes (e.g. friendship).

Graph based features are features that can be extracted from graphs. Social graphs can be constructed by using profiles as nodes and their connections as edges, to represent the type of relations the nodes have with each other. There are many ways that a Twitter graph can be created, and it entirely depends on the judgement of the researchers to decide on what direction to follow, but the most common structure is using Twitter accounts as nodes and their following information as edges. Then, by applying graph analytics algorithms and metrics, graph features can be extracted to train a model. Some of these graph features could be the clustering coefficient or the betweenness centrality, which are metrics often used for graph analysis.

Hybrid based features are essentially a mixture of user/content and graph-based features. Many works use hybrid features to increase the accuracy and recall scores. Other works might apply graph analytics tools to find suspicious nodes in the graph and then examine them by using classification methods on user/content-based features, and that is considered a hybrid approach as well.

Machine Learning (ML): is a field of Artificial Intelligence that studies algorithms and techniques that are used in classification or clustering of data.

Classification: is the process that an algorithm follows to predict and assign classes to unclassified data in Machine Learning.

Class: is the resulting field that holds the category of the data item in question (e.g. bot).

The classification process can be separated into two parts: training, which is the process of “learning”, by gaining knowledge from the features of classified data, and testing, which is the process of applying the knowledge gained from the training process to the unclassified data. A model is the result of classification.

All methods described above lay under the umbrella of Machine Learning. In this work we review the works that use classification methods to detect bots/spammers.

By reviewing the undermentioned literature, it easily becomes apparent that most researchers do not deviate a lot from one another. Most of them follow the fairly straight process of collecting a dataset, extracting some features and applying core Machine Learning methods on them. Only some works seem to not follow the curve, by examining the produced social graph and extracting graph-based features. As the authors of these works argue, the selection of each feature is very coupled with the context of the acquired data. Thus, in the Sections that follow, there has been an effort
to catalogue the generic ideas that were followed by researchers firstly, and the collection of each specific feature is presented, only, at the end of this chapter in Table 2.

2.1.1 Machine Learning Based Methods

To begin with, the works below constitute what is the state-of-the-art in the field of bot detection currently. What sets them apart from other works is the extraordinary results that they achieved and the valuable datasets that they provided to the community for further exploration and research. As is explained below, they followed the norm in a Machine Learning task which is extracting user and content-based features and then using various combinations of off-the-self algorithms, which they fit on the data.

- In [17] Yang et al. review the different types of social bots and their interaction methods, and propose a supervised machine learning approach, based on random forest classifiers, for bot detection in Twitter. They state that they extracted 1209 features, however they do not provide a list of them. They also provide a UI tool, Botometer, where Twitter users can give a @username as input and the tool will extract the @username’s information from the Twitter API and use it to create features to test their pretrained classification model, to predict if the @username is a bot.

- Yang et al. [18] in a more recent work propose a framework that uses a subset of only user-based features to improve the scalability of the implemented system. They proceed in creating a dataset consisted by already existing labeled datasets in the literature with the addition of three novel datasets. They then train a Random forest classifier using different subsets of all the available datasets to improve the model’s accuracy and maximize the generalization in the testing dataset subsets.

The works below follow a rather similar pipeline with one another. More specifically, they collected data from Twitter, which they then manually labeled, and then, as is the norm, extracted user and content-based features and trained various classification algorithms. They provided a valuable asset for the research that was conducted in the context of this thesis, as they were a resource for understanding which of the characteristics of both tweets and users provide the best results.

- In [19] Benevenuto et al. focus their work on spam detection by investigating content attributes and user behavior attributes of Twitter users. They then proceed to create and manually label a dataset of approximately 1000 users, including a sufficient amount of spam users, in order to test their assumptions on the differences between spammers and non-spammers. Based on the theory that non-
spammers spend more time replying and retweeting from users that they usually know, and tweeting without URLs, these actions should be mirrored in the data collected.

- Similarly, Chen et al. [20] study the behavior of content polluters, a category of spammers on Twitter. To do that, they manually labeled a dataset of 10000 tweets that they crawled from the Twitter API and performed a classification method, using a Random Forest classifier, to detect spam tweets. Then, a user is characterized as a content polluter if most of the tweets they posted are classified as spam. They achieved high accuracy and precision results, but they do not provide any feedback on the results of their predictions.

- In [21] Lin et al. examine the behavior of 800 long existing accounts that they manually evaluated as spam accounts and train their classification model on only two features to check their effectiveness on detecting spam accounts on their collected dataset, created by manually labeling profiles crawled from the Twitter API. The results concluded in high precision and recall scores, and the classification with only two features worked significantly good in detecting spam accounts.

- The authors in [22] suggest some novel features and compare the performance of four traditional classifiers on user and content-based features for spam detection on Twitter. They manually labeled a dataset of 1000 users, based on their profile information and their 100 most recent tweets. The evaluation concluded that the Random Forest classifier produced the best results.

- A similar work in [23] also studies the performance of three classifiers using user-based and content-based features to conduct spam detection on collected Twitter profiles. For the input dataset they used the one that was created in [17], taking into account 1065 users of which the 355 were labeled as spammers and the 710 as legitimate users. They also tested the dataset using different dimensions of features to observe the impact of them on the classification, pointing out that by using less features the classification increased the accuracy of detecting bots and decreased the computational time.

Also, worth mentioning are the works that follow, who by deploying honeypot accounts as bait were able to create much more effortlessly a large collection of bot accounts.

- Lee et al. [24] follow a similar direction in spam detection. They created a mixture of honeypot accounts, accounts designed to lure spammers to interact with them and log information about their profiles (followers, posts, etc.). Some of the honeypots contained profile information and others not, and some would post tweets while others would not, to make observations about the behavior of
spammers with each profile. After collecting potential spam profiles attracted from
honeypot accounts, they manually labeled a set of them and added some
legitimate user profiles to create their dataset. They trained a classification model
with user/content features provided by the Twitter API with the addition of some
text classification features extracted from the 20 most recent tweets of a user’s
profile.

- Following their previous work, Lee et al. [25] created a bigger network of
honeypot accounts that only interacted with each other, in order to lure spam
accounts into engaging with them and hence track their information and behavior.
They trained 30 classifiers with an updated set of features.

- Stringhini et al. [26] adapted the method of creating 300 honeypot profiles to
collect spam data and used classification to detect both individual spam bots but
also groups of spam profiles that were operated by the same one spammer. The
honeypot accounts would only receive any follow requests and log the activity
(account information, tweets and direct messages) of the accounts that made
interactions. They proceeded by manually labeling the accounts collected from the
honeypots and combined them with legitimate accounts to create their dataset
that was used in a Random Forest classifier.

2.1.2 Graph Based Methods

A work that uses graph properties to detect spammers on a graph that connects
normal users with spammers can be found in [27]. Spam nodes, nodes that represent
spammers in a graph, tend to form edges between them to alter the structure of the
graph in a way that matches the structure of connections between legitimate users.
First, they test the graph for clustering and neighborhood independence properties to
mark suspicious nodes in the network. Applying the proposed algorithms (GREEDY,
TRWALK) to the graph, can expose spammers that reach a large, random set of normal
users for fulfilling their malicious purposes.

The study in [28] suggests handling spam detection on social media platforms
as a horizontal anomaly detection problem. Firstly, they form the reputation graph
and then apply the fuzzy system to evaluate each user over the features examined for
their interactions in the social network. This graph is then used to calculate the self-
healing cost of each user. They then calculate the neuro-fuzzy cost, which is used to
determine whether a user is an anomaly or not.

In another work [29] the authors analyze the cyber-criminal ecosystem, the
long existing spam accounts on Twitter. They use an existing labeled dataset to create
a graph consisted of 2000 spam accounts. They then apply graph metrics to the graph
to gain information about the relationship characteristics of the nodes and the outer
social relationships between criminal accounts and their supporters. They present common characteristics of these supporters and investigate how criminal accounts could be concealed into the Twitter space. They also propose a Criminal Accounts Inference Algorithm to unmask criminal accounts.

2.1.3 Hybrid Based Methods

Although graph-based approaches might be effective in detecting bot/spam accounts, the fact is that most of the time they cannot be used in a real-time framework, because the created account network might not be complete at prediction time (it might lack edges and/or nodes). However, using a hybrid approach, with graph properties only in cases where information about the node in question is available, would lead to better classification results since the model would decide for the classification class with bigger certainty. Works that proceeded with this point of view are reviewed below.

