

Instituto Tecnológico y de Estudios Superiores de Monterrey

Monterrey Campus

School of Engineering and Sciences



**Anomaly Detection as a Method for Uncovering Twitter Bots**

A thesis presented by

**Javier Israel Mata Sánchez**

Submitted to the  
School of Engineering and Sciences  
in partial fulfillment of the requirements for the degree of

Master of Science

in

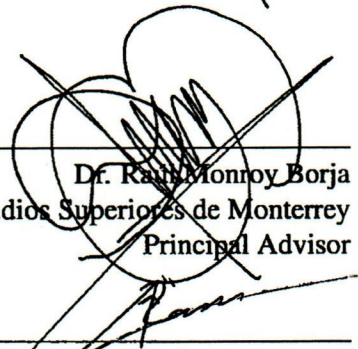
Computer Science

Monterrey, Nuevo León, November, 2019

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

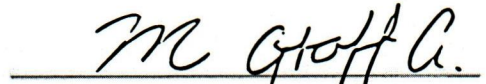
The committee members, hereby, certify that have read the thesis presented by Javier Israel Mata Sánchez and that it is fully adequate in scope and quality as a partial requirement for the degree of Master of Science in Computer Sciences.



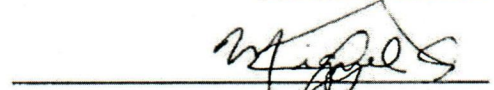
Dr. Raúl Monroy Borja  
Instituto Tecnológico y de Estudios Superiores de Monterrey  
Principal Advisor



Dr. Jorge Rodríguez Ruiz  
Instituto Tecnológico y de Estudios Superiores de Monterrey  
Co-Advisor



Dr. Mario Graff Guerrero  
Infotec  
Committee Member



Dr. Miguel Ángel Medina Pérez  
Instituto Tecnológico y de Estudios Superiores de Monterrey  
Committee Member



Dr. Rubén Morales Menéndez  
Associate Dean of Graduate Studies  
School of Engineering and Sciences

Monterrey, Nuevo León, November, 2019

# Declaration of Authorship

I, Javier Israel Mata Sánchez, declare that this thesis titled, *Anomaly Detection as a Method for Uncovering Twitter Bots* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- An article titled *A One-class Classification Approach for Bot Detection on Twitter* [100] was derived from this thesis work; its authors are Jorge Rodríguez-Ruiz, Javier Israel Mata-Sánchez, Raúl Monroy, Octavio Loyola-González, and Armando López-Cuevas. The article was submitted to the *Computers & Security Journal*, and it's currently under its second review. Section 2.2 of this present work lends material from the mentioned article. Moreover, I had the opportunity to contribute importantly to the creation of such material, which is fundamental for the appropriate structuration of the thesis.



---

Javier Israel Mata Sánchez  
Monterrey, Nuevo León, November, 2019

©2019 by Javier Israel Mata Sánchez  
All Rights Reserved

# Dedication

The process of earning a Master's degree has been more enriching than I could have ever imagined. I feel blessed for having this opportunity; everything has happened in the best way and at the best time, or at least it feels that way.

This path has been one of hard work and much discipline, but also of constant happiness. I want to thank especially my family, friends, and teachers for supporting me at all times, good and bad.

Likewise, I want to extend my deepest gratitude to all the people who continuously contribute to the existence and maintenance of free, good-quality online education. Thanks to them, millions of humans around the globe have had the opportunity to continue their studies and obtain valuable knowledge. I firmly believe that this type of education helps to improve the life quality of many people around the world, including me, and, therefore, contributes to the creation of more united and harmonious societies.

# Acknowledgements

First of all, I would like to thank my advisors, Raúl Monroy, and Jorge Rodríguez, for their patience and guidance. Second, I express my gratitude to Miguel Angel Medina and Octavio Loyola, as they were a vital part of this process; their feedback throughout the investigation process helped me to increase the quality of this research.

Finally, the support of Tecnológico de Monterrey made all this work possible, as they provided me with a 100% scholarship for my master's degree studies. Also, a special mention to CONACyT, for supporting me with living expenses during my studies. I feel honored to have had this opportunity; thank you all for contributing to the generation of knowledge and enhancement of education quality in the country.

# **Anomaly Detection as a Method for Uncovering Twitter Bots**

by

**Javier Israel Mata Sánchez**

## **Abstract**

During the past decades, online social networks (OSNs) have steadily grown to become the mainstream communication channels they are today. One of the most popular OSN is Twitter, a micro-blogging platform, which by 2019, had approximately 139 million daily active users. Interestingly enough, a relevant portion of the accounts registered in this social network is not human. Researchers have found that approximately 15% of all Twitter accounts, which is close to 48 million users, exhibit an automated behavior. Such automatically managed accounts are called bots. Bots have exhibited a diversity of behaviors, and therefore, of objectives. Some good uses of bots include automatically posting information about relevant news and academic papers, and even to provide orientation during emergencies. Unfortunately, malicious bots are also abundant. These types of bots have been used to distribute malware, send spam, and even to affect political discussions negatively. Moreover, malicious bots have also promoted terrorist propaganda and online extremism proselytism.

Diverse bot detection methods and tools have been developed by researchers and by the social network companies themselves. Although there exist unsupervised learning bot detection methods, most of the state-of-the-art bot detection mechanisms make use of supervised learning machine learning algorithms. Due to their nature, these methods require examples of different bot types to detect them effectively. Nevertheless, obtaining examples of all the different bot types present on Twitter is not a trivial task. Moreover, bots are continuously *evolving* to evade current detection mechanisms.

This thesis proposes to approach the Twitter bot detection problem by making use of one-class classifiers. Classifiers of this type only require examples of a *normal* class to detect anomalous behavior, thus are capable of overcoming the limitations of state-of-the-art methods. The experiments developed in this work demonstrate to what extent multi-class and binary classifiers are different from one-class in terms of performance. Also, the significance of these differences is measured. Results show that one-class classifiers yield higher and more stable performance than the other two types of classifiers when detecting bot types that were not used in their training phase of the algorithm. Additionally, the difference in performance is statistically significant. On the other hand, binary classifiers perform better than one-class, when detecting bots of a type that was present in the training phase of the algorithm.

Given our results, one-class classifiers could serve as an early-warning system, detecting anomalous patterns of account behavior, which could represent a new bot. The results presented in this work can also contribute to the development of hybrid systems that combine features of binary-classifiers with the benefit of one-class methods. Such systems would represent a step towards broadening the protection of OSNs' users from malicious bots, therefore benefiting a primary part of society.

# List of Figures

1.1	Number of total social network users (in billions) from 2010 to 2018.	2
2.1	Confusion matrix layout.	8
3.1	Classification area, with three classes of normality, learned by a one-class classifier.	18
3.2	Region of normality learned by a one-class classifier.	18
3.3	Classification regions learned by a one-class classifier. Colors represent different degrees of membership to the normal class, being the lightest an indicator of a 0% degree of membership to the normal class.	19
3.4	Comparison of classification models learned by a one-class and a binary classifier. Figure inspired by Rodríguez et al. [101].	21
4.1	Data partitions performed by a 10-fold cross-validation procedure.	36
4.2	Example of a Critical Difference diagram.	37
5.1	Performance of binary classifiers when using social1 bots in the training phase, and different bots at testing.	40
5.2	Performance results of the experiment set A when using social2 bots in training.	41
5.3	Performance results of the experiment set A when using social3 bots in training. Note that Logistic Regression yield the second-best performance, only outperformed by PBC4cip.	43
5.4	Performance results of the experiment set A when using traditional bots in training.	44
5.5	Box and whisker plots of the AUC performance, per bot type, for each table in Experiment Set A. The plots are labeled in order of appearance.	45
5.6	CD diagrams showing the significant statistical differences among the different types of bots. Each diagram corresponds to a table in experiment set A, in order of appearance.	46
5.7	Performance results of the experiment set B.	47
5.8	CD diagram of the performance in experiment set B. In this experiments, classifiers are trained with three types of bots and tested with the remaining one.	48
5.9	Performance results of the experiment set C.	49
5.10	CD diagram of one-class classifiers' AUC performance for Twitter bot detection.	50
5.11	CD diagram of the two best multi-class and one-class classifiers used in the Twitter bot detection experiments.	52

# List of Tables

2.1	Previous works comparison.	13
3.1	Datasets released by Cresci [38].	25
3.2	Features provided for each tweet in the selected Cresci's datasets.	27
3.3	Features provided for each user account in the selected Cresci's datasets.	28
3.4	Feature vector extracted for each Twitter user.	29
5.1	Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type social1 account examples. A bold typeface is used to convey the results of testing the same type of bots used in the training set.	39
5.2	Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type social2 account examples. A bold typeface is used to convey the results of testing the same type of bots used in the training set.	41
5.3	Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type social3 account examples. A bold typeface is used to convey the results of testing the same type of bots used in the training set.	42
5.4	Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type traditional account examples. A bold typeface is used to convey the results of testing the same type of bots used in the training set.	44
5.5	Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of different types. A bold typeface is used to convey the best AUC obtained by a classifier when testing a specific type of bot.	47
5.6	Results of testing different types of bots when classifiers have used a training dataset containing legitimate users examples. A bold typeface is used to convey the highest performance attained by a classifier per testing dataset. BTPM and BRM where used with two different measures: Euclidean(E) and Mahalanobis(M)	49



5.7	Comparison of binary and one-class classifiers' performance. In each row, bold typeface numbers are used to convey the highest AUC performance obtained for the given training/testing set combination. PBC4cip is the best-performing classifier, nevertheless, it is marked with gray bold typeface, as it is analyzed separately at first. . . . .	51
-----	---	----

# Contents

<b>Abstract</b>	v
<b>List of Figures</b>	vi
<b>List of Tables</b>	viii
<b>1 Introduction</b>	<b>1</b>
1.1 Twitter	1
1.2 Twitter Bots	3
1.2.1 Malicious Uses of Bots	3
1.2.2 Impact of Social Twitter Bots	3
1.2.3 Bot Detection	4
1.3 Hypothesis	5
1.4 Objectives	5
1.5 Contributions	6
1.6 Document Structure	6
<b>2 State-of-the-art Methods for Bot Detection on Twitter</b>	<b>7</b>
2.1 Prevalent Performance Measures in Bot Detection Research	7
2.2 Previous Works	9
2.2.1 Supervised Bot Detection Approaches	11
2.2.2 Unsupervised Bot Detection Approaches	12
2.3 Implications of the Evolution in Bots' Behavior	14
2.3.1 Paradigm-shift in Bot Design	14
2.3.2 Relationship Between Bot Behavior and Classifiers Performance	14
2.4 Summary	15
<b>3 One-class Classification for Twitter Bot Detection</b>	<b>16</b>
3.1 Fundamental Approaches for Anomaly Detection	16
3.1.1 Unsupervised Clustering	16
3.1.2 Supervised Classification	17
3.1.3 Semi-supervised Classification	17
3.2 Classification in Machine Learning	20
3.2.1 Training and Testing a Supervised Classification Model	20
3.2.2 One-class, Binary, and Multi-class Classification	20
3.2.3 Measures to Compare Classification Performance	22

3.3	Advantages and Limitations of Existing Approaches for Bot Detection in Twitter	22
3.3.1	The Supervised Approach for Bot Detection	22
3.3.2	The Semi-supervised Approach for Bot Detection	23
3.4	Methodology	23
3.4.1	Experiments	24
3.4.2	Datasets Used in the Experiments	25
3.4.3	Features for Characterizing Twitter User's Behavior	26
3.4.4	Selected Classifiers	26
3.4.5	Selected Performance Measure	32
3.5	Summary	33
<b>4</b>	<b>Experimental Setup</b>	<b>34</b>
4.1	Experiments	34
4.1.1	Experiment Set A (Binary Classifiers) Description	34
4.1.2	Experiment Set B (Multi-class Classifiers) Description	35
4.1.3	Experiment Set C (One-class Classifiers) Description	35
4.2	Performance validation	36
4.3	Testing the significance of results	36
4.4	Summary	37
<b>5</b>	<b>Experimental Results and Discussion</b>	<b>38</b>
5.1	Results of Experiment set A (Binary Classifiers)	38
5.1.1	Classifiers Trained with Social1 Type Bots and Genuine Accounts	39
5.1.2	Classifiers Trained with Social2 Type Bots and Genuine Accounts	40
5.1.3	Classifiers Trained with Social3 Type Bots and Genuine Accounts	40
5.1.4	Classifiers Trained with Traditional Type Bots and Genuine Accounts	42
5.1.5	Summary of Results of Experiment Set A	43
5.2	Results of Experiment set B (Multi-class Classifiers)	45
5.3	Results of Experiment set C (One-class Classifiers)	48
5.4	Results Comparison	50
5.4.1	Statistical Differences Among Approaches	52
5.4.2	Generalization Capabilities	52
5.5	Summary	53
<b>6</b>	<b>Conclusions</b>	<b>54</b>
6.1	Future Work	56
6.1.1	Hybrid approach	56
6.1.2	Proactive approaches	56
	<b>Bibliography</b>	<b>68</b>

# Chapter 1

## Introduction

The number of online social networks (OSNs) users has increased significantly in recent years. In 2018, the total number of OSNs accounts was approximately 2.65 billion [108]. Even though each OSN offers distinct features, most users use them with the same purpose: to share opinions, upload photos, post videos, and even consume news and articles. This digital interchange of information allows users to connect, form networks, and create communities. Since OSNs can be accessed almost anywhere in the world, and due to the vast number of accounts registered in them, most of the participants in social platforms, such as Twitter, do not know each other personally.