The authors in [30] suggest a hybrid approach on detecting spammers through using features collected from Twitter in combination with graph-based features, because of the tendency of spammers on adapting their interactions to evade the detection mechanisms. Graph based features increase the certainty of answering whether a user is a bot or not, since spammers in general follow random users, and hence they will probably not be part of a community.

In [31] Song et al. propose a spam filtering method for Twitter by using graph-based features and a classification algorithm, because they observed that spammers rarely connect with non-spammers, but they form groups with other spammers with only a few attack edges to honest regions. They built a directed network of followers and followees and analyzed the users in pairs. Combining the graph-based features with some user related features did indeed lead to better classification results.

In [32] Mateen et al. used graph-based features along with some user/content-based features to combat evading tactics of spammers. They analyzed a selection of features and then used them in a dataset provided by another work. They trained the dataset using various classifiers and they observed that using both kinds of features for classification translated into better results with high accuracy.

The authors in [33] deployed honeypot profiles on Twitter to study link fraud detection and monitor users that have a suspicious behavior in the network. They used popular services available online to purchase fake followers and used them on the honeypots. Since all of the honeypots' followers were fake, they used the Twitter API to collect information for these accounts and some known legitimate accounts to create a labeled dataset. They then developed egonets, one graph for each fake
account as the central node followed by their connections as the neighbor nodes, for each of the fake accounts, to analyze the patterns that were created and the differences between the services. They measured the entropy of the fake followers' features and then performed classification to evaluate them.

In another work [34] both graph-based features and user-based features are proposed for spam detection on Twitter. The author modeled Twitter as a directed graph of users as nodes and their connections between them as edges. Using the Twitter Streaming API, he collected 25000 accounts and used a group of which he manually labeled into spammers and non-spammers to create the dataset, where the 3% of it comprised of spammers to reflect the percent of spam accounts on Twitter.

Yang et al. [35] propose a classification method that uses some novel graph-based features, computed by graph metrics, along with some known user/content-based features for detecting spammers in Twitter. They initially crawled twenty accounts as seed from the Twitter Streaming API, and then for every account they crawled its followers and followees and repeated this process to create the dataset. They then define spam tweet as a tweet containing at least one malicious or phishing URL. They manually labeled 2000 accounts with high spam tweet ratio as spam accounts and by analyzing their behavior they extracted some user/content-based features. They then handle the network as a graph and extract graph-based features which they used along with the user/content-based features to train their classifier.

Worth mentioning is a selection of literature ([36], [37], [38], [39]) that used a different approach to detect spam on social media. They performed text analysis on the text of each tweet to identify the URLs, the hostnames and other URL related information, and then attempted to answer whether a URL linked to suspicious or blacklisted websites. The basic idea is that users who post tweets containing URLs of that nature are in fact spam users. Since spammers have found ways to conceal their URLs by using URL shorteners, this suggested method can no longer be applied for bot/spam detection.

2.1.4 Categorization of Literature

The following Table summarizes the works found in the literature that focused on Machine Learning based methods and Hybrid based methods in order to easily compare them according to the features and classifiers that each uses, how they output the model predictions, and whether or not they provide datasets and UI tools.
<table>
<thead>
<tr>
<th>Authors</th>
<th>Number of features</th>
<th>User / Content based features</th>
<th>Graph based features</th>
<th>Provide new dataset</th>
<th>Classifier</th>
<th>Prediction with class</th>
<th>Prediction with probability score</th>
<th>UI Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yang et al. (2019) [17]</td>
<td>1209</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Random Forest</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Yang et al. (2020) [18]</td>
<td>20</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Random Forest</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benevenuto et al. (2010) [19]</td>
<td>62</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>SVM</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lee et al. (2011) [25]</td>
<td>19</td>
<td>✓</td>
<td></td>
<td></td>
<td>Naive Bayes, Logistic Regression, SVM, Tree-based algorithms</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lin et al. (2013) [21]</td>
<td>2</td>
<td>✓</td>
<td></td>
<td></td>
<td>J48</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>McCord et al. (2011) [22]</td>
<td>8</td>
<td>✓</td>
<td></td>
<td></td>
<td>Random Forest, SVM, Naive Bayes, K-Nearest Neighbor</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meda et al. (2014) [23]</td>
<td>14</td>
<td>✓</td>
<td></td>
<td></td>
<td>SVM, ELM, Random Forest</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stringhini et al. (2010) [26]</td>
<td>6</td>
<td>✓</td>
<td></td>
<td></td>
<td>Random Forest</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fazil et al. (2018) [30]</td>
<td>19</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Random Forest, Decision Tree, Bayesian Network</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Song et al. (2011) [31]</td>
<td>14</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Bagging, LibSVM, FT,</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.1.5 Categorization of Features

The following Table summarizes and categorizes the collection of features found in the literature.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Type</th>
<th>Value</th>
<th>Works</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of hashtags per number of words of each tweet</td>
<td>content</td>
<td>float number</td>
<td>[19]</td>
</tr>
<tr>
<td>number of URLs per words of each tweet</td>
<td>content</td>
<td>float number</td>
<td>[19]</td>
</tr>
<tr>
<td>number of words of each tweet</td>
<td>content</td>
<td>integer number</td>
<td>[19]</td>
</tr>
<tr>
<td>number of characters of each tweet</td>
<td>content</td>
<td>integer number</td>
<td>[19], [23]</td>
</tr>
<tr>
<td>number of symbols of each tweet</td>
<td>content</td>
<td>integer number</td>
<td>[20]</td>
</tr>
<tr>
<td>number of URLs of each tweet</td>
<td>content</td>
<td>integer number</td>
<td>[19], [20]</td>
</tr>
<tr>
<td>number of URLs per number of tweets</td>
<td>content</td>
<td>float number</td>
<td>[24], [25], [30], [35]</td>
</tr>
<tr>
<td>number of unique URLs per number of tweets</td>
<td>content</td>
<td>float number</td>
<td>[24], [25], [30], [35]</td>
</tr>
<tr>
<td>total number of URLs of all tweets</td>
<td>content</td>
<td>integer number</td>
<td>[22], [23]</td>
</tr>
<tr>
<td>number of tweets with URLs</td>
<td>content</td>
<td>integer number</td>
<td>[34]</td>
</tr>
<tr>
<td>number of tweets with URLs per number of tweets</td>
<td>content</td>
<td>float number</td>
<td>[21], [26], [31], [34]</td>
</tr>
<tr>
<td>Number of Tweets with Mentions with URLs per Number of Tweets with Mentions</td>
<td>Content</td>
<td>Float Number</td>
<td>[31]</td>
</tr>
<tr>
<td>Number of Tweets with Mentions per Number of Tweets</td>
<td>Content</td>
<td>Float Number</td>
<td>[21], [31], [32], [34]</td>
</tr>
<tr>
<td>Number of Mentions per Number of Tweets</td>
<td>Content</td>
<td>Float Number</td>
<td>[24], [25], [30], [35]</td>
</tr>
<tr>
<td>Number of Unique Mentions per Number of Tweets</td>
<td>Content</td>
<td>Float Number</td>
<td>[24], [25], [30]</td>
</tr>
<tr>
<td>Number of Mentions of Non-Followers per Number of Mentions</td>
<td>Content</td>
<td>Float Number</td>
<td>[31]</td>
</tr>
<tr>
<td>Number of Hashtags on Each Tweet</td>
<td>Content</td>
<td>Integer Number</td>
<td>[19], [20]</td>
</tr>
<tr>
<td>Total Number of Hashtags of All Tweets</td>
<td>Content</td>
<td>Integer Number</td>
<td>[22], [23]</td>
</tr>
<tr>
<td>Number of Hashtags per Number of Tweets</td>
<td>Content</td>
<td>Float Number</td>
<td>[30], [35]</td>
</tr>
<tr>
<td>Number of Tweets with Hashtags per Number of Tweets</td>
<td>Content</td>
<td>Float Number</td>
<td>[31]</td>
</tr>
<tr>
<td>Number of Tweets with Hashtags per Total Number of Tweets Multiplied by the Number of Tweets with Unique Hashtags</td>
<td>Content</td>
<td>Float Number</td>
<td>[32]</td>
</tr>
<tr>
<td>Number of Tweets with URLs per Total Number of Tweets Multiplied by the Number of Tweets with Unique URLs</td>
<td>Content</td>
<td>Float Number</td>
<td>[32]</td>
</tr>
<tr>
<td>Number of Media on Each Tweet</td>
<td>Content</td>
<td>Integer Number</td>
<td>[20]</td>
</tr>
<tr>
<td>Number of Numeric Characters (i.e. 1,2,3) that Appear on the Text</td>
<td>Content</td>
<td>Integer Number</td>
<td>[19], [23]</td>
</tr>
<tr>
<td>Number of Users Mentioned on Each Tweet</td>
<td>Content</td>
<td>Integer Number</td>
<td>[19], [20]</td>
</tr>
<tr>
<td>Number of Times the Tweet Has Been Retweeted</td>
<td>Content</td>
<td>Integer Number</td>
<td>[19], [20]</td>
</tr>
<tr>
<td>Total Number of Retweets in All of the Tweets</td>
<td>Content</td>
<td>Integer Number</td>
<td>[22]</td>
</tr>
<tr>
<td>Number of Retweets per Number of Tweets</td>
<td>Content</td>
<td>Float Number</td>
<td>[30]</td>
</tr>
<tr>
<td>Number of Times the Tweet Has Been Favorited</td>
<td>Content</td>
<td>Integer Number</td>
<td>[20]</td>
</tr>
<tr>
<td>Number of Tweets with At Least One Word from List of Spam Words per Number of Tweets</td>
<td>Content</td>
<td>Float Number</td>
<td>[19]</td>
</tr>
<tr>
<td>Total Number of Spam Words of All Tweets</td>
<td>Content</td>
<td>Integer Number</td>
<td>[23], [32]</td>
</tr>
<tr>
<td>Number of Spam Words per Number of Tweets</td>
<td>Content</td>
<td>Integer Number</td>
<td>[22]</td>
</tr>
<tr>
<td>Sum of Word Weights for All Words of the Tweet for All of the User’s Tweets</td>
<td>Content</td>
<td>Integer Number</td>
<td>[22]</td>
</tr>
<tr>
<td>Number of Tweets That Are Reply Messages per Number of Tweets</td>
<td>Content</td>
<td>Float Number</td>
<td>[19]</td>
</tr>
<tr>
<td>Metric</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Number of tweets of the user with URLs per number of tweets</td>
<td>Content</td>
<td>Float number [19]</td>
<td></td>
</tr>
<tr>
<td>Sensitive tweet</td>
<td>Content</td>
<td>Boolean [19]</td>
<td></td>
</tr>
<tr>
<td>Average content similarity over all pairs of tweets</td>
<td>Content</td>
<td>Float number [24], [25], [26], [31], [35]</td>
<td></td>
</tr>
<tr>
<td>Average content and hashtag similarity</td>
<td>Content</td>
<td>Float number [30]</td>
<td></td>
</tr>
<tr>
<td>Average tweet length</td>
<td>Content</td>
<td>Float number [22]</td>
<td></td>
</tr>
<tr>
<td>Total number of mentions of all tweets</td>
<td>Content</td>
<td>Integer number [22], [23]</td>
<td></td>
</tr>
<tr>
<td>Number of automated tweets per number of tweets</td>
<td>Content</td>
<td>Float number [20], [30], [35]</td>
<td></td>
</tr>
<tr>
<td>Number of automated tweets with URLs per number of automated tweets</td>
<td>Content</td>
<td>Float number [30], [35]</td>
<td></td>
</tr>
<tr>
<td>Average automated tweet similarity</td>
<td>Content</td>
<td>Float number [30], [35]</td>
<td></td>
</tr>
<tr>
<td>Distribution of tweets over 24-hour period (8 periods)</td>
<td>Content</td>
<td>Float number [22]</td>
<td></td>
</tr>
<tr>
<td>Tweet time standard deviation</td>
<td>Content</td>
<td>Float number [30], [33]</td>
<td></td>
</tr>
<tr>
<td>Tweet time interval standard deviation</td>
<td>Content</td>
<td>Float number [30], [31]</td>
<td></td>
</tr>
<tr>
<td>Number of duplicate tweets per number of tweets</td>
<td>Content</td>
<td>Float number [31]</td>
<td></td>
</tr>
<tr>
<td>Number of duplicate tweets</td>
<td>Content</td>
<td>Integer [34], [35]</td>
<td></td>
</tr>
<tr>
<td>Number of followers</td>
<td>User</td>
<td>Integer number [18], [19], [20], [25], [22], [23], [32], [33], [34], [35]</td>
<td></td>
</tr>
<tr>
<td>Number of followees</td>
<td>User</td>
<td>Integer number [18], [19], [20], [25], [22], [23], [32], [34], [35]</td>
<td></td>
</tr>
<tr>
<td>Number of followers per number of followees</td>
<td>User</td>
<td>Float number [18], [19], [20], [24], [25], [26], [35]</td>
<td></td>
</tr>
<tr>
<td>Number of followees per number of followers</td>
<td>User</td>
<td>Float number [19], [20], [24], [25], [31], [32]</td>
<td></td>
</tr>
<tr>
<td>Number of friends per number of followees</td>
<td>User</td>
<td>Float number [24], [25], [30], [35]</td>
<td></td>
</tr>
<tr>
<td>Follower based reputation</td>
<td>User</td>
<td>Float number [30]</td>
<td></td>
</tr>
<tr>
<td>Number of friends per number of followers</td>
<td>User</td>
<td>Float number [25]</td>
<td></td>
</tr>
<tr>
<td>Number of friends</td>
<td>User</td>
<td>Integer number [26], [33], [35]</td>
<td></td>
</tr>
<tr>
<td>Standard deviation of unique numerical IDs of followees</td>
<td>User</td>
<td>Float number [25]</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Type</td>
<td>Reference</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>standard deviation of unique numerical IDs of followers</td>
<td>user</td>
<td>float</td>
<td>[25]</td>
</tr>
<tr>
<td>number of followers per number of followers and followees (reputation)</td>
<td>user</td>
<td>float</td>
<td>[30], [32], [34], [35]</td>
</tr>
<tr>
<td>average number of all followers followees to number of followers</td>
<td>user</td>
<td>float</td>
<td>[30]</td>
</tr>
<tr>
<td>number of followers to account’s age</td>
<td>user</td>
<td>float</td>
<td>[18]</td>
</tr>
<tr>
<td>number of followees to account’s age</td>
<td>user</td>
<td>float</td>
<td>[18]</td>
</tr>
<tr>
<td>number of tweets the user favorite to account’s age</td>
<td>user</td>
<td>float</td>
<td>[18]</td>
</tr>
<tr>
<td>number of lists including the user to account’s age</td>
<td>user</td>
<td>float</td>
<td>[18]</td>
</tr>
<tr>
<td>number of tweets</td>
<td>user</td>
<td>integer</td>
<td>[18], [19], [20], [25], [23], [26], [32], [33], [35]</td>
</tr>
<tr>
<td>age of the user account in days</td>
<td>user</td>
<td>integer</td>
<td>[19], [20], [24], [25], [23], [31], [32], [33], [35]</td>
</tr>
<tr>
<td>number of times the user was mentioned</td>
<td>user</td>
<td>integer</td>
<td>[19]</td>
</tr>
<tr>
<td>number of times the user was replied to</td>
<td>user</td>
<td>integer</td>
<td>[19]</td>
</tr>
<tr>
<td>number of times the user replied to someone</td>
<td>user</td>
<td>integer</td>
<td>[19]</td>
</tr>
<tr>
<td>number of followees of the user’s followers</td>
<td>user</td>
<td>integer</td>
<td>[19]</td>
</tr>
<tr>
<td>number of tweets received from followees</td>
<td>user</td>
<td>integer</td>
<td>[19]</td>
</tr>
<tr>
<td>existence of spam words on the user’s screenname</td>
<td>user</td>
<td>boolean</td>
<td>[19]</td>
</tr>
<tr>
<td>time between tweets (frequency)</td>
<td>user</td>
<td>float</td>
<td>[18], [19], [23], [32], [35]</td>
</tr>
<tr>
<td>number of tweets posted per day</td>
<td>user</td>
<td>integer</td>
<td>[19], [24], [25], [23]</td>
</tr>
<tr>
<td>number of tweets posted per week</td>
<td>user</td>
<td>integer</td>
<td>[19], [23]</td>
</tr>
<tr>
<td>likelihood of screenname</td>
<td>user</td>
<td>float</td>
<td>[18]</td>
</tr>
<tr>
<td>number of characters of the username</td>
<td>user</td>
<td>integer</td>
<td>[18], [25]</td>
</tr>
<tr>
<td>number of characters of the screenname</td>
<td>user</td>
<td>integer</td>
<td>[18]</td>
</tr>
<tr>
<td>number of numeric characters in the username</td>
<td>user</td>
<td>integer</td>
<td>[18]</td>
</tr>
<tr>
<td>number of numeric characters in the screenname</td>
<td>user</td>
<td>integer</td>
<td>[18]</td>
</tr>
<tr>
<td>Feature</td>
<td>Type</td>
<td>Description</td>
<td>References</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>------------</td>
<td>--------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>number of characters of the description</td>
<td>user</td>
<td>integer number</td>
<td>[18], [25]</td>
</tr>
<tr>
<td>profile location</td>
<td>user</td>
<td>boolean</td>
<td>[20], [33]</td>
</tr>
<tr>
<td>profile URL</td>
<td>user</td>
<td>boolean</td>
<td>[20]</td>
</tr>
<tr>
<td>profile description</td>
<td>user</td>
<td>boolean</td>
<td>[20]</td>
</tr>
<tr>
<td>verified user</td>
<td>user</td>
<td>boolean</td>
<td>[18], [20], [33]</td>
</tr>
<tr>
<td>default profile</td>
<td>user</td>
<td>boolean</td>
<td>[18], [20], [33]</td>
</tr>
<tr>
<td>default profile picture</td>
<td>user</td>
<td>boolean</td>
<td>[20], [33]</td>
</tr>
<tr>
<td>use of background image</td>
<td>user</td>
<td>boolean</td>
<td>[18]</td>
</tr>
<tr>
<td>number of tweets the user favorite</td>
<td>user</td>
<td>integer number</td>
<td>[18], [20], [33]</td>
</tr>
<tr>
<td>change rate of number of following</td>
<td>user</td>
<td>float number</td>
<td>[25], [35]</td>
</tr>
<tr>
<td>number of all first names of followees per number of unique first names of followees</td>
<td>user</td>
<td>float number</td>
<td>[26]</td>
</tr>
<tr>
<td>the number of lists including the user</td>
<td>user</td>
<td>integer number</td>
<td>[18], [20], [31], [33]</td>
</tr>
<tr>
<td>protected profile</td>
<td>user</td>
<td>boolean</td>
<td>[33]</td>
</tr>
<tr>
<td>English language</td>
<td>user</td>
<td>boolean</td>
<td>[33]</td>
</tr>
<tr>
<td>clustering coefficient</td>
<td>graph</td>
<td>float number</td>
<td>[30], [35]</td>
</tr>
<tr>
<td>community reputation</td>
<td>graph</td>
<td>float number</td>
<td>[30]</td>
</tr>
<tr>
<td>community clustering coefficient</td>
<td>graph</td>
<td>float number</td>
<td>[30]</td>
</tr>
<tr>
<td>length of the shortest path between users (distance)</td>
<td>graph</td>
<td>integer number</td>
<td>[31]</td>
</tr>
<tr>
<td>number of independent paths between nodes (connectivity)</td>
<td>graph</td>
<td>integer number</td>
<td>[31]</td>
</tr>
<tr>
<td>random walk used in PageRank</td>
<td>graph</td>
<td>integer number</td>
<td>[31]</td>
</tr>
<tr>
<td>in/out degree</td>
<td>graph</td>
<td>float number</td>
<td>[32]</td>
</tr>
<tr>
<td>betweenness centrality</td>
<td>graph</td>
<td>float number</td>
<td>[32], [35]</td>
</tr>
<tr>
<td>local clustering coefficient</td>
<td>graph</td>
<td>float number</td>
<td>[35]</td>
</tr>
<tr>
<td>average number of neighbors’ followers</td>
<td>graph</td>
<td>float number</td>
<td>[35]</td>
</tr>
<tr>
<td>average number of neighbors’ tweets</td>
<td>graph</td>
<td>float number</td>
<td>[35]</td>
</tr>
<tr>
<td>number of followees to average number of neighbors’ followers</td>
<td>graph</td>
<td>float number</td>
<td>[35]</td>
</tr>
</tbody>
</table>