Social platforms allow for accounts to be automated, enabling companies and users to obtain extra benefits from all the features offered by the OSN. These automated accounts, known as *bots*, have been regulated by the OSN providers, by providing guidelines that aim to prevent the platform from being misused. Unfortunately, bots have been widely used for malicious purposes, such as to distribute spam and malware [128]. Politics and public health are domains of particular interests, as studies [21, 109, 27, 48] have shown that bots have been used to influence people's opinions during political campaigns and to generate more conversations on a vaccine debate. Due to the impact that bots can cause in society [49, 18], researchers have developed different bot detection methods. Most of the detection methods have their basis on supervised machine learning approaches [111], and even though no method has proven to be perfect, useful tools for detecting known types of bots have been developed.

### 1.1 Twitter

The number of social network users, worldwide, has shown an increase of approximately 273% from 2010 to 2018, with an estimated 2.65 billion users at the end of 2018 [108]. Figure 1.1 shows how the number of users increases every year. OSNs differ from each other in their interface and features. Nevertheless, most of them are used for the same purposes. Typical uses of social platforms include sharing comments, uploading photos, and following public figures. Users of OSNs also use them to keep up to date with the topics of their interest [33, 112, 80], and even to consume scientific articles [60]. News is also a prevalent topic of conversation in sites such as Twitter, where studies have shown that, at specific moments, approximately 85% of the most popular content in the networks has been related to headline

news [74].

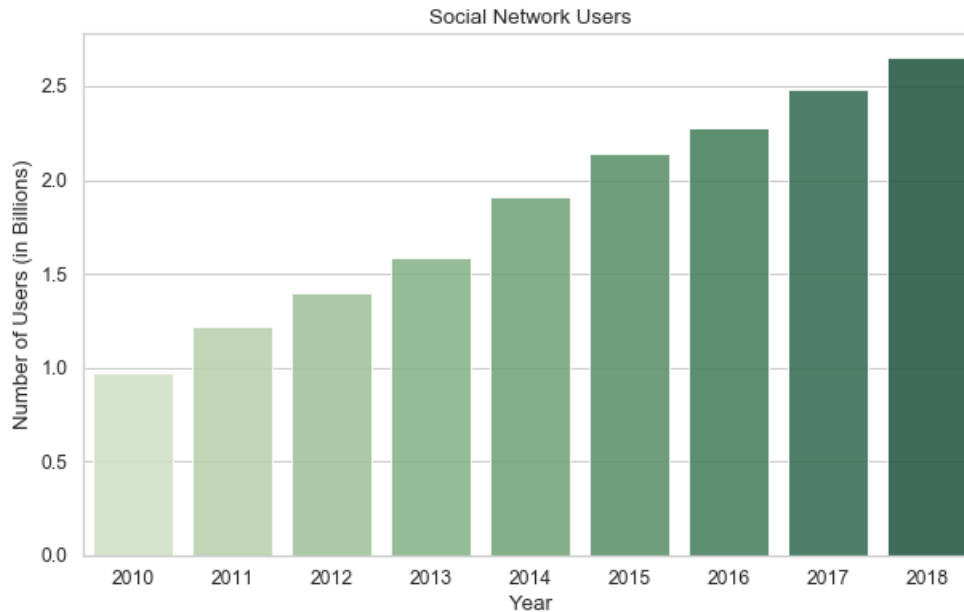


Figure 1.1: Number of total social network users (in billions) from 2010 to 2018.

Twitter<sup>1</sup> is a popular micro-blogging OSN launched in 2006 [70, 81]. The Twitter platform allows users to post messages, known as tweets. Also, they can share the content generated by other members of the network, an action called retweeting; and can follow other accounts to see their updates on the homepage. Additionally, tweets can include multimedia elements such as images, GIFs, short videos, URLs, and hashtags [115]. One of the main characteristics of this OSN is that the number of characters per publication is limited. When Twitter was originally launched, only 140 characters per tweet were allowed, but in 2017 the maximum number of characters doubled to 280 for most languages. Chinese, Japanese, and Korean languages are the exception since accounts configured with those languages stayed with a limit of 140 characters per tweet [6].

The popularity of this micro-blogging platform has increased over the years. By the second quarter of 2019, Twitter reached an average of 139 million daily active users [116]. In July 2019, SimilarWeb<sup>2</sup>, a web analytics service, ranked Twitter as inside the top six websites with most traffic<sup>3</sup>, with more than 4.03 billion monthly visits.

Users registered in this OSN can access it through a web page, mobile applications, and Application Programming Interfaces (APIs). Twitter's APIs allow a set of different actions, such as consulting public tweets and replies, create and manage ad campaigns, manage an account's profile and settings [117]. There exists a set of automation rules that outline the proper and improper use of automated accounts [118], which aim to guide users of the API.

<sup>1</sup>[www.twitter.com](http://www.twitter.com)

<sup>2</sup><https://www.similarweb.com/>

<sup>3</sup><https://pro.similarweb.com/#/industry/topsites/All/999/1m?webSource=Total>

Over the years, several applications have been developed to enhance the user's experience in the online platform. These same applications allow for accounts to be managed automatically by different programs; such accounts are known as Twitter bots [33].

## 1.2 Twitter Bots

Over the years, Twitter bots have been used to enhance the experience of users. Some common proper uses of these automated accounts include retweeting exciting and relevant content for specific communities [33, 112, 80], providing an aggregation of tweets about a specific area, and communicating with customers to increase the impact of marketing efforts [90]. Most of the Twitter bots used for legitimate purposes share only one type of content and are transparent about themselves and their motivations [129].

### 1.2.1 Malicious Uses of Bots

Bots have also been used with malicious intent. Some of the most common malicious bots are used to send spam, spread malware, and perform other illicit activities [128]. More sophisticated bots can interact with other users of the OSN. These bots, called social bots, can avoid detection, perform more complex tasks, and take advantage of human vulnerabilities. Such vulnerabilities include the tendencies of paying attention to trending topics and the fact that different social factors impede people's willingness to verify the information [69]. Social bots can adopt different strategies to impersonate human users. Impersonation can be achieved by emulating temporal patterns of content posting and consumption by humans and interacting with other users of the OSN by engaging in conversation with them [64]. A more extreme example of human impersonation done by social bots is identity theft [36]. When several bot accounts act as a group and coordinate efforts, a *Twitter botnet* is formed. Frequently, botnets are used to generate false popularity [5]. When inspected individually, accounts on a botnet may seem genuine, but when a large number of accounts act in a strongly coordinated manner, their intentions become noticeable [31].

### 1.2.2 Impact of Social Twitter Bots

The impact of social bots is not limited to a single topic or domain. Since social media represent the perfect space for people to express their concerns and points of view, bots can take advantage of this space to influence opinions. A particular domain of great importance for the government and society is health [8]. There is evidence of social bots disseminating antivaccine messages. These social bots, along with Russian trolls<sup>4</sup>, created false equivalency<sup>5</sup>, eroding public consensus on vaccination. The public health implications are significant, as directly confronting vaccine skeptics enables bots to legitimize the vaccine debate [27, 48]. Smoking is another topic where bots have been used to influence the public's perception.

---

<sup>4</sup>A troll is someone who aims to upset people on social media by posting unkind or offensive content.

<sup>5</sup>A false equivalence occurs when, in an argument, two opposing ideas appear to be logically equivalent when they are not.

Recent research suggests that bots posted between 70% and 80% of tweets mentioning e-cigarettes [35]. Moreover, studies on e-cigarette-related attitudes and behaviors have made evident the importance of distinguishing if Twitter posts come from bots or humans when attempting to understand attitudes and behavior [7, 9].

Automated accounts have achieved an impact on politics more than once. Since social media has become a widely used platform for democratic discussion on social issues related to policy and politics, social bots have been used to influence people's opinions actively. Studies around the 2016 U.S. Presidential election have found that social bots have indeed the capacity to negatively affect political discussion, which could alter public opinion and endanger the integrity of elections and democracy in general [21, 109]. Scientists found that during that period, humans and bots retweeted each other substantially at the same rate. Moreover, the most positive tweets about Donald Trump were generated by bots [129]. Again, for the 2016 U.S. presidential election, bots played a crucial role in the spread of low-credibility content [105]. Researchers have also studied the run-up to the 2017 French presidential election; during that political process, anomalous account usage patterns were found. The fact that hundreds of bots posting about the French elections also participated actively in the 2016 U.S suggests the existence of a black-market for political disinformation bots [47].

Terrorist propaganda and online extremism proselytism also represent topics of public interest that have been promoted by social bots. Ferrara et al. [49] analyzed a dataset of millions of examples of extremist tweets and discovered that social bots produced some of these content. A 2015 study [18] analyzed Islamic State-supporting accounts and unveiled that social bots were being used to spread the ideology of this terrorist organization. During the Syrian civil war in 2012, botnets were found to misdirect the online discussion [1].

A more general domain is fake news. This domain's impact does not cover a specific topic, but at the same time, it remains relevant. This relevance comes from the fact that social bots target influential accounts to amplify misinformation before it becomes viral [20, 84]. Users can also spread misinformation unintentionally. For example, after the Boston Marathon bombing, 29% of the most viral content on Twitter were rumors and fake content. However, this information was not shared by bots, but the other way around, the fake content spreaders were users with high social reputation and verified accounts [58].

### 1.2.3 Bot Detection

To prevent social bots from causing harm to society, the Twitter platform, academia, and industry have developed a wide variety of bot detection methods across the years. Nevertheless, Twitter bots differ significantly on their behavior and account characteristics, making their detection a non-trivial task. Some of the first documented attempts aimed to create groups of people to detect bots via a crowd-sourcing bot-detection platform [126]. However, since bots can replicate at a much higher rate than human experts can gather, such a method lacked from the scalability; thus, it was unviable.

The only viable way to confront automated accounts is through the use of computational tools and frameworks such as machine learning. Although unsupervised methods have also been explored [88, 31], most common bot detection approaches are based on supervised machine learning algorithms [111]. Approaches based on machine learning are not flawless, even though they are more effective than human detectors. Supervised machine learning algorithms

require many examples of labeled accounts to function correctly. Therefore, researchers have to collect Twitter accounts and then manually label them to form a useful dataset. This human factor makes the labeling process error-prone. This propensity is due to the subjectivity and lack of agreement between annotators [121]. The difference between annotators' opinions may be influenced by the lack of ground-truth about what a social bot is, which, at the same time, prevails due to the gray area between human and bot behavior [129]. Despite the obstacles present in bot characterization, practical, commercial, and useful bot-detection tools have been developed [40, 129].

Existing bot detection methods differ on the algorithms used, as well as in the features used to characterize the account's behavior. Researchers have come up with distinct categories of features, but there is no general agreement on a specific group of characteristics that work best for detecting all types of bots. The lack of consensus is mainly due to the differences in bot's behavior; each bot type shows different posting and consuming patterns. Moreover, bot designers continuously seek to avoid detection mechanisms. As a result, hybrid bot approaches that mix automated and human activity have been developed [57]. Bots designed with such approaches have proven to be harder to detect [38].

### 1.3 Hypothesis

One-class classifiers are a type of supervised machine learning classifiers that are trained only with examples of one class and are capable of determining if an unknown object belongs to the class used in the training phase of the algorithm, or not. As they can model normality, one-class classifiers are commonly used to detect anomalies. They are especially useful in domains where examples of anomalies may be scarce or non-representative. [113, 99, 29].

We hypothesize that one-class classifiers can detect instances of new classes of bots, even if those classes were not present in the training phase of the algorithm. Such a detector can be elaborated by training the one-class classifiers with examples of *genuine* behavior expressed on twitter by humans. The validation of our hypothesis would allow us to overcome the main limitation of state-of-the-art methods and complement existing approaches.

The following research questions guide this work:

- Does the performance of multi-class methods decrease when it comes to identifying bots of an unknown class?
- Do one-class classifiers perform better than multi-class methods when it comes to identifying bots of an unknown class?
- When identifying bots of an unknown class, is there a statistically significant difference between the performances obtained by the two methods?

### 1.4 Objectives

The main objective of this thesis is to improve performance in the timely detection of bots. Some of the particular objectives of this thesis include:



- Select a database of Twitter bots with a variety of bot types and adequate number of examples so that it can be used to compare different detection methods
- Design a short-length feature vector, based on the bot detection literature that captures the behavior of Twitter accounts in a way that allows distinguishing bot from human accounts.
- Define a set of experiments to test our hypothesis; these experiments should allow for a direct comparison between the performances of multi-class and one-class classifiers.
- Identify several supervised multi-class classifiers that are representative of the state-of-the-art bot detection mechanisms, to use them in the set of experiments.
- Select different state-of-the-art one-class classifiers to be used in the set of experiments, to test their performance on the Twitter bot detection context.
- Analyze the experimental results and perform parametric tests to evaluate if there exists a significant difference between the performances of multi-class and one-class classifiers.

## 1.5 Contributions

This thesis presents research that contributes to the scientific community in different ways. Some of the most relevant contributions are:

- A carefully-selected feature vector that encompasses the behavior of Twitter accounts in a way that allows distinguishing humans from bots, using supervised classification methods.
- An evaluation of one-class methods' performance in the Twitter bot detection context. To the best of the author's knowledge, no research has proposed one-class classification methods for bot detection on Twitter so far.
- A direct comparison of the performances of multi-class and one-class methods in the Twitter bot detection context, along with the significance test of their differences.