Table 2: Categorization of Features
2.1.6 Explainable Artificial Intelligence

A classifier is often described as a “black box”, because information about the process of the prediction is not known, as in which feature results in what class, which is important for validating and trusting the decision process. A field of Artificial Intelligence (AI), known as Explainable Artificial Intelligence (XAI) serves the purpose of giving insights to this complex problem. Some indicative works are reviewed below:

In [40] Samek et al. point out the importance of explainable AI models and propose two explanation methods for deep learning models, Sensitivity Analysis and Layer-Wise Relevance Propagation. The first measure, sensitivity analysis, assumes that the most relevant input features are those to which the output is most sensitive. The second method, layer-wise relevance propagation, explains predictions relative to the state of maximum uncertainty by decomposition. The result of the explanation process is a heatmap, which visualizes the importance of each feature for the prediction.

The authors in [15] propose a novel methodology, Local Interpretable Model-agnostic Explanations (LIME), that is able to provide explanations about the predicted classes of any Machine Learning method by learning an interpretable model locally around the prediction. LIME samples instances, gets their predictions from the classifier, and weighs them by the proximity to the instance being explained. They also propose SP-LIME, Submodular Pick Lime, a method that is able to explain models by selecting diverse and representative instances’ predictions and their explanations, to approach the submodular optimization problem. This work was one of the core pieces of research that was leveraged in the development of the software that was created in the context of this thesis.

2.2 Comparison and Opportunities

The presented works in Section 2.1 summarize the range of approaches followed on the field of bot/spam detection on social media using Artificial Intelligence. By studying the literature on bot/spam detection and by observing the Table 1 above, it becomes obvious that the majority of the reviewed papers are not addressing or are missing some important concepts.

Firstly, only a few works provide their trained dataset to the public, and even if they do, the collected data available only represents a very small sample of users that could not reflect sufficiently the size of a network like Twitter. By recent estimations, Twitter’s daily use base is in the order of hundreds of millions, who tweet close to a billion times per day. So, it only makes sense for more and more data to become available for research, which is one of the pains this thesis attempts to address.
Secondly, only one work in [17] provided a public User Interface tool, Botometer, where users can subject @usernames of Twitter accounts and get real-time predictions of whether the input @username is considered a bot or not. Providing a tool like this can speed up the process of detecting bots/spammers instead of manually investigating hundreds of tweets to conclude whether an account is suspicious or not, and especially when the tool achieves high accuracy and recall scores it can respond with greater certainty. As such, the existence of easy-to-use websites that can inform about potential bots without requiring any specific training in computer science or software in general, from the user, is obviously a welcoming addition to the social media ecosystem.

And last but not least, a very important concept is missing from all the works studied above, which is the explainability of the classification results. This constitutes the cornerstone of the software that was developed in the context of this thesis. As mentioned in Section 1.2.5 the predicted scores of such a model must be followed up with an explanation of its decision making under the scope of the European Union General Data Protection Regulation.

To conclude, as mentioned in Section 1.3.3 and above, the contribution of this thesis is creating a system that provides the user with insights about the possibility of a Twitter account belonging to a bot, enhancing that knowledge with easy-to-understand reasons, as to why the model came to that conclusion.

More specifically, firstly, a very large dataset of tweets along with their corresponding labeled authors is created. Afterwards an effort is made to develop and train an explainable Machine Learning model. Lastly, a web application is developed and deployed, that uses the trained model in order to make real-time predictions about Twitter accounts’ @usernames and present the user with satisfactorily results. Details on the methodology and the implementation of the system that was developed can be found on the succeeding chapters of this thesis.
Chapter 3
Methodology

In this Chapter, the infrastructure that was created is explained in detail. Firstly, an overview of the system is presented, and continuing each one of its individual components is detailed.

3.1 System Requirements and Architecture

As it was stated in previous Sections, the end goal of this thesis is the development of a user-friendly environment as a web application, where its users can input Twitter @usernames and receive explainable results regarding the state of the account in question, to better understand whether an account shows signs of a bot-like behavior. To that end, the developed system must comply to some requirements as presented below. More in detail, the system should:

- Be online
- Have a simple and user-friendly design
- Be in the form of a web application
- Receive a big in volume dataset
- Receive user input
- Collect tweets from the Twitter API
- Make predictions about the input user
- Provide the most accurate results possible
- Perform as fast as possible
- Provide explainable results
- Provide the option of submitting feedback about the results anonymously
- Store all data about the predictions and user feedback
- Be reusable

For the reader’s better comprehension, the developed product of this work should follow the structure shown in Figure 9:
A more detailed representation of the prediction process that the model should follow can be viewed in Figure 10 below. In the Sections that follow, the system structure is further analyzed.

![Figure 10: Prediction Process](image)

### 3.2 User Interface

As explained previously, one of the core goals of this thesis, is, to provide an easy form of interaction between the users and the system, and thus, developing a User interface (UI) is important. The task of this interface is, given a Twitter @username as input, to present a resulting score and the reasons why that score was computed, into a comprehensive manner. In that way, the system is essentially accessible by anyone, and does not only address to individuals with a required theoretical and technological background. In order for the UI to be able to present the results to the screen, it is required asking a server by sending a HTTP request and attaching the @username. The way the API server performs is described in Section 3.3 below.
3.3 API Server

The server constitutes the core of the system, which is responsible for collecting data for a single Twitter account at a time, given its @username, then convert this data into the appropriate format as it is described in Chapter 4, and lastly by consulting a Machine Learning model to output the computed results accordingly. For the collection of data, the server makes a request to the Twitter API, in order to retrieve an account’s 20 most recent tweets, a number that was deemed appropriate after experimentation. Afterwards, the server is tasked to convert the available information into a format compatible with the decision system described in the next Section so that it can draw conclusions about the results.

3.4 Machine Learning Model

3.4.1 Data Collection

In order to train a Machine Learning model, the first necessary step is collecting data. That is, the information required to depict the problem. To that end, more than 2 million tweets were collected from the Twitter API. More specifically, the topic that was chosen for data collection, was that of cryptocurrencies. Cryptocurrencies was a very trending topic in 2018, and it seemed reasonable to speculate that they would provide a fairly sizeable pool of bots/spammers. To reinforce that dataset, three more datasets provided by other works, manually annotated, were used.

3.4.2 Dataset Creation

After collecting the tweets from the Twitter API, the next task was to find a way of evaluating the state of each tweet’s author. Since it was physically impossible to manually investigate 2 million tweets and their authors, the idea was to use a third-party app that would provide a fairly confident ground truth about the likelihood of a user being a bot. Luckily, an app called Botometer, developed from the authors of the state-of-the-art paper in the field [17], is available for that purpose and its accuracy promises good results. The author of each tweet was fed as an input to Botometer, and the output scores were stored to be used later on the model, as the class value of each instance. For the three labeled datasets a different approach was followed. For each user in the datasets, ten tweets were collected, and a Random Forest classifier was trained with half of that data. The other half of the data were used to create a second dataset as described in Chapter 4, using the predicted values of that same classifier.
3.4.3 Regressor Algorithm

The main point of this work, as it was previously mentioned, is not developing a Machine Learning model that answers whether a tweet comes from a bot or not, but developing a model that explains its decisions on the subject in a way that is easily understood by humans, on top of that. Thus, many of the widespread techniques used in the field of natural language processing are unfortunately not fitted for this thesis's purpose, e.g. Deep Neural Networks, for the reason that they are not easily explainable. Instead, there is a limitation in existing algorithms. After experimentation, as described in Chapter 4, the model that was chosen was a Stacking regressor with two regressors, namely a Random Forest and a Gradient Boosting one, using Ridge Regression, a particularly effective regression method when there is collinearity in inputting features.

3.5 Conversion of Results to Human Readable Explanations

Last but not least, this step is the most important of the created infrastructure. It involves converting the predictions of the regressor into justifications understandable by humans. It is based mainly on the work of Marco Tulio Ribeiro, Sameer Singh and Carlos Guestrin that developed a Python library named “LIME” [15], which, by introducing an instance for prediction, through mathematical equations can extract information about which of the sample’s features had the biggest impact on the outcome score.

More specifically, by using the trained model’s prediction function, and a given point in its hyperplane, a local linear approximation is being performed, by sampling other instances around that point. By weighting samples around a given point, a linear model could approximate the generic model well, in a small vicinity. One can think of this process, as trying to use a form of linear interpolation around known instances, to find out where the probability function of the model changes the most.

Afterwards, having these numeric data available for each feature, by using a programming script that is based on manual inspection of each rule, the output is a group of reasons written in natural language about the input score. In essence, by leveraging the local linear approximation that was described previously, one can fairly easily conclude what exactly means for a specific feature to lead to high scores, because the problem can safely be reduced to a form of binary problem, that is, if a specific range of feature values is present, it is only there for bots and not for real users and vice-versa.
Chapter 4
Implementation

This Chapter concerns all the implementation details regarding the infrastructure that was created. In regard to the architecture that was detailed in the previous Chapter, in the Sections that follow, the thought and implementation process about each individual component is described.

4.1 Implementation

4.1.1 Twitter API and Data Collection

The Twitter API is a protocol that allows engagement with the Twitter platform and access to public data through a collection of endpoints. For someone to get access to Twitter’s API, they must apply for a developer account and agree to the “Developer Agreement and Policy”, followed by a statement of what use they will make of it, and their assurance that they will not diverge from the rate limiting.

The API provides endpoints for a variety of functionalities. For the process of collecting the tweets for the dataset, the endpoint “Standard search API” was used along with the hashtags #ico, #crypto and #cryptocurrency for a twenty-day period of October-November 2018. The rate limiting from Twitter is set to 450 requests within a 15-minute period. Those three specific hashtags were selected because bitcoin and cryptocurrencies in general were very popular at the time and it was suspected that a huge number of tweets regarding this topic would be generated and obtained. The collected tweets reached 2.1 million in total.

The endpoint returns each tweet into a JSON4 format (JavaScript Object Notation), a text format that is completely language independent, easy for humans to read and write and easy for machines to parse and generate. Part of a tweet represented as a JSON Object can be viewed in Figure 11:

---

4 [https://www.json.org/json-en.html](https://www.json.org/json-en.html)
From each tweet entity, when it is given in this extended JSON format, it is possible to extract valuable information about the tweet’s author, e.g. the number of followers or whether the profile has a description.

4.1.2 Botometer API

After collecting the tweets, the next step was to establish a ground truth about each tweet’s user’s status. For that purpose, as it was stated in Section 3.4.2, a third-party app, Botometer\(^5\), was chosen. Botometer is a web application, developed by the authors of the state-of-the-art paper in the field [17], that checks the activity of a Twitter account and assigns it a score based on how likely the account is to be a bot.

\(^5\) [https://botometer.iuni.iu.edu/](https://botometer.iuni.iu.edu/)
The scores are in the scale of [0-5]. Higher scores are more bot-like. Botometer also provides access to its API\textsuperscript{6}, which in the background makes requests on the Twitter API to collect the required information about an account, and thus rate limiting exists in this case too, allowing 2000 requests per day per account.

The basic idea was to ask Botometer to assign scores to the collected tweets’ authors. In order to do so, Botometer API receives as input the Twitter data in JSON format through a POST request. Then some analysis is performed in the background, the data is scored, and classification results are returned in JSON format. A successful response can be viewed in Figure 12:

\begin{verbatim}
{
  "cap": {
    "english": 0.12,
    "universal": 0.11
  },
  "scores": {
    "english": 0.34,
    "universal": 0.36
  },
  "display_scores": {
    "english": 1.7,
    "universal": 2.0,
    "friend": 2.3,
    "sentiment": 1.7,
    "temporal": 2.8,
    "user": 1.5,
    "network": 2.2,
    "content": 2.1
  },
  "categories": {
    "friend": 0.45,
    "sentiment": 0.34,
    "temporal": 0.55,
    "user": 0.29,
    "network": 0.43,
    "content": 0.41
  },
  "user": {
    "screen_name": "Botometer",
    "id": "2451308594"
  }
}
\end{verbatim}

\textit{Figure 12: Example of Botometer API response}

\textsuperscript{6} https://botometer.iuni.iu.edu/#!/api
The process of asking Botometer API about each tweet’s author was completed in 10 days, by using 3 different accounts to speed up the process. For a large number of tweets, the API could not provide answers about their authors’ scores, since those accounts did not exist anymore. Assuming that Twitter was able to identify those accounts as spam and deactivate them, it seemed reasonable to assume that their scores, predicted by the Botometer API, would be high. Thus, it was decided to assign them with a universal score of 5.1 to distinguish them from other accounts, but still keep them in the data.

4.1.3 Data Storage

Unprocessed data in their JSON format is not easily accessible, especially when its volume is as big as that of 2.1 million tweets. In order to be able to make real-time queries, questions, about the tweets, the use of a database is necessary. A database is an organized collection of data, generally stored and accessed electronically from a computer system [41]. One of the most popular and easy to use databases is MongoDB. MongoDB7 is a free to use, general purpose, document-based, distributed database. It stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time [42].

A database consists of one or more collections of data. A collection is a grouping of MongoDB documents. Typically, all documents in a collection have a similar or related purpose. A document is a record in a MongoDB collection and the basic unit of data in MongoDB. Documents are analogous to JSON objects but exist in the database in a more type-rich format. Each document can hold a number of fields. A field is a name-value pair in a document [43].