## 1.6 Document Structure

The rest of the thesis is organized as follows. Chapter 2 elaborates on the state-of-the-art methods for bot detection on Twitter, as well as the implications of the evolution of bots' behavior. Chapter 3 explains one-class classification techniques, and their application for detecting unknown types of bots on Twitter is detailed. Chapter 4 presents the experimental setup used to test our hypothesis. Chapter 5 shows the experimental results and discussion. Chapter 6 concludes this research and describes possible paths for future works regarding bot detection on Twitter.

## Chapter 2

# State-of-the-art Methods for Bot Detection on Twitter

Automated accounts present on social networks, such as Twitter, have demonstrated to be influential in a wide variety of topics that range from politics to public health debates. Aware of their influence, researchers across the world have developed their bot detection methodologies. In recent years, several bot detection proposals have emerged, being supervised classification the most common approach among them. For detecting bots, supervised approaches need features that represent and characterize the accounts. These features allow the classifier to distinguish between bots and legitimate accounts. However, there is not a standardized set of features used on previous bot detection research. Even the performances are evaluated in different manners, which does not allow for a direct comparison of the methods used. Nevertheless, from an analysis of previous works, we have obtained useful information about the features used and their merits for bot classification. These past works have served as a foundation for new detection methods and account characterizations. On the other hand, bots have also evolved. For the existing detection mechanisms to maintain their effectiveness, they need to evolve along with paradigm shifts in bot design.

Almost the entirety of the material presented in Section 2.2, which addresses the state-of-the-art in bot detection, is taken directly from the article *A One-class Classification Approach for Bot Detection on Twitter* [100]. In that article, I had the opportunity to contribute importantly to the creation of the material presented, which is fundamental in the structure of this chapter, and this thesis in general.

## 2.1 Prevalent Performance Measures in Bot Detection Research

Performance measures are used to evaluate the effectiveness of a given method. Determining a performance measure also enables a direct comparison between methods that share it. Most of the performance measures used in the bot detection literature can be derived from a table showing the relationship between observed and predicted values in a classification problem, known as a confusion matrix [91]. This matrix contains four values: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

		Predicted class	
		<i>P</i>	<i>N</i>
Actual class	<i>P</i>	True Positives (TP)	Flase Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)

Figure 2.1: Confusion matrix layout.

In bot detection literature, there are three standard performance measures used. Since there is no ranking regarding their importance, these performance measures are listed in alphabetical order. The first performance measure is accuracy, which is given by the percentage of the correctly classified bots and legitimate accounts, from all the accounts on the testing datasets.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

The second measure used is Area Under the Curve (AUC) [68] of a Receiver Operating Characteristics (ROC) curve. Two measures are needed to calculate the ROC curve: Recall and False Positive Rate (FPR). Recall, also known as Sensitivity, is the percentage of correctly predicted bot accounts of all the bot accounts in the testing dataset. FPR refers to the percentage of incorrectly classified accounts, the false positives, from all the legitimate accounts in the testing dataset.

$$Recall = \frac{TP}{TP + FN} \quad (2.2)$$

$$FPR = \frac{FP}{FP + TN} \quad (2.3)$$

The ROC is calculated by plotting the recall against the FPR, and the AUC summarizes the performance in a single numerical value. The AUC defined by one run of the algorithm is also known as Balanced Accuracy [107]. AUC can be calculated by the following formula [106]:

$$AUC = \frac{Recall + Specificity}{2} \quad (2.4)$$

Specificity measures the proportion of true negatives, over the total number of instances predicted to be negative. The following formula calculates specificity [1107]:

$$Specificity = \frac{TN}{FP + TN} \quad (2.5)$$

The third measure is the F1-score [78]. This measure is given by the harmonic mean of precision and recall, where precision is the percentage of correctly classified bot accounts from all the accounts classified as a bot.

$$Precision = \frac{TP}{TP + FP} \quad (2.6)$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (2.7)$$

Unlike accuracy, F1 and AUC have the advantage of being balanced performance measures and robust against imbalanced class problems [68, 78]. Class imbalance occurs when there are more examples of one class than another, which is common in the Twitter bot detection context [65]. Additionally, some authors [37, 82] complement the previous performance measures with the Matthews correlation coefficient (MCC) [24]. MCC is a correlation coefficient between the observed and predicted classifications. Since it takes into account true and false positives and negatives, it is suitable for evaluating class imbalance problems [24].

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (2.8)$$

## 2.2 Previous Works

Regardless of the online platform, bot detection bases on the premise that automated accounts show different behavior than accounts managed solely by humans. This premise is taken as true, disregarding of the bot's level of automation or sophistication. In Twitter, the behavior of an account is characterized using different features drawn from the account and the content it posts. Along with the different bot detection methods proposed in the literature, scientists have used distinct feature representations for yielding accurate bot detection models [38, 82, 40, 86]. Such features can be into five categories:

**Content:** Tweets can contain up to 280 characters, which can be used to post text, mention other users, post urls, and images. By parsing the text and analyzing the other elements, telltale signs of a bot can be found. One of the earliest features used was the number of urls posted, the numbers of unique urls, and the number of mentions to other users per tweet [76, 56, 4], because bots were commonly used to spam links to malicious pages using the same message but mentioning different users. The text can also be analyzed to characterize bot behavior, as the language and phrase construction may differ between bots and legitimate users [40, 121]. The construction of the phrases and words can also

be used to separate between bots and legitimate users, as the language used could be different [40].

**Sentiment:** Further analysis of the text of the tweet can be done to extract the sentiment, and measure how the poster of the tweet feels about the topic at hand. While these characteristics are related to the content, they require further analysis and do not allow for direct extraction [82]. Dickerson et al. [43] found that bots accounts express more consistent sentiments for a topic, as they are commonly used to support or oppose a topic. Sentiment features are the less used ones, [48, 82], or complete accounts depending on the average sentiment expressed [43].

**Account Information:** These features are obtained from the account used to post a tweet, and have the advantage of not suffering big changes over time, as the information does not change with every tweet. For this reason, these features can be measured with less frequency. Features such as the account age, if there was a biography or description of the account, and if the account had an image could be used to detect bots, as these accounts could be empty and be relatively new [122]. However, bot accounts sometimes stay dormant until needed, and are filled with fake information to avoid detection [38]. One of the most useful feature of this type is the friends to followers ratio as legitimate accounts who follow other humans should have a similar number of friends and followers, as human relationships tend to be reciprocal. While friends to follower ratio is a widely used feature [82, 40, 76], it can be subverted as sophisticated bots un-follow friends that do not reciprocate.

**Account Usage:** These features measure how the user or bot uses the account, and how it interacts with the Twitter service. Bots can have distinct posting patterns than a legitimate user, like posting at regular intervals, or posting in volumes that would be impossible for humans [34]. Furthermore, bots generally access the Twitter platform via the API or other ways that allow automated programs to post, while legitimate users tend to use the web or mobile interfaces [34]. Account usage features can also obtain single metrics from aggregating content ones, such as analyzing the difference between messages, as bots could post the same message to different users [76].

**Social Network:** These features measure the interactions between different accounts, as a single account may not express suspicious behavior, but when analyzed in group bots could be found. Examples of this features can be a measurement of the different in text or topic of accounts, as groups of bots tend to post about the same topic [72]; the difference in posting times of messages, as bots tend to retweet the same message in short time spans; or the sequence of actions as bots may perform the same actions in the same order while legitimate users do not [38].

Characterizing bots using features of five different categories has enabled to approach bot detection task using supervised and unsupervised classification. In the next sections, the most prominent Twitter bot detection mechanisms are discussed. For each approach, we discuss the feature representation, the classification method used, and the reported performance.

## 2.2.1 Supervised Bot Detection Approaches

Lee et al. [76] proposed one of the earliest methods for Twitter bot detection. They created a classifier called Decorate, which uses content, tweet account, and account usage features, on a dataset collected by them. This method achieved an F1 value of 0.88. Similarly, Wang [124] approaches the problem as Lee et al., but obtains the features only from the last 20 tweets in the account, and uses a naïve Bayes classifier, obtaining an F1 value of 0.91.

In a later study, Ahmed and Abulaish [4] used the Weka data mining tool [59] to test the performance of three classifiers: naïve Bayes, J48, and Jrip. By restricting each classifier to different feature subsets, they found that account usage features are more discriminative than tweet content ones. Regarding the performance of the classifiers, Jrip attained the best performance, achieving a recall of 0.987.

Most recent research, seeking to improve the performance previously obtained, have exploited other account usage features. Chu et al. [34] claimed to have achieved 96% of recall, using Random Forest [26], when including *usage* features such as how the client logs into the Twitter platform or the client regularity of tweet posting.

Yang et al. [128] improved the F1 score previously obtained by Decorate and Bayesian network classifiers by using features of type tweet usage and social network. They obtained an F1 equal to 0.9 using Random Forest, while the previously mentioned naïve Bayes and Decorate methods obtained an F1 of 0.88 and 0.83, respectively.

To optimize the amount of data that needs to be gathered, Wang et al. [125] focused on the detection of spam tweets by relying only on content features. In their experiments, two different datasets were used, but the best results were obtained using the Social HoneyPot Dataset created by Lee et al. [77]. This dataset consists of 22,223 spammers and 19,276 legitimate users, along with their most recent tweets. Random Forest was the best performing algorithm; it was capable of detecting bots with an F1 score of 0.946.

Gilani et al. [56] adopted a different methodology, which added the multimedia elements in the tweets to the account usage features. The authors separated the accounts into different datasets, depending on the number of followers. By using a Random Forest classifier, he achieved an F1 close or equal to 1.0 in the best case, finding bots accounts with more than 10 million followers. In these experiments, they found that, with their feature representation, it is more difficult to detect a bot with less than one thousand followers. In such cases, Gilani et al. method obtained an F1 of 0.84.

Cresci et al. [37] proposed a DNA-like analysis of the tweet usage account. They hypothesized that not only are the actions that allow discerning if an account is a bot but the sequence of the actions as well. Cresci et al. assign a letter to each tweet of an account, depending on the types of tweets shared. Their encoding was done in the following way: **A** for a simple tweet, **T** for a reply, and **C** for a retweet. Next, a DNA fingerprint of an account is obtained by creating a string with all the assigned letters of each tweet, appearing in chronological order. To test the approach, they examined the similarities between the DNA of the accounts. Cresci et al. considered DNA similarity as a proxy for automation; thus, large groups of accounts with an exceptionally high level of similarity were most likely to have an anomalous behavior. Precisely, the similarity among the accounts was quantified by looking at the Longest Common Substring (LCS) among digital DNA sequences. With this method, Cresci et al. obtained an F1 of 0.97.

Loyola et al. [82] emphasized the need for understandable classification models and proposed to approach Twitter bot detection using contrast-pattern based classifiers. Because contrast-pattern models have high explanatory power, they might help experts in different scenarios [44]. For example, these models can support decision making, or be used to forewarn an account holder about suspicious activity [83, 82]. The second contribution of Loyola et al. is the introduction of a new feature model, which includes features out of Twitter account usage and tweet content sentiment analysis. The proposed feature representation provided the lowest standard deviation, among all the tested classifiers. Regarding classification algorithms, the top-ranked (using AUC score as criteria) (LCmine+Hell)+Filt+PBC4cip, and (LCmine+Hell)+PBC4cip obtained an average AUC of 0.9976.

Currently, state of the art for Twitter bot detection is the Botometer tool proposed by Yang et al. [129]. Botometer uses a Random Forest classifier [26] over 1,200 features from all the categories. Their results are reported using bots of different types, obtaining a score of up to 1.0 AUC, when classifying political bots. Overall, this method achieves a cross-validation performance of 0.97 AUC.

Most of the approaches for Twitter Bot detection rely on supervised classifiers. However, unsupervised classification methods have also been used to detect groups of related bots, and have even been tested on supervised datasets to obtain measurements of performance.

## 2.2.2 Unsupervised Bot Detection Approaches

Unsupervised bot Detection approaches represent an alternative that aims to uncover different types of bots. Regularly, unsupervised methods make use of some clustering algorithm to detect correlated bot accounts. When these accounts act in a highly synchronized manner and share their objective, they are referred to as *botnets*.

Ahmed and Abulaish [4] proposed to find groups of suspected spam campaign accounts using the Markov clustering algorithm [119]. To find groups of spammers, first, they created an undirected, fully connected graph where each node is a suspected spam account. This graph contains the weight of the links, which is the similarity between accounts using tweet usage features; then, the clustering algorithm is applied. On the downside, while they claim that campaigns were successfully detected, they do not give any performance measure.

Miller et al. [88] used a clustering approach to find groups based on features of either tweet account or account usage. Since account usage varies over time, Miller et al. use Den-Stream [123] and StreamKMmeansCC [2], which are techniques adapted for data streams. The authors claim to have obtained an F1 of 0.88 using this approach.

Chavoshi et al. [31] analyzed the behavior of Twitter users and observed that it is highly unlikely that several users retweet a message in less than 20 seconds after it is posted, or that they post a message with a related trending topic in the same second. To support this claim, they use social network features together with a lag-sensitive hashing technique and a warping-invariant correlation measure to organize the accounts in clusters of abnormally correlated accounts. The results showed that, when the accounts are used to post for the same trending topic in a short time frame, they find groups of bots for Twitter with a precision of 0.94.