For the purpose of storing the collected data, a database in MongoDB was created. Two collections, tweets and users namely, were used to store the collected tweets and the information about the authors’ scores accordingly. A tweet collection’s document contains all the fields that a tweet in its JSON object carries, as in Figure 11. A user collection’s document contains all the fields that the JSON object from the response of the Botometer API carries, as in Figure 12. For the users that the Botometer API could not respond, the only available document fields are their “id_str”, which is a unique string identifier of Twitter users, and “display_scores” that holds the “universal” score of the user that was set to the value of 5.1. The tweets collection holds 2.1 million documents and the users collection holds 59.3 thousand documents, that will result in the creation of a very big dataset.

---

7 https://www.mongodb.com/
After storing the database, indexes were created for both of the collections, to help speed up querying time later for the creation of the dataset. Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect. Indexes are special data structures that store a small portion of the collection’s data set in an easy to traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field. The ordering of the index entries supports efficient equality matches and range-based query operations. In addition, MongoDB can return sorted results by using the ordering in the index [44].

MongoDB also provides a Graphical User Interface (GUI) called MongoDB Compass8, that allows for exploration and manipulation of data, that has proven very helpful in visualization and analysis of documents.

4.1.4 Programming Language and Libraries

The most broadly used programming language for Machine Learning is Python9, because apart from being simple and easy to learn, offers multiple Machine Learning libraries and frameworks, is portable and extensible, and has community and corporate support [45]. For that reasons, Python 3.7.5 was selected for the development process of the system. Along with Python, some popular libraries that were used are:

Pandas10 is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis manipulation tool available in any language.

Pymongo11 is a Python distribution containing tools for working with MongoDB and is the recommended way to work with MongoDB from Python.

Sklearn12 is a Machine learning module for Python, integrating classical machine learning algorithms in the tightly knit world of scientific Python. It aims to

---

8 https://www.mongodb.com/products/compass
9 https://www.python.org/
10 https://pandas.pydata.org/
11 https://api.mongodb.com/python/current/
12 https://scikit-learn.org/stable/
provide simple and efficient solutions to learning problems that are accessible to everybody and reusable in various contexts: Machine Learning as a versatile tool for science and engineering.

Re\textsuperscript{13} is a module that provides regular expression matching operations.

NumPy\textsuperscript{14} is the fundamental package for scientific computing with Python.

4.1.5 Feature Extraction and Dataset

The information that was provided from the Botometer API, as described in Section 4.1.2, is a score that is based on the likelihood of an account being a bot. The basic idea for taking advantage of this information was that a tweet’s score should be analogous to that of its author. In other words, in the context of this thesis, the dataset is comprised of tweets, where each one has a score that represents the likelihood of it being authored by a bot account. In that way, if a second account follows a similar behavior (e.g. similar way of tweeting, common profile characteristics) it should also have a high bot score.

In the case of this thesis database, the knowledge about an account’s characteristics comes from the documents of the tweets collection, which means that the information flow begins from the tweet and not from its author/user. In other words, many tweet documents might belong to the same author/user in the database, but not all of a user’s tweets are available, because the request made to the Twitter API in order to collect the data was tweet-based and not user-based. This results in a limited set of features that can be extracted to create the dataset, and thus, features that combine information from a user’s multiple tweets properties (e.g. total number of hashtags of all tweets) cannot be used in this case. Also, graph features could not be extracted in this case, since information about the following activity of the user is not available in the created database.

By consulting the reviewed works in Section 2.1 and by observing Table 2, the 37 in total features that were selected for this dataset were:

- number of URLs in tweet (content)
- number of words in tweet (content)
- number of characters in tweet (content)
- number of hashtags in tweet (content)
- number of symbols in tweet (content)
- number of numeric characters in tweet (content)

\textsuperscript{13}https://docs.python.org/3/library/re.html
\textsuperscript{14}https://numpy.org/
• number of users mentioned in tweet (content)
• number of media in tweet (content)
• number of URLs per words in tweet (content)
• number of hashtags per number of words in tweet (content)
• number of times the tweet has been retweeted (content)
• number of times the tweet has been favorited (content)
• sensitive tweet (content)
• number of followers (user)
• number of followees (user)
• number of followers per number of followees (user)
• number of followees per number of followers (user)
• number of tweets (user)
• number of public lists the user is a member of (user)
• number of tweets the user favorited (user)
• default profile (user)
• default profile image (user)
• verified (user)
• profile location (user)
• profile URL (user)
• profile description (user)
• number of characters in username (user)
• number of characters in screen name (user)
• number of characters in description (user)
• number of numeric characters in username (user)
• number of numeric characters in screen name (user)
• number of hashtags in username (user)
• number of hashtags in description (user)
• number of URLs in description (user)
• bot word in username (user)
• bot word in screen name (user)
• bot word in description (user)

For the extraction of each of the aforementioned features, along with the class/score for each instance, the appropriate functions were developed in Python code, and the complete dataset was created using a DataFrame, a data structure provided by the pandas library, which was later stored in a file created by pickle, a Python module that enables objects to be serialized to files on disk and deserialized back into the program at runtime\textsuperscript{15}.

\textsuperscript{15} https://fileinfo.com/extension/pkl
In order to strengthen the accuracy of the predictions of the model, it was argued that manually labeled data from other datasets should be added to the dataset. To that end, three datasets provided by the authors in [17] and [18] were used. These datasets contain 1992 Twitter ids of users and the prediction of the model developed in the corresponding work, as in bot/human. Using the Twitter API, the ten most recent tweets of each user, if its account was still active, were obtained, and the features that were mentioned above were extracted to create a dataset of 17554 tweets. Half of the dataset, 8779 rows, was used to train a Random Forest classifier. Each tweet/row was assigned a score of 0 if its author was previously classified as human, or 1 if its author was classified as bot. The Classifier was optimized using Grid Search and 5-fold cross-validation and reached $\approx 0.85$ F1 Score. The other half of the dataset, 8775 rows, was used to get the score of each tweet from the classifier as a probability $[0.0 - 1.0]$ multiplied by 5 to match the scores that the Botometer API produces. Eventually, to make the final dataset, this classified dataset was merged into the one created previously using the Botometer API. The final dataset$^{16}$ consists of 2,074,627 rows and is publicly available.

4.1.6 Model

In most cases encountered in Section 2.1 the nature of the problem was predicting the class in which an instance belongs to. In other words, whether a Twitter account was a bot/spam or non-bot/non-spam. This is characterized as a binary problem, since there are only two classes available. The method that is followed in such cases is that of classification, and more specifically binary classification. In other cases, when more than two classes exist, the suggested method is multiclass or multinomial classification.

However, in the case of this thesis, the problem is different. The purpose is predicting a score in the range of $[0-5]$ for every instance. This problem could not be approached by applying a multiclass classification algorithm, since the number of classes that could exist is not finite. On the contrary, the main task is approximating a mapping function ($f$) from input variables ($X$) to a continuous output variable ($y$) [46]. This task in Machine Learning is called regression.

The model that was selected after experimentation, as described in Chapter 5, was a Stacking Regressor. Stacking is an ensemble learning technique to combine multiple regression models via a meta-regressor [47]. Essentially, the output of individual estimators is used as input of a final estimator. The two estimators that were chosen for the final model are a Random Forest Regressor, with a maximum depth of trees value of 13, and a Histogram-based Gradient Boosting Regressor, with a L2 regularization parameter. The final estimator that was chosen is RidgeCV, a Ridge

$^{16}$ https://bot-detective.csd.auth.gr/datasets
**Regressor** with built-in cross-validation. Sklearn provides all the previously mentioned algorithms in Python. Thus, a programming script was developed to implement the model and fit it to the dataset, and then store it in a .pkl file, in order for it to be used for multiple predictions in the future.

A Random Forest Regressor is a meta estimator that fits a number of decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size, but the samples are drawn with replacement [48].

A Gradient Boosting Regressor builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function [49].

Ridge Regressor is exceptionally good, when features are co-dependent, as in this case where the features are the outputs of two algorithms that were trained on the same dataset.

The metric that is used to evaluate the quality of the predictions in regression, is MSE, Mean Squared Error regression loss, which measures the average of the squares of the error, as in the average squared difference between the estimated values and the actual value [50]. The final model achieved a \(\sim 0.60\) MSE score.