Author	Performance	Method	Relevant Details	Approach
Lee et al. [76]	F1=0.88	Decorate classifier	Use content, tweet account, and account usage features.	Supervised
Wang et al. [124]	F1=0.91	Naive Bayes	Similar approach to Lee et al., but using only last 20 tweets of each account.	Supervised
Ahmed et al. [4]	Recall=0.987	Jrip	Features explored are interaction-driven, tweet-driven and URL-driven.	Supervised
Chu et al. [34]	Recall=0.96	Random Forest	Includes usage features, such as how the client logs in and regularity of tweet posting	Supervised
Yang et al. [128]	F1=0.9	Random Forest	Uses features of type tweet usage and social network	Supervised
Wang et al. [125]	F1=0.94	Random Forest	Content and sentiment of the tweet; information and account use.	Supervised
Giliani et al. [56]	F1=1.0	Random Forest	Separates the accounts according to the number of followers. F1 = 1 for accounts with number of followers $\geq 10$ Millions	Supervised
Giliani et al. [56]	F1=0.84	Random Forest	For accounts with number of followers $< 1,000$ .	Supervised
Cresci et al. [37]	F1=0.97	DNA-like analysis (Longest Common Substring)	Implements a DNA-like analysis of the tweet usage account	Supervised
Loyola et al. [82]	AUC=0.99	Contrast-patterns based classifiers	Uses contrast-patterns based classifiers along with features out of Twitter account usage and tweet content sentiment analysis.	Supervised
Yang et al. [129]	AUC=1	Botometer (Random Forest-based)	Uses over 1,200 features from all the categories. Trains with different types of bots. Best AUC performance was obtained when classifying political bots.	Supervised
Miller et al. [88]	F1=0.88	DBScan y K-means++	Uses a clustering approach to find groups based on features of either tweet account or account usage.	Unsupervised
Chavoshi et al. [31]	Precision=0.94	Lag-sensitive hashing technique and a warping-invariant correlation measure	This method considers cross-correlating user activities and requires no labeled data.	Unsupervised

Table 2.1: Previous works comparison.



## 2.3 Implications of the Evolution in Bots' Behavior

As the evolution of bot behavior continues, current detection models become obsolete, as they are not efficient in uncovering new types of bots. Moreover, since bots continue to evolve and diversify, obtaining examples of them is not a trivial task. Therefore, new detection methods are required, methods that do not require previous knowledge of all the different types of bots that populate the network.

### 2.3.1 Paradigm-shift in Bot Design

Investigators of social networks have documented the evolution of bot behavior [48, 132]. A particular type of bot that has changed its behavior over time is the spambot. Spambots have modified their modus operandi to evade basic detection techniques [38], such as the ones based on posting patterns [110], textual content of posted messages [76] and relationship between accounts [55]. Additionally, the number of OSNs users fooled by bots increases along with bot's behavior complexity [37].

Machine-learning applications have not only been used for developing new detection tools, but also for automatically generating bot content [51] and emulating human posting and consumption behavior patterns [64]. Related groups of bots, botnets, may aim to increase the popularity of specific accounts, topics, or opinions by posting similar content. Since these accounts do not publish spam individually, but altogether, their behavior is generally harder to uncover [129]. Researchers have developed unsupervised techniques to detect botnets when they act in a strongly coordinated manner [31, 32]. Unfortunately, these methods have the limitation of relying on checking a single feature, which may allow for different types of botnets to pass undetected.

### 2.3.2 Relationship Between Bot Behavior and Classifiers Performance

Researchers have shown that emerging waves of spambots have characteristics that yield ineffective some conventional detection methods, such as the ones evaluating single accounts based on a set of established features tested over known datasets [38]. Moreover, the paradigm-shift in bot design, and therefore bot behavior has caused unsatisfactory performances in several state-of-the-art bot detection methods [38]. Additionally, experiments have shown that testing datasets containing new types of bots tend to lead to lower performances when compared against models that had the opportunity to learn from those instances [129].

A proactive approach for bot detection, which makes use of synthetically evolved versions state-of-the-art social bots, has also been proposed [39]. Such research provides evidence of how so-called evolved spambots evade current detection techniques. Specifically, Cresci et al. [39] developed experiments using three state-of-the-art bot detection techniques. By evaluating the accounts' behavior and content posted, their results showed that as much as 79% of evolved spambots are capable of evading detection.

Due to the unsatisfactory detection results of state-of-the-art techniques, there is a need for novel approaches that are capable of ensuring the timely detection of new types of bots [38]. Moreover, to provide detection mechanisms robust enough to keep up with all kinds

of coordinated bots, new algorithms and approaches are needed, ones that exploit different dimensions of similarity between genuine and bot accounts [129].

## 2.4 Summary

Twitter bot detection is a problem that has gained popularity in recent years. Scientists have developed several supervised and unsupervised detection methods with the objective of uncovering malicious bots and ban them from the network before they continue causing harm. These methods have shown to be effective when detecting known bot types, but they have some limitations. The most prominent limitation of current bot detection mechanisms is the need to possessing bot type examples beforehand to be able to detect them. This requirement threatens the effectiveness of current detection systems due to the evidence of a paradigm shift in bot behavior and design.

## Chapter 3

# One-class Classification for Twitter Bot Detection

The main focus of this research is to overcome one of the main limitations in the state-of-the-art methods for bot detection, which is the need of possessing specific bot-types examples to construct reliable classifiers. In particular, this thesis aims to show that one-class classifiers complement existing approaches, as they only require genuine (human) users' behavior to learn representations that help them to discern between human and bot accounts. Therefore, the hypothesis is that, given an appropriate feature vector, one-class classifiers may be able to detect instances of new, previously unseen, classes of bots. Thus, overcoming the main limitation of state-of-the-art methods. Another contribution of the present work is the proposed feature vector, which was designed by studying the literature and carefully selecting the features that have demonstrated to provide the most discriminatory power when used to discern between bots and human users.

The following sections describe one-class classifiers, the advantages, and limitations of these classifiers in the bot detection context, and the experimental methodology.

### 3.1 Fundamental Approaches for Anomaly Detection

Hodge [63] summarizes the outlier detection problem into three basic approaches that are analogous to the basic types of machine learning. The following subsections address each one of these three approaches.

#### 3.1.1 Unsupervised Clustering

In an unsupervised clustering approach, outliers are determined without prior knowledge of the data. Also, the data is assumed to be static. Once data is given as input, the most remote objects, the ones separated from the cluster of 'normal' data, are considered to be outliers. This approach can be adapted to be suitable for both classification and outlier detection by subdividing the main cluster into smaller ones. Even though an unsupervised anomaly detection requires all the data to be static and available before processing, in a sufficiently large

database, one with good spatial distribution coverage, this method becomes suitable for comparing new items with the existing data [63].

Rousseeuw and Leroy [103] describe two sub-techniques commonly applied within this approach: diagnosis and accommodation. In a diagnosis approach or diagnostics, as referred by Rousseeuw and Leroy, the system removes potential outlying points before further processing the data. Frequently, this is done by iteratively pruning all the outliers and then performing clustering to the remaining data. The second alternative, accommodation, makes use of classification and results in a robust method where outliers can be incorporated into the generated model.

An accommodation methodology can take outliers in the data and still represent normal behavior by inducing a margin of normality around the majority of the data; this overcomes the limitations of non-robust classifiers, which can produce skewed representations if outliers are considered together with the rest of the data. As for the computational cost, non-robust methods are cheaper, but they are susceptible to distortion in cases where there are a large number of outliers.

### 3.1.2 Supervised Classification

The supervised classification approach (for anomaly detection) models both normality and abnormality; therefore, it requires the data to be pre-labeled as normal or abnormal. Ideally, a trained classifier would learn the area where the normal class is located, leaving the outlier examples outside of it. Moreover, a supervised approach not only can be used to perform a normal/abnormal classification, but it is also capable of providing a classifier with  $k$ -classes of normality [63]. The different normal classes are obtained by sub-dividing the classification area into  $k$  distinct classes according to the requirements of the system. Figure 3.1 shows the classification area for the normal class learned by a one-class classifier with three classes of normality.

This approach is suitable for on-line classification, where the classifier first is trained to learn a classification model and then classifies new instances accordingly. As with other approaches, a given classifier labels as *normal* the instances located inside a region of normality; instances located in any other regions are labeled as outliers. Figure 3.2 illustrates this example. To allow a good classifier generalization, the data used should cover the entire distribution of normal instances. This generalization capacity might allow the classifier to label objects located inside the *known* distribution correctly. On the contrary, the generalization requirements represent a limitation because objects derived from an unknown, previously unseen, region of the distribution, might not be classified correctly.

### 3.1.3 Semi-supervised Classification

Semi-supervised detection approaches can be found in the literature under the name of novelty recognition or novelty detection. Though it can also be used to model abnormality, in the majority of the cases, semi-supervised detection approaches focus on only modeling normality [46, 66].

In a semi-supervised approach, the algorithm is only shown objects belonging to the normal class, but through this process, it learns to recognize abnormality. As with the previous

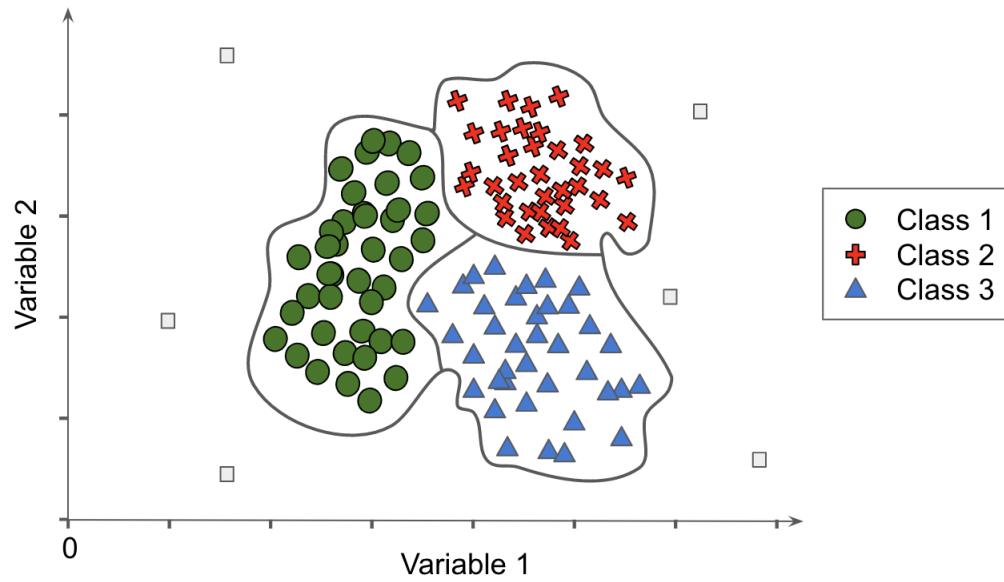


Figure 3.1: Classification area, with three classes of normality, learned by a one-class classifier.

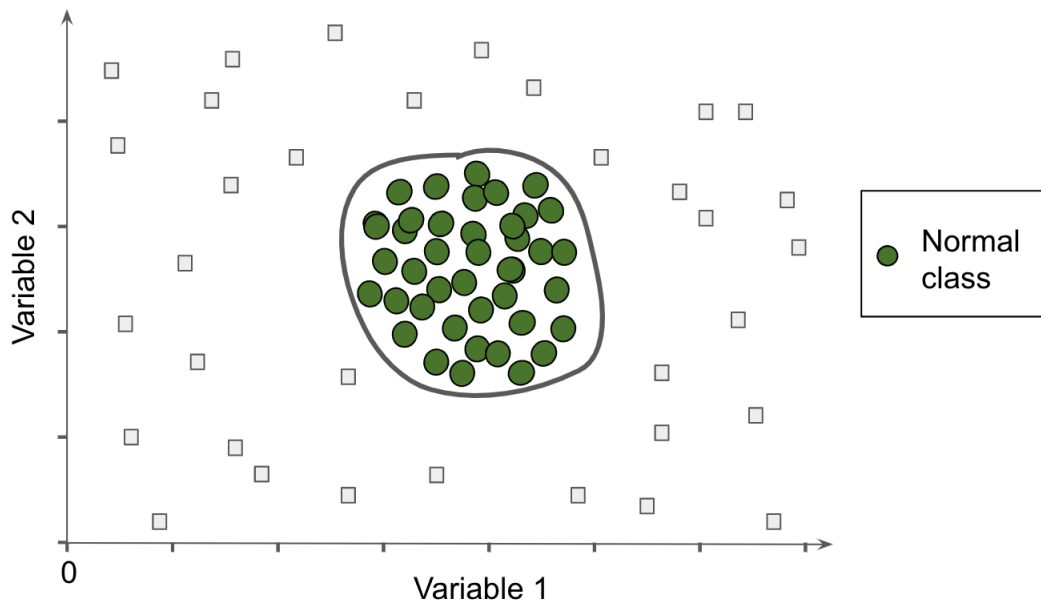


Figure 3.2: Region of normality learned by a one-class classifier.

approach, the classifier needs labeled data. This approach has the advantage of being suitable for both static and dynamic data since it only learns a single class, which provides the model

of normality. Additionally, the model can learn incrementally as new examples arrive, thus updating the model with every new example [63].

Common to the other two main approaches, a semi-supervised detection system classifies a new object as normal if it is located within the boundary of normality; otherwise, it is categorized as an outlier. Depending on the underlying detection algorithm, the normality boundary can be hard or soft. In a hard boundary, the objects lie either inside or outside the boundary, resulting in a hard, binary classification. On the contrary, when the boundary is soft, objects are assigned a degree of class membership or *outlierness*. Figure 3.3 shows different degrees of *outlierness* learned by a one-class classifier.