### 4.1.7 Explainable Model

As stated in Section 3.5, LIME\(^{17}\), a Python library, was used in order to make the predictions of the developed model explainable. Firstly, the LimeTabularExplainer module was used to create the explainer. It receives as input the trained dataset instances with their respective class/scores, the labels of the features and the indexes of the features that are categorical. The explainer was stored to be used in the development of the final system. The explainer can then receive the trained model and any instance for prediction. The output of the explainer is a list containing the weights of the features around the predicted instance. Those weight values can either be negative, zero or positive. A negative value indicates that the feature affected the model into predicting a low score (non-bot account), while a positive value indicated that the feature affected the model into predicting a high score (bot account). Features with a zero-weight value did not affect the model about the prediction of the specific account’s score.

After collecting the lists of weights for all 20 tweets retrieved for the prediction of the account, the intersections of the positive and negative values accordingly are

\(^{17}\) [https://github.com/marcotcr/lime](https://github.com/marcotcr/lime)
computed separately. The two remaining sets contain the weights of the features that affected the model’s predictions about the tweets.

Even though the results of the explainer provide insights about the decisions that the model made for the overall prediction, it was argued that those explanations could be improved by presenting them in a more detailed manner. For this purpose, a programming script was developed in Python, able to justify the weight of each feature and present the results in natural language, produced by manual extraction of rules. To make these explanations easier to comprehend, the mean value of each feature was incorporated into the text, which was computed for accounts with medium to high score (bot-like), including the accounts that were assigned with a score of 5.1.

4.1.8 Server Development

The architecture of the developed system follows the client-server model. A server is a computer program or a device that provides functionality for other programs or devices, called clients. A single server can serve multiple clients, and a single client can use multiple servers. A client process may run on the same device or may connect over a network to a server on a different device.

Client–server systems are today most frequently implemented by (and often identified with) the request–response model: a client sends a request to the server, which performs some action and sends a response back to the client, typically with a result or acknowledgement [51].

The server’s functionality was programmed in Python. The prediction function requires as input a Twitter @username. Using the Tweepy\(^{18}\) library, a Python library that provides access to the Twitter API, and the Twitter developer’s account credentials, a request is made to the Twitter API regarding the account’s profile and tweeting information. After experimentation, information about the 20 most recent tweets is requested, in order for the Machine Learning model to be able to make accurate predictions about the account’s state, whilst keeping the computational time low. It is possible to receive all of an account’s tweets information, but it would be very time consuming to process each tweet object.

After retrieving the necessary data, the features are extracted, as mentioned in Section 4.1.5, and a dataset of 20 rows is constructed (each row consists of features of a single tweet), using a DataFrame. Then the explainable model that was developed in Section 4.1.7 receives the dataset along with the trained regression model, developed in Section 4.1.6, and generates explanations about the features of the collected account’s tweets in natural language.

\(^{18}\) https://www.tweepy.org/
Based on the fact that the implementation of the system was developed using Python, it made sense to develop the server in Python as well. In order to do so, the Flask framework was used, a web framework that allows building web applications. The server exposes a single route, the /predict/:USERNAME, which requires a Twitter account’s username. Additionally, in order for someone to be able to make requests to the API, they must own a Twitter account and add their access tokens generated by Twitter to the endpoint as parameters. The request returns the predicted score of the account’s state and explainable results about the prediction in JSON format. The API is publicly available as well.

4.1.9 User Interface

The system’s User Interface was developed using the JavaScript web programming language (NodeJS), and the ReactJS library. The design of the website was made using the Semantic UI framework integration for React. The one-screen clean design of the application opts for a better user experience. The landing screen includes an input form, where users can subject Twitter @usernames. When submitting an input, the application makes a request to the server on the background. The server returns the results, that are then presented to the user in the screen. The web application’s name is Bot Detective, and a live version can be found here.

4.2 Usage

The overall structure of the system is fairly simple. A good practice when designing User Interfaces is keeping everything simple and hiding all complexity from the user. The landing screen of the web application can be viewed in Figure 13:

---

19 https://www.palletsprojects.com/p/flask/
20 https://bot-detective.csd.auth.gr/use
21 https://reactjs.org/
22 https://react.semantic-ui.com/
23 https://bot-detective.csd/
Using the app requires logging in with a Twitter account and granting Bot Detective access permissions, for it to make requests on the Twitter API on behalf of that account. The API requests are read-only, they make no changes to the account.

The users can provide an input @username and receive detailed explanatory results about the prediction score of that account. An example of usage can be viewed in Figure 15.
Figure 15: Example of Usage

Bot Detective

Bot Detective checks the activity of a Twitter account and assigns it a score based on the likeliness of the account being a bot. Higher scores are more bot-like. Exploded bots are detected per Twitter account with emphasis on the most relevant. Use of this service requires Twitter authentication and permission (TOS). If something is not working or you have questions, please contact us only after reading the FAQ.

Bot Detective is an ongoing project of the Missing at Dissemination University of Thessaloniki, Greece. Team members: Maria Kavouli - Bois Dieudiste - Athena Vasil.

Check if a Twitter account is a bot

BotID? Check account

Score: 4.1

Why bot?

- Low number of tweets, the account has favorited (1). Non-bots usually favorize 6224 tweets on average.
- Small number of retweets indicates that a tweet is more probable to have been produced by a bot account.
- This account is not verified. While this does not say a lot, if it were, it could increase the certainty that they are not a bot.
- There are special words in the account's screen name that indicate a bot-like account (e.g., "bot").
- This account's description is very small in length (0). Bots have less than 63.2 characters on their description, on average.
- This account has a default profile. This is on par with 66% of bots, on average.
- This account has not set a URL on their profile. Most non-bot users do.
- This account is not a member of many lists. Most non-bots belong to lots of lists. This account belongs to 3 lists.
- This account's number of numeric characters in their screen-name is suspicious (1 character).
- This account mentions other accounts frequently (1.0 accounts per tweet). Bots usually have 0.87 mentions in their tweets, on average.
- This account's number of numeric characters in their name is suspicious (1 character).
- This account does not have a URL in their profile's description. Most non-bot users do.
- This account uses numeric characters frequently (1.15 characters per tweet). Bots usually have 1.66 numeric characters per tweet, on average.

Why NOT bot?

- Average number of followers (7). Bots usually follow 3658 accounts on average.
- Average number of followers (138). Bots usually are followed by 3469 accounts on average.
- This account's number of tweets is rather large. This occurs mostly in non-bot accounts.
- This account shares their location on their profile. Most bot accounts do not.
- This account's followers to followees ratio is low (19.71), which is normal. Bots usually have 34.58 followers per followees, on average.
- This account's screen name length is normal (7 characters). Bots have 11.3 characters on their name, on average.
- This account does not have hashtags in its profile description. Only 31% of non-bot accounts have hashtags in their profile descriptions.
- This account's followees to followers ratio is not very high (0.05), which is normal. Bots usually have 3.91 followers per followees, on average.
- Normal number of hashtags on tweets. Bots usually have 3.48 hashtags on their tweets and this account has 2.
- Normal amount of hashtags per words in tweets. Bots usually have 0.24 hashtags per words in their tweets and this account has 0.
- Normal average number of URLs per tweet (0.45). Bots usually have 0.55 URLs per tweet.
- This account's tweets are rather small in length (14). Non-bots, usually tweet small pieces of text.
- This account's name length is normal (13 characters). Bots have 12.3 characters on their name, on average.
The predicted score measures the likelihood of an account being a bot in the scale of [0-5]. The higher the score, the more likely the account is a bot. The list of explanations is generated based on the features’ weights of the 20 instances that the explainer produced, according to the regression model, as described in Section 3.5.

In order to collect user feedback about the application, a “Feedback” button option was also included. By clicking this button, a modal opens up on the screen and requests from the user to select a category for the inputted account (Bot, Human,
Human Using Automation, Organization). A list of checkboxes is also included to urge the user to evaluate the explanations about the model’s prediction. An example can be viewed in Figure 16 above. Then, by clicking the “Send” button, the user feedback, along with the prediction and the tweets that were collected in order to get the predictions about the account in question, are stored in a MongoDB database. These data will be available for further research and for evaluation about the developed model.

Additional features that Bot Detective provides are the option of viewing the account’s in question profile, tweeting through the app and getting information and statistics about an account as viewed in Figure 17:

![Figure 17: Account Details](image-url)
Chapter 5
Experimentation and discussion

This Chapter covers the specific experimentation details that was performed during the implementation process. Specifically, it focuses on the details of the Machine Learning pieces of the infrastructure, discussing in depth their rationale alongside the required results after using the aforementioned dataset that is available. Moreover, a comparison is being made of different ways to approach the problem, with arguments for and against them.