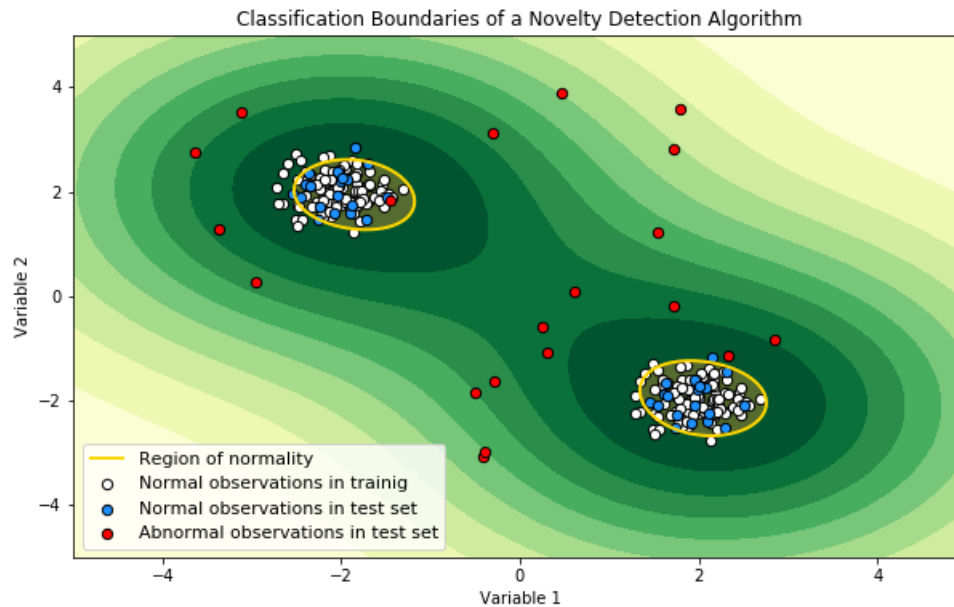


Figure 3.3: Classification regions learned by a one-class classifier. Colors represent different degrees of membership to the normal class, being the lightest an indicator of a 0% degree of membership to the normal class.

This approach requires big amounts of normal data to compensate for the lack of abnormal instances. Feeding the model with a big, good-quality dataset, often results in a model with good generalization capabilities [63]. This technique is the most suitable for domains where abnormal data is expensive or difficult to obtain. Such is the case in fraud detection [42], fault detection in an engine [14], personal-risk detection [99], and malware detection [67].

Additionally, semi-supervised approaches solve some of the problems that are present in supervised classification approaches. One of the problems in the second approach is that it does not always handle well outliers from unexpected regions. Hodge [63] demonstrates this situation in the fraud detection context, where a new fraud method never previously encountered may not be handled correctly by the classifier that uses the second approach unless its generalization capacity is outstanding. In contrast, a system based on a semi-supervised

approach can succeed in detecting the fraud as long as it lies outside the boundary of normality. To keep up with changes in the normality, methods using this third approach may be shifted by re-learning the data model. As an upgrade alternative, if the underlying modeling technique allows it, the model itself can be shifted; such is the case of evolutionary neural networks.

As we shall see next, one-class classification is closely related to a semi-supervised anomaly detection approach. Therefore, most of the applications of one-class classifiers involve the detection of anomalies in the behavior of the analyzed objects. In this thesis, one-class classifiers are used to overcome the limitations in bot detection, which are due to the lack of bot type examples.

## 3.2 Classification in Machine Learning

Classification is one of the most popular Machine Learning tasks. Algorithms used to perform classification take as input a set of instances and learn a set of rules from them, using a given mathematical model. These rules are then used to classify new instances. Moreover, the specific properties of each instance, or object, can be represented through a collection of numerical or categorical variables, also called features [92].

### 3.2.1 Training and Testing a Supervised Classification Model

In supervised classification, the target, also known as a class, of every instance is known. Therefore, for every instance, there is a vector of features that describes it and its true class label. It is a common practice that, to create a classification model, the researcher splits the available data into two different sets: training and testing [73].

The instances contained in the training set are given as input to the classifier to fit the model. These instances are used to adjust the parameters of a given algorithm by using a supervised learning method. Thus, allowing the algorithm to learn a relationship between the features of the objects and the labels. Some famous classification algorithm examples are Random Forest, Support Vector Machines, and Naive Bayes.

The objective of the testing set is to provide an unbiased evaluation of the model fitted on the training dataset. For this test to be reliable, the instances of the testing and training sets must be independent [97]. The process to assess the generalization capabilities of the fitted model is straight forward. First, the trained model is used to classify the instances in the testing set, and then this prediction is compared to the actual label. The generalization capabilities can be measured using different metrics, depending on what aspect of the classifier wants to be evaluated. Later on this chapter, we address the most often used performance measures.

### 3.2.2 One-class, Binary, and Multi-class Classification

Models using binary and multi-class are designed to receive data that contains labels for each instance and learn rules to distinguish each one of them, based on pattern present in their feature vector. Moreover, in the case of multi-class classification, all the possible distinct labels are present in the dataset; this means that when a new instances need to be classified, the

algorithm assigns it the label of its most similar class. This similarity is measured according to the specific algorithm used to construct the model.

One-class classifiers are different from traditional methods just in the training phase. One-class methods work by characterizing a training set of instances that belong to just one class. Then, at the classification phase, one-class methods aim to detect if a new, previously unseen object, resembles such set [113]. For one-class classifiers to work correctly, the target class is required to be appropriately defined and sampled. As an advantage, non-target objects are not required to constitute a meaningful class; there is no need for them to be sampled thoroughly and extensively [92].

In the literature, authors refer to one-class classification in different ways. These terms have their origin in the different applications of one-class classification [113]. The term “one-class classification” was first introduced in 1993, in an article by Moya et al. [89]. Later, other authors refer to such applications of this technique as novelty detection [22], outlier detection [98] or concept learning [114]. Through this document, those terms are used interchangeably.

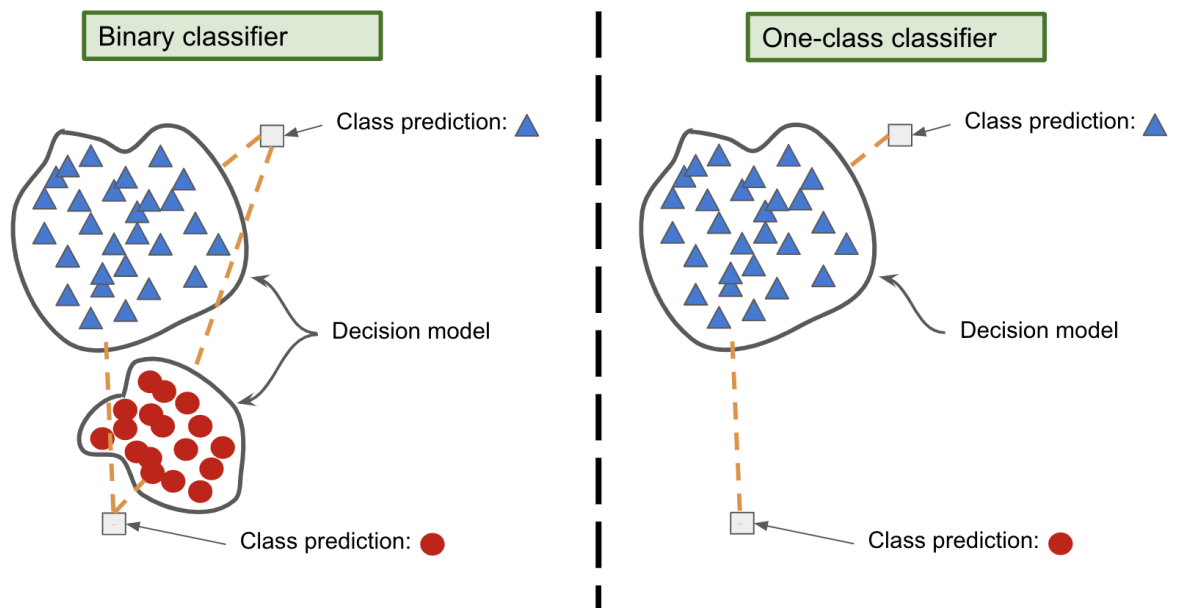


Figure 3.4: Comparison of classification models learned by a one-class and a binary classifier. Figure inspired by Rodríguez et al. [101].

### Comparison of Classification Methods

Figure 3.4 compares the classification area learned by a one-class classifier and a binary classifier. The thick blue line represents the region of the cluster that represents the class it encloses, while the squares outside the clusters represent an object from the testing set. In the one-class example, the classifier calculates the distance from the object to the cluster and then uses a threshold to classify the object as normal or abnormal. In contrast, a binary classifier calculates the distance of the object to both clusters and label the queried object with the class of



the nearest cluster.

### 3.2.3 Measures to Compare Classification Performance

The performance measures that can be used to evaluate a classifier vary depending on the classification task, which can be binary, multi-class, multi-labeled, or hierarchical [107]. According to Sokolova Lapalme [107], the most used measures, based on the confusion matrix, for binary classification are Accuracy, Precision, Sensitivity, F1, Specificity, and AUC. Recall these measures are described in detail in Section 2.1.

Selecting the appropriate measure is crucial in any experiment, since comparing classifiers' performance with an inadequate measure can lead to misleading interpretations. For example, imagine a binary classifier always predicts a given instance is from class A. If we have 5 instances of class B and 95 instances of class A, this classifier yields 95% accuracy. This score might sound outstanding. Still, in reality, this classifier is useless due to its inability to detect instances of class B. This example represents an imbalanced class problem, a scenario in which the accuracy metric can be misleading. From all the mentioned measures, F1 and AUC have the advantage of being balanced performance measures, which are robust against imbalanced class problems [68, 78].

## 3.3 Advantages and Limitations of Existing Approaches for Bot Detection in Twitter

### 3.3.1 The Supervised Approach for Bot Detection

Most of the works regarding Twitter bot detection approach the problem as one of supervised classification. Since the goal of this task is to differentiate between normal and bot accounts, it is very intuitive to tackle the problem with a binary classification approach. The main advantages and limitations of this approach are address in the following paragraphs.

#### Advantages of a Supervised Approach for Bot Detection

Through the years, supervised bot detection approaches have contributed to determine which individual features, or combinations of them, allow for a better characterization of the twitter user behavior. Moreover, when the bot type is known, supervised approaches, such as [129, 82], have led to the highest performance scores reported so far.

#### Limitations of a Supervised Approach for Bot Detection

Adopting a supervised classification strategy for bot detection has some limitations. For example, a bot might pass undetected if its class is different from those used in the training phase of the classifier. The effectiveness of the detection system depends on how different the bot behavior is, compared to previously seen bots. In real-life scenarios, supervised detection systems are vulnerable because the behaviors of bots are not homogeneous. Researchers have

shown that there are types of bots aiming to emulate social behaviors to avoid detection and that there are new types of bots entering Twitter regularly [38, 39].

This vulnerability represents a threat to detection mechanisms based solely on a supervised classification approach. In both, a multi-class and binary class classification methods, new instances are categorized as one of the pre-defined classes, even if it is the case that this new example belongs to another, unknown category. This assumption made by the classifier entails a risk in cases where complete information about specific instances' types is not available. Even more, this lack of information could be detrimental to the performance of the classifiers.

### 3.3.2 The Semi-supervised Approach for Bot Detection

This thesis work proposes to approach bot detection using a semi-supervised approach, namely one-class classifiers. One-class classifiers only require examples of one class, which in this case are genuine users examples to learn a classification model capable of identifying if a new object belongs to that class or not. Since bot type examples are scarce, and there is no ground truth for the behavior of a bot, detection mechanisms can benefit from this type of classifier. The main advantages and limitations of this approach are address in the following paragraphs.

#### Advantages of a Semi-supervised Approach for Bot Detection

As a way to deal with the vulnerabilities of supervised methods, we propose to approach the Twitter bot detection task using one-class classification. Specifically, using a one-class classifier, with a semi-supervised internal mechanism, to model the behavior of genuine (human) Twitter users, would provide systems the capacity to detect bots regardless of the bot type. Likewise, it would be able to detect entirely new types of bots. In other words, mechanisms that make use of the proposed approach can detect bots based on how the behavior they expressed deviates from normal (genuine) user behavior. Thus, one-class classifiers are capable of detecting novel bots and overcome one of the main limitations of state-of-the-art supervised methods.

#### Limitations of a Semi-supervised Approach for Bot Detection

Despite the benefits it brings, a semi-supervised approach for bot detection is not flawless. The performance of one-class detection mechanisms is closely related to the characterization of the behavior of Twitter accounts. If the feature space does not provide enough information to differentiate the behavior of bot and human accounts, one-class methods may perform poorly. Also, as the experimental results in Section 5 show that, if the bot types are already known, binary classifiers tend to perform better.

## 3.4 Methodology

To validate the hypothesis that one-class classifiers may be able to detect instances of new classes of bots, we designed three experiments sets, which answer the following research questions:

- Does the performance of multi-class methods decrease when it comes to identifying bots of an unknown class?
- Do one-class classifiers perform better than multi-class methods when it comes to identifying bots of an unknown class?
- When identifying bots of an unknown class, is there a statistically significant difference between the performances obtained by the two methods?

### 3.4.1 Experiments

Three experiment sets were designed to test the advantages and limitations of a semi-supervised anomaly detection approach for bot detection. These experiments aim to compare fairly and directly the performances of multi-class and one-class classifiers. The obtained results show the generalization capacity of multi-class classifiers that have been trained with three classes of bots and tested with bots of an unknown class.

The descriptions presented in this section are done at a high-level. The specifics of every experiment are detailed in Chapter 4. Each experiment set deals with a different *type* of classifier (binary, multi-class, or one-class). Once all the experiments are carried out, the performances are compared to answer the research questions.

#### Experiment Set A: Binary Classifiers

This set of experiments involves binary classifiers. In this set of experiments, the binary classifiers are trained with a set containing instances of genuine users and one type of bots and then tested using a set containing different types of bots. This procedure is repeated until all the available types of bots are used in the training phase.

The obtained results give information about the generalization capabilities of binary classifiers. We hypothesize that, if the behavior shown by the different bot types really differ from one another, the classifiers' performance will be outstanding when classifying the same type of bot. Still, it will decrease when classifying other bot types.

#### Experiment Set B: Multi-class Classifiers

This set of the experiment involves a multi-class version of the binary classifiers and simulates more closely how classifiers can be used to detect bots. This simulation is done by imitating the strategy of most publicly available bot detection tools. The strategy consists in using a diverse set of bot examples, one that encompasses multiple behaviors, for the classifier to learn better how to discern between legitimate and bot accounts.

In experiment set B, the multi-class classifiers are trained with a set containing instances of genuine users and three (out of four) type of bots and then tested using a set containing the remaining type of bots. This procedure is repeated until the four types of bots are tested.

### Experiment Set C: One-class Classifiers

This set of experiments involves one-class classifiers. Since one-class classifiers have the particularity of only requiring only legitimate user accounts in their training phase, the training sets created for the previous experiment set can be used without further modifications.

In this set of experiments, the selected one-class classifiers are trained with legitimate accounts and tested with a dataset that contains legitimate and bot account examples of only one type. For each classifier, this procedure is repeated until all types of bots are used in the testing phase. Due to the nature of one-class classifiers, we hypothesize that their performance will be stable, this means there will be a low standard deviation among the performances obtained when classifying different types of bots.

### 3.4.2 Datasets Used in the Experiments

Understanding the rationale behind the design of Twitter bots is essential, as it can help to develop better detection mechanisms. Recent studies, such as the one performed by Cresci et al. [38], show that there exists a paradigm shift in the design of Twitter bots. In their study, the bots found were grouped into nine different types, according to their behavior or purpose. These groups are present in a single database, which constitutes real-world data. Table 3.1 describes Cresci’s Twitter accounts groups.

For the experiments carried out in the present work, five of Cresci’s datasets are used. These datasets are genuine accounts, traditional spambots #1, and all social spambots. Cresci et al. collected examples of the three social spambot groups by using a Twitter crawler to retrieve data of accounts that were suspected of belonging to one of those groups. Then, a manual verification phase was carried out to certify the automated nature of all the collected accounts. This process resulted in 991, 3,457, and 646 verified spambots accounts, for the groups of social spambots #1, #2 and #3, respectively.

The genuine-accounts dataset contains 3,474 human-operated accounts. Those accounts were randomly sampled and then contacted to verify if they were genuine users. The traditional spambots #1 dataset contains a set of 1,000 malware spammers accounts. Those spammer accounts were used in the work of Yang et al. [128] as the training set of a classifier that aims to detect evolving spambots on Twitter.

Dataset	Description	Accounts	Tweets	Year
Social spambots #1	Retweeters of an Italian political candidate	991	1,610,176	2012
Social spambots #2	Spammers of paid apps for mobile devices	3,457	428,542	2014
Social spambots #3	Spammers of products on sale at Amazon.com	464	1,418,626	2011
Genuine accounts	Verified accounts that are human-operated	3,474	8,377,522	2011
Traditional spambots #1	Training set of malware spammers used by Yang et al. in [128]	1,000	1,418,626	2009
Traditional spambots #2	Spammers of scam URLs	100	74,957	2014
Traditional spambots #3	Group automated accounts spamming job offers #1	433	5,794,931	2013
Traditional spambots #4	Group of automated accounts spamming job offers #2	1,128	133,311	2009
Fake followers	Automated accounts that inflate the number of followers of another account	3,351	196,027	2012

Table 3.1: Datasets released by Cresci [38].

The rest of the datasets used in Cresci’s research [38], namely traditional spambots #2, traditional spambots #3, traditional spambots #4, and fake followers, were not used in the present study. These datasets were excluded because they did not provide the information

required to obtain the feature vector proposed in Section 3.4.3. The five datasets that were indeed used were obtained from the public Bot Repository<sup>1</sup>, a web portal self-defined as a *centralized place to share annotated datasets of Twitter social bots*.

### 3.4.3 Features for Characterizing Twitter User’s Behavior

Each of the datasets described in Table 3.1 comes in a folder with two comma-separated values files: *tweets* and *users*. The *tweets* file contains information regarding the tweets of collected users; it provides a vector with 25 features for each tweet. On the other hand, the *users* file contains a vector of 40 features for each account. The features provided in the *users* files contain three different data types: numeric, nominal, and text. Tables 3.2 and 3.3 describe all the features contained in the selected datasets.

As mentioned before, the classification performance of any algorithm is often related to several factors, including the quantity and quality of the features used to train it. In the bot detection domain, it is well known that some of the state-of-the-art methods make use of more than a thousand features [129]. However, the feature extraction process is often very time-consuming, and not all the features contribute to the same amount to the improvement of the detection performance.

#### Proposed feature vector

Taking into account the discoveries made in the previous works, a feature vector, per account, is proposed. According to the information presented in the bot detection literature, several features were carefully selected to make up the vector used to feed the classifiers. Moreover, the chosen features have shown to provide good representations when used in different supervised classification methods. During the selection process, it was taken into account that the features of the vector used for training the algorithms must be discriminative to discern bot accounts [4].

The proposed feature vector includes 13 features extracted for each account. The features that conform to the input vector are described in Table 3.4; their category is also provided. The data was extracted from the original *users* files used in Cresci et al. [38] to obtain the information for each feature. Furthermore, statistical information, such as the ratio between unique URL elements and total tweet count, was obtained for each user from the *tweets* files. This information is useful for determining the posting behavior of the users and the content of each post.

### 3.4.4 Selected Classifiers

A vital component of the hypothesis testing is the selection of binary and one-class classifiers. Moreover, the performance of a classifier is dependent on the underlying assumptions, features’ type used, and also on the statistical distribution of each feature [50]. Due to the variability of the classifier’s operating conditions, the experiments are done using a set of diverse classifiers. Moreover, each classifier is representative of a supervised classifier family [11]. This diversity prevents the results from being bias towards certain types of classifiers. In

<sup>1</sup><https://botometer.iuni.iu.edu/bot-repository/>

Name	Type	Description
Id	Numeric	Unique identifier of the tweet
Text	Text	Text contained in the tweet
Source	Text	Source of the tweet
User Id	Numeric	Unique identifier of the user who tweeted
Truncated	Nominal	Indicates if tweet is truncated
In Reply To Status Id	Numeric	ID of the status which was answered
In Reply To User Id	Numeric	Id of the user who was answered
In Reply To Screen Name	Text	Screen name of the user who was answered
Retweeted Status Id	Text	Status id of the retweeted post
Geo	Nominal	Geolocation of the user when posting the tweet
Place	Nominal	Place the user was in when the tweet was posted
Contributors	Text	Tweet's contributors
Retweet Count	Numeric	Number of retweets of the tweet
Reply Count	Numeric	Number of replies to the tweet
Favorite Count	Numeric	Number of favorites of the tweet
Favorited	Nominal	Indicates if the tweet has been favorited by the account used to access it
Retweeted	Nominal	Indicates if the tweet has been retweeted by the account used to access it
Possibly Sensitive	Nominal	Flag indicating if the content of the tweet is possibly sensitive
Num Hashtags	Numeric	Number of hashtags contained in the tweets
Num Urls	Numeric	Number of URLs contained in the tweets
Num Mentions	Numeric	Number of mentions contained in the tweets
Created At	Datetime	Datetime on which the tweet was created
Timestamp	Datetime	Datetime on which the tweet was published
Crawled At	Datetime	Datetime on which the tweet was crawled
Updated	Datetime	Datetime on which the tweet was updated

Table 3.2: Features provided for each tweet in the selected Cresci's datasets.

this section, we describe the five one-class classifiers used in the experiments, since they may not be well-known. On the contrary, binary classifiers are not described, as they are widely used in diverse contexts.

### Multi-class and Binary Classifiers

Ten widely-known binary classifiers were selected for the bot detection experiments. These classifiers are representative of the methods used in the bot detection literature. Using this variety, we test different situations so as not to be biased towards one or another composition of the data. The classification algorithms used are Bayes Network [17], J48 [96, 71], Random Forest [62], Adaboost [52, 133], Bagging [25], K-Nearest Neighbors (KNN) [10], Logistic Regression [130], Multilayer Perceptron (MLP) [61, 104], Naïve Bayes [131], PBC4cip [83], and Support Vector Machines [28].

The most direct manner of comparing the performance of the proposed one-class approach to the one obtained by state-of-the-art methods would have been possible if all the implementations were publicly available. However, not all the authors provide a public implementation of their bot detection method; or the implementations are already trained, which does not allow to compare the effect of unknown bot types for that method.

The classification methods reported in the literature, but not included in the experiment

Name	Type	Description
Id	Numeric	Unique identifier of the account
Name	Text	Name of the account
Screen Name	Text	Screen name of the account
Statuses Count	Numeric	Number of statuses posted
Followers Count	Numeric	Number of followers
Friends Count	Numeric	Number of friends
Favourites Count	Numeric	Number of favorited tweets
Listed Count	Numeric	Number of tweets listed
Url	Text	Url of the account's profile
Lang	Nominal	Language of the account
Time Zone	Nominal	Time zone the account uses
Location	Nominal	Location of the account
Default Profile	Text	Default profile of the account
Default Profile Image	Text	Default profile image of the account
Geo Enabled	Nominal	Indicates if geo is enabled
Profile Image Url	Text	URL of the account's profile image
Profile Banner Url	Text	URL of the account's profile banner
Profile Use Background Image	Text	URL of the account's profile background image
Profile Background Image Url Https	Text	Profile background image url https
Profile Text Color	Text	Profile text color
Profile Image Url Https	Text	Profile image url https
Profile Sidebar Border Color	Text	Profile sidebar border color
Profile Background Tile	Text	Profile background tile
Profile Sidebar Fill Color	Text	Profile sidebar fill color
Profile Background Image Url	Text	Profile background image url
Profile Background Color	Text	Profile background color
Profile Link Color	Text	Profile link color
Utc Offset	Numeric	Universal Time Coordinated (UTC) offset
Is Translator	Nominal	Indicates if the account is translator
Follow Request Sent	Nominal	Indicates if friend request has been sent to the account
Protected	Nominal	Indicates if the account is protected
Verified	Nominal	Indicates if the account is verified
Notifications	Nominal	Indicates if notifications for this account are active
Description	Text	Description of the account
Contributors Enabled	Nominal	Indicates if contributors are enabled
Following	Nominal	Indicates if the account is being followed
Created At	Datetime	Datetime of the account's creation
Timestamp	Datetime	Datetime on which it was published
Crawled At	Datetime	Datetime on which it was crawled
Updated	Datetime	Datetime on which it was updated

Table 3.3: Features provided for each user account in the selected Cresci's datasets.

Feature	Description	Category
Retweets	Ratio between retweet count and tweet count.	Account Usage
Replies	Ratio between reply count.	Account Usage
Favoritec	Ratio between favorited tweet and tweet count.	Account Usage
Hashtag	Ratio between hashtag count and tweet count.	Account Usage
Url	Ratio between url count and tweet count.	Account Usage
Mentions	Ratio between mention count and tweet count.	Account Usage
Intertime	Average seconds between postings.	Account Usage
Favorites	Number of tweets favorited in this account.	Account Usage
UniqueHashtags	Ratio between unique hashtag count and tweet count.	Account Usage
UniqueMentions	Ratio between unique mention count and tweet count.	Account Usage
UniqueUrl	Ratio between unique urls count and tweet count.	Account Usage
Ffratio	Friends-to-followers ratio.	Account Information
Listed	Number of listed tweets in the account.	Account Information

Table 3.4: Feature vector extracted for each Twitter user.

section, are described in the following paragraphs, along with the reasons why they are excluded.

First, the Decorate classifier proposed by Lee et al. [76] was not included because, additionally to not being widely used, other popular methods have achieved higher performances. Next, Cresci’s DNA-like [37] analysis method for bot detection is not publicly available; therefore, it is not used. Moreover, further studies by the same author [39] demonstrate that relatively simple bot design strategies can be used to fool such a detection method easily. Because of its nature, Loyola-et al. [82] method, based in contrast patterns, is not directly comparable with the one-class method because it performs classification per tweet, instead of per account. Consequently, this contrast pattern method falls outside the scope of this thesis. Lastly, the state-of-the-art commercial tool, Botometer [129], is not used in the experiments. This method was excluded for several reasons. First, the Botometer detection mechanism requires 1,200 features that are automatically extracted from the profile of a given active Twitter user. Requiring so many features represents a limitation because not all the accounts included in the dataset are still active, as they have been blocked or deleted due to the infraction of Twitter’s terms and conditions. Also, the publicly available version of Botometer is already trained, hence comparing the performance it yields to the one obtained by the one-class classifiers does not result in a fair comparison. Nonetheless, in a paper that discusses the previous version of Botometer, Davis et al. [40] mentioned that this tool is based on the Random Forest algorithm, which is included among the selected algorithms.