5.1 Feature Selection

One of the first challenges of the implementation process was selecting the features to use in the dataset. As it was stated in Section 4.1.5, including features that combine information from multiple tweets of the same author is not possible, since the number of tweets available for a single author in the database varies. Also, extracting some specific features turned out to be very computationally expensive. For instance, initially it was hypothesized that including the feature “number of duplicate tweets” would result in more accurate predictions, since bots have the tendency to repost tweets with the same content. However, comparing every new instance’s text with 2.1 million other tweets’ text from the database would not be completed in a desirable timeframe. On top of that, the computational cost would also hinder the real-time application’s usage, because in order to convert a new tweet into a feature vector every tweet produced by its author must also be retrieved. Overall, special thought was put into selecting features whose calculation would be computationally cheap, because the end goal is for the model to be used in a real-time web application.

5.2 Model Selection and Hyperparameters Optimization

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters and have to be tuned so that the model can optimally solve the machine learning problem. Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data. The objective function takes a tuple of
hyperparameters and returns the associated loss. Cross-validation is often used to estimate this generalization performance.

The traditional way of performing hyperparameter optimization has been Grid Search, or a parameter sweep, which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set. Since the parameter space of a machine learner may include real-valued or unbounded value spaces for certain parameters, manually set bounds and discretization may be necessary before applying grid search [53].

The Sklearn’s GridSearchCV was applied in order to finetune the hyperparameters. The grid search provided by GridSearchCV exhaustively generates candidates from a grid of parameter values specified with the param_grid parameter [54].

The initial approach was to use a Random Forest Regressor. As mentioned in Section 4.1.6, Random Forests are an ensemble of multiple sub-regressors, and specifically of Decision Trees. As an ensemble method, it usually provides better results than standalone models. Moreover, this model was chosen because it is regarded one of the best algorithms to use by the community of researchers. According to Kaggle, the biggest community of data scientists and Machine Learning practitioners, Random Forests, alongside XGBoost, are “winning practically every competition in the structured data category” [55]. After some initial experimentation with a small sample of the dataset, it was observed that different values of the model’s hyperparameters, did not affect its results significantly. However, exhaustive search was done on the algorithm’s two main configurations; the number of estimators that were used and the max depth of each one of them. For the first hyperparameter, different values in the range 100 to 1000 were used, whereas for the second, values in the range 5 to 13, alongside the option of the tree having pure leaves, were chosen. After fitting the model with all available permutations of the above, using 5-fold cross validation, the result was a random forest comprising of 100 decision trees with max depth of 13 nodes.

The results of the prediction can be seen on the plot in Figure 18 below. As is evident, when comparing the tweets with the theoretical best red line, the regressor was able to capture the general form of the data successfully. However, the resulting MSE of $\sim0.94$, almost 20% of the scale, did not seem to be satisfactory enough.
Continuing, in an effort to improve results, a Gradient Boosting algorithm was used. As mentioned, boosting is a Machine Learning technique where in each iteration the model tries to focus more and more of instances where its prediction was wrong previously. As such, it was assumed that in this case it could provide lower error scores.

As before, different values for each hyperparameter were examined, but the main focus of the exhaustive search was mainly on the type of data regularization. Regularization is a technique especially useful when the inputting data is very high in volume. The most commonly used types of regularization are L1 (or Lasso) regularization and L2 (or Ridge) regularization. The first one deals mainly with the number of features whereas the second with the number of data instances. All three cases (L1, L2, neither of the two) were tested and, as expected, L2 was the one to provide the best results. Specifically, the fitted model achieved an MSE of ~0.78. The resulting plot is shown in Figure 19 below.

![Random Forest Prediction Results](image-url)
Combining different predictive models in order to improve the quality of the predictions is a very common method in Machine Learning. In the case of dealing with regressors, the suggested approach is using Stacking. Stacking, as stated in Section 4.1.6, trains different regressors on the same dataset and then combines them using their outputs as input to train a meta-regressor.

Benefiting from the optimized previously trained regressors, Random Forest and Gradient Boosting, those regressors applied with the same hyperparameters were used to construct the Stacking regressor. For the meta-regressor the Ridge regressor was selected, since in cases like this when the features are co-dependent, due to them being the outputs of regressors trained in the same dataset, it performs well. The model achieved an MSE of ~0.60, the best amongst the previously mentioned models’ scores. The resulting plot is shown in Figure 20 below.
5.3 Comparison of Explainable Models

Other than the Python library LIME that was mentioned in Section 3.5, another Python library was encountered during the implementation process. The library is called TreeInterpreter, and supports a very limited group of regressors and classifiers from Sklearn (DecisionTree, ExtraTree, RandomForest, ExtraTrees), so, in this case only the RandomForestRegressor could be examined. TreeInterpreter investigates which of the features are used in comparisons by the model. It allows decomposing each prediction into bias and feature contribution components. For a dataset with n features, each prediction on the dataset is decomposed as:

\[
\text{prediction} = \text{bias} + \text{feature}_1\text{contribution} + \ldots + \text{feature}_n\text{contribution}
\]

On the other hand, Lime is a model-independent method which is based on local linear approximation. It firstly samples instances around X, the instance being explained, and weights them according to their proximity to X. Then learns a linear model that approximates the model well in the vicinity of X, but not necessarily globally.

---

24 https://github.com/andosa/treeinterpreter
Comparing the two is not feasible, since they are fundamentally doing quite different things. Lime considers a feature important if by altering it a bit, this then affects the model into resulting in a different prediction. TreeInterpreter considers a feature important if it was compared to a threshold in one of the tree nodes and this resulted in a split that drastically changed the end prediction.

Even though TreeInterpreter was able to produce explanations about the features’ weights, it wasn’t applied to the end model because it is not compatible with the Stacking Regressor from Sklearn, which was used for the model implementation.
Chapter 6
Conclusions and Future Work

In this Chapter the conclusions that arise from the results of the previous Chapters of this thesis are presented, alongside with possible research paths for future work, in the context of bot detection in social media platforms through using Machine Learning.

6.1 Conclusions

The present thesis was focused on the issue of bot detection in social media platforms using Machine Learning. More specifically, this work attempts in explaining the decisions made from Machine Learning models about their results of identifying accounts in Twitter.

In the Introductory Chapter 1, firstly, a background on the subject of spam/bots and the effect on social media was presented. It was highlighted that the issue does not only lie in identifying spam/bot accounts, but it extends in justifying the reasons behind why an account is characterized as a bot. In Chapter 2 the literature on spam/bot detection using Machine Learning was reviewed, and the different approaches on the subject were discussed and compared according to their suggested methods and features. In the end of the Chapter, the opportunities that emerged were presented, leading to the purpose of the present work in extending the related work on the field of Machine Learning, by developing an explainable Machine Learning model for an online web application. In Chapter 3, the methodology of the proposed approach was analyzed in detail, describing the flow of the overall system and the connection between its consisting components. Continuing, in Chapter 4, the procedures on how the methodology was applied in practice during the implementation phase was described and details about the development decisions were also stated. Lastly, in Chapter 5, ideas that were considered and experimentation about the developed models were presented and compared.

Overall, the creation of a novel dataset, the development of an explainable model that produces accurate results and the deployment of a web application available to the public, compose the accomplishments and contributions of this thesis.

Not only the research community will benefit from the addition of this work on the subject of bot detection using Machine Learning, but it is also important for ordinary users of social media platforms, regardless their technological background, to have a tool as their ally in identifying bots in their everyday interactions in the network.
6.2 Future Work

The completion of this work gave birth to a number of ideas for further research on the subject of this thesis. Indicatively listed areas could be:

- Expanding the created dataset with tweets regarding other trending topics of Twitter. Trending topics are usually topics that bots select to tweet about, in order to conceal their intentions in the network.
- Integrating the proposed methodology to other social media platforms. It would be interesting to observe the different features and behavior of bots in other platforms and examine the predictions that the developed model would produce.
- Figuring ways of decreasing the resulted MSE score for the model to make more accurate predictions. More accurate predictions could potentially result in better explainability of the model.
- Incorporating to the proposed methodology graph-based features. For instance, introducing features that could be extracted from a graph created locally around the account in question, regarding their connections or interactions with other accounts in the network.
- Collecting the input @usernames from the User Interface tool and refitting the model with a new dataset.
- Conducting surveys about the User Interface, Bot Detective, to get feedback from its users and improve the user experience accordingly.
- Expanding the User Interface’s functionality to comply with users’ needs. For instance, providing the users with the functionality of performing graph analysis around an account.
References


[12] "Rights related to automated decision making including profiling," [Online]. Available: the GDPR also: requires you to give individuals specific information about the processing; obliges you to take steps to prevent errors, bias and discrimination; and gives individuals rights to challenge and request a review of the decision..


the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009.