Nonetheless, the selected classifiers are representative of the ones used in state-of-the-art bot detection mechanisms in the literature. As a reference, Table 2.1 contains the state-of-the-art Twitter bot detection works, along with the classifier used in each one.

### One-class Classifiers

When selecting the one-class classifiers to be included, the main objective was to select the ones belonging to the state-of-the-art in anomaly detection. This led to choose three



ensemble-based [93, 94, 102] classifiers: Bagging-RandomMiner (BTPM) [29], Bagging-TPMiner (BRM) [87], One-Class K-means with Randomly-projected features Algorithm-(OCKRA) [99]. The first two classifiers belong to the state-of-the-art in the masquerade detection problem domain. The third classifier, OCKRA, is one of the best performing algorithms in the personal risk detection task, outperforming one-class SVM, and different versions of the Parzen window classifier. Besides, One-Class Support Vector Machines(OC-SVMs) and One-Class Naïve Bayes (OC-NB) were used for comparison and completeness. Since, in contrast to the binary and multi-class classifiers, one-class methods are less known, a short description for each one of them is provided.

**BTPM:** *Bagging-TPMiner* [87] is a classifier ensemble designed for masquerader detection based on *typical* objects. The base classifier of this ensemble, TPMiner, has the key feature of attempting to capture examples of normal behavior that are located in regions that are not dense. Generally, this kind of behavior yields false alarms due to being overwhelmed by samples in denser regions. Bagging-TPMiner differs from existing clustering techniques in the sense that it can capture the, often hidden, structure of the ordinary user behavior. This capture is done by giving similar weight, or attention, to dense and sparse regions. At the training phase, the classifier aims to find similarity patterns among the instances. The inter-object distances are calculated and, for each instance in the training set, an object closer than a threshold is chosen as a *typical* object that represents it. This method performs bagging with 100 TPMiners to add diversity and thus improve performance. At the testing phase, BTPM classifies a new object as typical or atypical by outputting 0 or 1. The classification process is done computing the distance to the nearest *typical* object, transforming that distance into a similarity value and averaging the similarities. Depending on the averaged similarities, and the threshold, the output of the classification can be 0 or 1. In the masquerade detection context, 0 means that the processed object represents masquerader behavior; an output of 1 indicates the contrary.

**BRM:** *Bagging-RandomMiner* [29] is a classifier ensemble that consists of multiple instances of RandomMiner. The rationale behind the algorithm of this classifier is similar to the one used in Bagging Random Miner; these two classifiers differ only at the training phase. At BRM's training phase, the base classifier randomly selects a percentage of the instances in a given training dataset. By performing a random selection, the classifier has the advantage of taking into account common user behavior zones and give less importance to objects located in isolated zones. The obtained samples in this phase can resemble the set of *typical* objects, similar to the ones obtained by BTPM, with the difference that no inter-object distance calculation is required. By doing this, building the model requires significantly less time. Furthermore, the classification results yielded, in an access-based masquerade detection context, do not have a significant statistical difference compared to the ones of Bagging-TPMiner.

For the classification phase, the algorithm receives as input an ensemble of trained classifiers generated in the previous phase and the object to classify. Each classifier computes the minimum distance between the object to be classified and an object in the representative behavior cluster. This distance is used to calculate a similarity value.

The similarity values of all classifiers in the ensemble are averaged, and then a final score is calculated. This score is used as the output classification.

**OCKRA:** *One-Class The K-means with Randomly-projected features Algorithm* [99], consists of an ensemble of one-class classifiers based on k-means++ [13]. This method works by creating multiple projections of the dataset using a random subset of features, thus obtaining a high diversity among the classifiers in the ensemble. This mechanism is analogous to other ensemble methods such as Random Forest. The training phase begins with an initial training dataset. Random feature selection with replacement is then applied. Afterward, duplicated features are removed; authors have found that this process extracts approximately 63% of the original features on average. Then, a new training dataset is created using the subset of selected features; further, the algorithm uses k-means++ to obtain  $k$  clusters and calculates their centroids. Next, centroids are used to construct one-class classifiers that return the similarity of a new, queried object, to the normal class. For each of the individual classifiers, the training phase returns a set comprising the parameters consisting of three elements: the randomly-selected features, the computed centroids of each cluster, and the distance threshold.

At the classification phase, every classifier in the ensemble makes a projection of the queried object using its subset of features. Next, the Euclidean distance of the projected object to the centroids is calculated, and the nearest cluster is obtained. Next, the distance between the projected object and its nearest cluster centroid is transformed into a similarity value in the interval  $[0, 1]$ . Once all similarity values are calculated, OCKRA calculates their average. A threshold is then set to determine if the object represent normal or abnormal behavior. The output of OCKRA is a single number; a value of zero indicates that the object represents abnormal behavior, and a value of one represents normal behavior. The accuracy of this ensemble method relies to some extent on the threshold used for determining which objects are anomalous and which are not.

**OC-SVMs:** The Support Vector Machines (SVMs) learning models, which are capable of performing classification and regression analysis, were first introduced by Vapnik et. al [120, 23]. Later, extensions to the SVM framework were developed [30]. SVMs construct, in a multidimensional space, hyperplanes that separate the objects according to their class. To perform an optimal construction of the hyperplane, SVM makes use of an iterative training algorithm, whose objective is to minimize an error function. In general, SVMs have a robust performance when dealing with sparse and noisy data, which makes them suitable for several application domains.

*One-Class Support Vector Machines* (OC-SVMs) are an extension of SVMs. At the training phase, OC-SVMs aim to find, in a projected space, a hyperplane that maximizes the separation of the instances. Given that only one class is present in the training phase, instead of looking for the maximum separation between classes, this method looks for the maximum separation between the origin and the instances in the projected space. Regarding the computational complexity, the training phase of OC-SVMs deals with a quadratic problem [23]. Nevertheless, once the algorithm is trained, the classification process's complexity is minimal.

**OC-NB:** The *Naïve Bayes* [113, 127] classifier is an efficient and simple supervised learning algorithm based on Bayes' Theorem. In general, this classifier calculates the probability that an object belongs to a certain class based on the values of its features. This classifier is called *naïve* because it assumes that the value of a given feature is independent of the value of other ones, which is not always true in real-life applications.

In a binary *Naïve Bayes* implementation, the highest probability of membership indicates the class to which the object belongs. In contrast, a one-class *Naïve Bayes* classifier works by considering only the probability of the queried object of belonging to the normal class. According to a threshold, the probability obtained is used to determine if the object represents normal or abnormal behavior.

**Isolation Forest:** *Isolation Forest* was originally proposed by Liu et al. [79]. This anomaly detection algorithm has been shown to perform well in high dimensional problems, even if they contain a large number of irrelevant attributes. Furthermore, it is still robust when anomalies are not available in training sample. As described by Liu et al. [79], *Isolation Forest* detects anomalies based on the concept of isolation and recursion. As a consequence, it has the ability to exploit subsampling to achieve a low linear time complexity, low memory consumption, and the capacity to deal effectively with the effects of swamping and masking. By definition, anomalies are uncommon, new and different. These characteristics make outliers susceptible to the isolation mechanism.

### 3.4.5 Selected Performance Measure

To determine if a given instance is normal or anomalous, most of the selected one-class classifiers require a threshold. This threshold may be chosen based on the needs of the user, such as increasing the detection power of anomalies or reducing the number of false positives. In other words, the threshold to select is application- and implementation-dependent. Because of that, ROC curves are used to observe the classifiers' performance on different thresholds. Moreover, the selected AUC measure summarizes the performance of a classifier into a single metric [15]. Therefore, the results of the experiments are reported using the AUC of the ROC curve is used.

AUC has the advantage of being a balanced performance measure and robust against class imbalanced problems. The AUC performance of a classifier is equivalent to the probability that the classifier ranks a randomly chosen positive instance higher than a randomly chosen negative instance [45, 16]. Furthermore, the AUC measure is capable of describing performance correctly, even if the dataset contains more objects of one class than another [95]. Since most of the experiments use fewer examples of each bot type than legitimate account examples, they are categorized as class imbalance problems. Therefore, it is appropriate to choose AUC as the performance measure.

## 3.5 Summary

One-class classifiers are used mostly for anomaly detection, where the objective is to detect if a given instance represents *normal* or *abnormal* behavior. This type of classifiers are different from binary and multi-class, in the sense that they only require examples of one class to learn representations of them and generalize the patterns that represent *normal* behavior. In the context of bot detection, we hypothesize that one-class classifiers could help overcome the main limitation of current state-of-the-art-classifiers, which is the requirement of possessing bot types examples beforehand to detect them effectively. Concretely, training one-class classifiers with genuine (human) users examples might give them the capacity to detect different bots, regardless of their type. The main limitation of this proposal is that, as with other approaches, the effectiveness of the classifier highly relies on the feature representation chosen to represent the account's behavior. Also, the effectiveness is dependant on the overall difference in the behavior of bots and genuine users. To prove if this hypothesis holds, we designed three experiments. These experiments aim to answer the following research questions: 1) Does the performance of multi-class methods decrease when it comes to identifying bots of an unknown class? 2) Do one-class classifiers perform better than multi-class methods when it comes to identifying bots of an unknown class? 3) When identifying bots of an unknown class, is there a statistically significant difference between the performances obtained by the two methods? The detailed description of the experimental setup is addressed in the following chapter.

# Chapter 4

## Experimental Setup

To test the advantages and limitations of a semi-supervised anomaly detection approach for bot detection, we designed three experiment sets. These experiments aim to compare fairly and directly the performance of multi-class and one-class classifiers. In this chapter, the experimentation procedure is described in detail. The following sections describe the procedures for constructing training and testing sets, the performance measure calculation, and the procedure for assessing the significance of the results.

### 4.1 Experiments

Three experiment sets were designed to answer the research questions, which aim to test the hypothesis that one-class classifiers may be able to detect instances of new, previously unseen, classes of bots. Thus, overcoming the main limitation of state-of-the-art bot detectors.

The experiment sets are labeled with letters. Experiment Set A deals with Binary Classifiers, experiment set B involves multi-class versions of the binary classifiers, and Experiment Set C is related to one-class classifiers.

#### 4.1.1 Experiment Set A (Binary Classifiers) Description

In experiment set A, four training sets were constructed, one for each bot type. Each one of these training datasets contains 90% of the legitimate account instances and 90% of the instances of a particular bot type. Note that, to avoid a performance variance due to different legitimate account examples, all of the training sets contain the same 90% legitimate account instances, regardless of the bot type used. Only the bot examples change among training sets; one type of bots is used at a time. For example, the *social1* training set contains 90% of legitimate account instances and 90% of the instances of social 1 type bots; the *social2* training set contains 90% of legitimate account instances and 90% of the instances of social 2 type bots, and so on. For the testing phase, the constructed dataset contains the remaining 10% of the legitimate account examples and the remaining 10% examples of the corresponding bot type. The classification process was carried out ten times to ensure statistical validity. Each time with a different 10% of bots and legitimate account examples, thus resulting in a 10-fold cross-validation procedure. See Section [4.2](#) for a detailed explanation of this procedure.

The experiment set A is intended to test the hypothesis presented by Cresci et al. [38]. Cresci et al. hypothesized that the bot's behavior varies according to their type. This hypothesis is tested by measuring the performance of different binary classifiers when classifying various groups of bots and legitimate accounts. To obtain the performance of the binary classifiers, they were trained using the training sets described above, and then each trained model was tested with all the different testing datasets, one at the time. For example, a binary classifier  $C_1$  was trained using the training set *social1*, which contains 90% of genuine account examples and 10% of social 1 type bots. Then, this trained model was tested four different times. First, the model was tested using the *social1* testing set, which contains 10% of the remaining social 1 type bots and 10% of genuine examples. Then, the model is tested using the *social2* testing set, which contains 10% of social 2 type bots, and the same 10% genuine examples as *social1*. This process continues until the four testing sets (*social1*, *social2*, *social1*, and *traditional1*) are given to the trained model. Afterward, the classifier  $C_1$  is trained with the next training set in the list, and the testing procedure repeated. This experiment gives an insight into how the performance of a given classifier changes when performing classification on a type of bot that is different from the one used in its training phase.

#### 4.1.2 Experiment Set B (Multi-class Classifiers) Description

The objective of the experiment set B is to simulate more closely how classifiers can be used to detect bots. This simulation is based on the strategy implemented by most publicly available bot detection tools. The strategy consists of using a training set that contains instances of different bot types. In most cases, this diversity in bot behavior examples allows the classifier to learn better how to discern between legitimate and bot accounts [129].

In experiment set B, the percentage of bots and genuine users are the same used in experiment set A. What differentiates this experiment set from the previous one is that the classifiers are trained with examples of more than one bot type. At the training phase, a set containing instances of genuine users and three (out of four) types of bots is used. Then, at the testing phase, a set containing the remaining type of bots, together with genuine user examples, is used. This procedure is repeated until the four types of bots are tested. As in experiment set A, a 10-fold cross-validation procedure is carried out to obtain the AUCs of the different tests.

#### 4.1.3 Experiment Set C (One-class Classifiers) Description

The objective of the experiment set C is to show how one-class classifiers perform when they are trained (only) with genuine (human) account examples, and tested with the different types of bots. For the training and testing steps, this experiment set follows the same steps as the ones used in experiment set A. Here, the main difference relies on the classifiers mechanisms; at training phase, one-class classifiers ignore the bot examples, and focuses only on the 90% of the human user account instances for constructing the model. After the models are trained, each classifier is evaluated using the same testing sets and cross-validation procedure used in experiment set A. For each experiment, when each cross-validation is done, the performance score is recorded.

## 4.2 Performance validation

When an algorithm is trained and evaluated using the same data, its performance tends to be deceptively large [75]. To avoid over-estimations of a given algorithm's performance, data is divided into two sets: training and testing. The training set is used for the algorithm to learn representations of the data; the testing set is used to assess the performance of the algorithm [12]. This procedure is known as a single hold-out method [19]. To better estimate the performance an algorithm has in a real scenario, a  $k$ -fold cross-validation procedure can be carried out. Cross-validation allows us to assess the generalization capacity of a predictive model, also helps to prevent overfitting [19]. Moreover, the variance of the resulting estimate reduces as  $k$  is increased [54].

Precisely,  $k$ -fold cross-validation consists of dividing the data into  $k$  subsets and using  $k-1$  of them to train the algorithm. Then, the model is tested using the remaining subset, sometimes called the validation set. This process is repeated until each of the  $k$  subsets has been tested. For each iteration, the performance is measured. Once all subsets have been run, the average of the  $k$  performance measurements is calculated and reported [19].

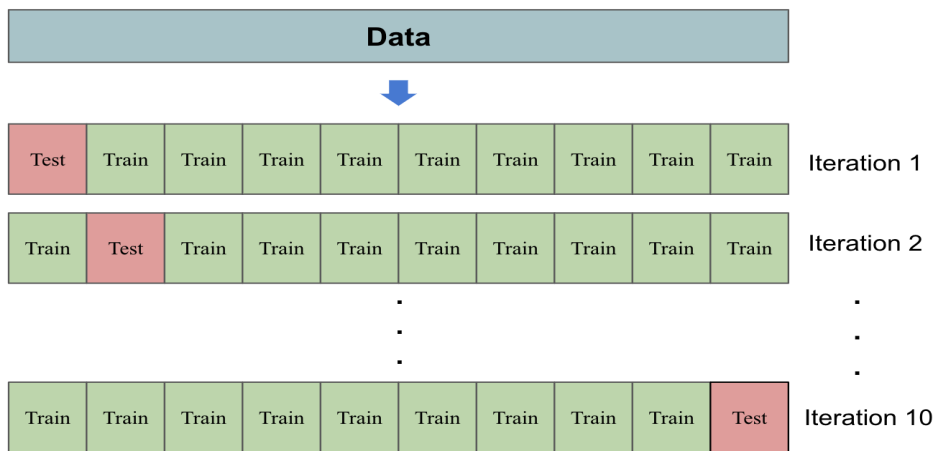


Figure 4.1: Data partitions performed by a 10-fold cross-validation procedure.

Due to its advantages, a  $k$ -fold cross-validation procedure was chosen to measure the performance of the algorithms. For each experiment, ten-fold cross-validation was done, so in the result tables, the number presented is the average of the AUC per classifier, and bot type in the testing dataset; standards deviations are also included. Figure 4.1 illustrates how a 10-fold cross-validation procedure divides the data.

## 4.3 Testing the significance of results

To determine if there exists statistical differences between the performance of the classifiers, the Friedman test with a Bergmann-Hommel and Schaffer post-hocs were used [53]. To provide a visual representation of the results obtained from the statistical tests *Critical Difference*

(CD) diagrams [41] are used. CD diagrams present the rank of an algorithm concerning a performance indicator, in this case, the AUC value. Chapter 5 presents the results of the experiments. In a CD diagram, the top-ranked algorithm appears rightmost, and a thick line joins statistically similar algorithms. These features of CD diagrams give information regarding both the quantity and significance of any difference between algorithms. Figure 4.2 shows an example of a CD diagram where, despite being the top-ranked, Algorithm A does not show a statistically significant difference from Algorithm B in terms of performance. Moreover, algorithms C and D do not show statistical differences between them; the same is true for algorithms D and E.

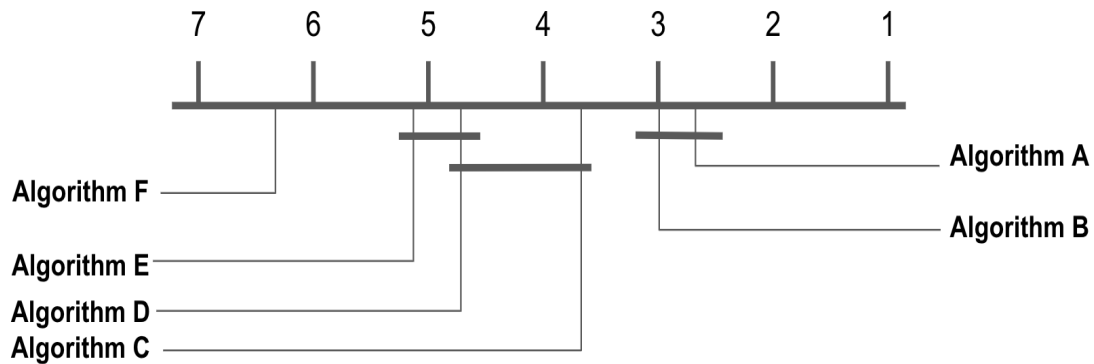


Figure 4.2: Example of a Critical Difference diagram.

## 4.4 Summary

This chapter explains the experimental setup used to test the advantages and limitations of a semi-supervised anomaly detection approach for bot detection. The datasets used in our experiments were created by Cresci et al. [38] and downloaded from a public repository. The original features from the files in the dataset were used to extract a novel feature vector, which has their basis on the previous works reported in the literature. Several authors have demonstrated that the features that make up the proposed feature vector provide information that is useful for characterizing the real behavior of the accounts. Moreover, the datasets were used to train ten widely-used binary classifiers and five one-class classifiers. Some of these one-class classifiers belong to the state-of-the-art on intrusion detection. The three experiments designed have the objective of providing a fair and direct comparison of the classifiers' performance. The first two experiments involve binary and multi-class versions of the same supervised classifiers. The third experiment involves the selected one-class classifiers. A ten-fold cross-validation procedure was carried out to validate the performance of the tested classifiers.



# Chapter 5

## Experimental Results and Discussion

This chapter presents the outcome of the three different experiments sets described in Chapter 4, along with the results of statistical comparisons between the performance of the classifiers used in each set. To determine if a given instance is normal or anomalous, most of the selected one-class classifiers require a threshold. This threshold may be chosen based on the needs of the user, such as increasing the detection power of anomalies or reducing the number of false positives. In other words, the threshold to select is application- and implementation-dependent. Because of those reasons, receiver operating characteristic (ROC) curves were used to observe the performance of the classifiers on different thresholds.

To report the performances of the classifiers, the area under the curve (AUC) of the ROC curve is used, as it summarizes the performance of a classifier into a single metric [15]. Furthermore, the AUC performance measure is capable of describing performance correctly, even if the dataset contains more objects of one class than another [95]. Since most of the experiments use fewer examples of each bot type than legitimate account examples, they are categorized as class imbalance problems. Therefore, it is appropriate to choose AUC as the performance measure. For each experiment, ten-fold cross-validation was done. In the result tables, the number presented is the average of the AUC per classifier, and per bot type in the testing dataset. Such tables also include standards deviations. A box and whisker diagrams accompany every set of experiments to simplify the understanding of all the information presented in the tables, box and whisker diagrams accompany every set of experiments.

The organization of this chapter is as follows. First, the results of experiments A, B, and C are presented. Next, a comparison of the results of each experiment set is made. Then, a discussion section follows. To conclude, a summary of the chapter is provided.

### 5.1 Results of Experiment set A (Binary Classifiers)

In this first experiment set, the hypothesis of Cresci et al. [38], that different types of bots have different behaviors, is tested. These experiments consist of different supervised binary classifiers trained with a set containing instances of genuine users and one type of bots and then tested using a set containing different types of bots. If the hypothesis is correct, then the behavior shown by the different bot types differ from one another. The hypothesis is accepted if the classifiers' performance, when classifying the same type of bot, is outstanding, but the

performance decreases when classifying other bot types.

### 5.1.1 Classifiers Trained with Social1 Type Bots and Genuine Accounts

The experiment begins by training the classifiers with examples of legitimate and social1 bot accounts. In Table 5.1, each column shows the type of bot used in the testing set, and the average performance obtained by all the classifiers. The results of this experiment show that the best AUC is obtained when performing the classification of bots of the same type. Bot accounts of type social1 are the best classified, with a mean AUC of 0.974 and a standard deviation of 0.01. The second-highest mean AUC, 0.901, is obtained for bots of type social3. When training against this type of bots, the performance is still acceptable, but the standard deviation increases considerably with respect to social1. These results could signal that bots of type social1 and social3 share characteristics, which allows the classifier to identify them even if they belong to different bot categories. The type of bots that disseminate spam and malware, traditional bots, cannot be easily identified when classifiers are trained with examples of social1 bots since the average AUC obtained is 0.692. The same happens for bots of type social2, where the average AUC is 0.56. As for the performance classifier-wise, Figure 5.1 depicts the overall superiority of PBC4cip, whose performance remains stable across the different testing sets. The rest of the binary classifiers show standard deviations close to 2, and a mean performance that does not exceed 0.82 AUC.

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	<b>0.972</b>	0.5	0.959	0.76	0.798	0.221
J48	<b>0.977</b>	0.517	0.957	0.733	0.796	0.216
Random Forest	<b>0.98</b>	0.496	0.96	0.719	0.789	0.228
Adaboost	<b>0.979</b>	0.497	0.948	0.772	0.799	0.221
Bagging	<b>0.975</b>	0.497	0.954	0.78	0.802	0.221
KNN	<b>0.974</b>	0.491	0.584	0.566	0.654	0.217
Logistic Regression	<b>0.969</b>	0.74	0.957	0.588	0.814	0.184
MLP	<b>0.97</b>	0.488	0.714	0.526	0.675	0.22
Naïve Bayes	<b>0.958</b>	0.477	0.937	0.523	0.724	0.259
SVM	<b>0.96</b>	0.482	0.951	0.658	0.763	0.234
PBC4cip	<b>0.996</b>	0.974	0.995	0.989	0.989	0.010
Mean	0.974	0.56	0.901	0.692		
Standard Deviation	0.01	0.156	0.129	0.139		

Table 5.1: Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type social1 account examples. A bold typeface is used to convey the results of testing the same type of bots used in the training set.

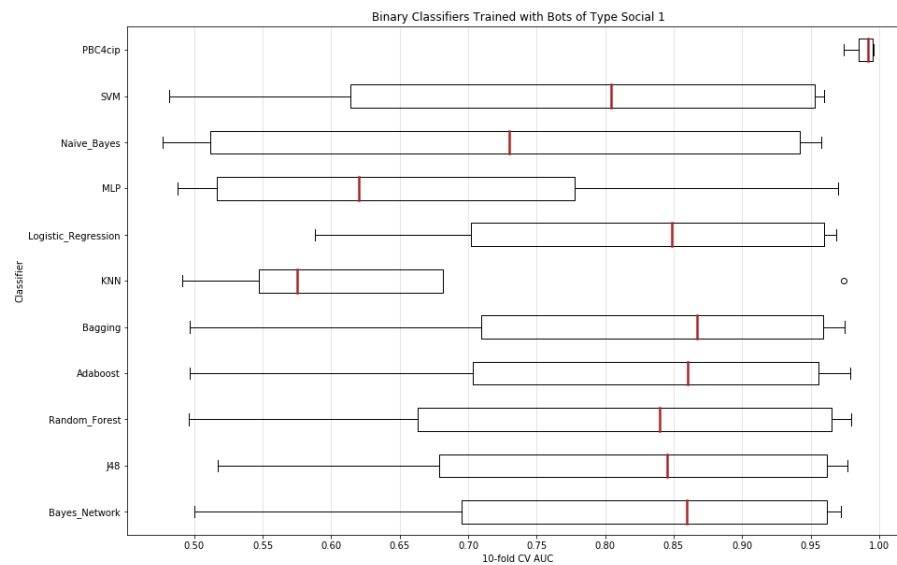


Figure 5.1: Performance of binary classifiers when using social1 bots in the training phase, and different bots at testing.

### 5.1.2 Classifiers Trained with Social2 Type Bots and Genuine Accounts

Table 5.2 shows the AUC performance the classifiers yielded when trained with instances of legitimate accounts and bots of type social2. Binary classifiers show excellent performance when differentiating legitimate accounts and bots of the same type as the ones used in training. Classification of social2 bot types results in a mean AUC of 0.991 and a standard deviation of 0.004. However, most classifiers perform poorly if an instance of a different bot type is shown to them, as the mean AUCs for the other types of bots are 0.577, 0.599, and 0.613. This low performance may indicate that social2 bots examples do not provide relevant information to distinguish between genuine accounts and other types of bots. Similar to the previous experiment, the mean AUC for all classifiers is always lower than 0.78. Similar to the previous experiment, PBC4cip shows a robust classification performance, even when new bot types are present in the test set. A 0.998 mean AUC is yielded by this classifier, with an approximate standard deviation of 0. Figure 5.2 illustrates that one more time, PBC4cip the best performing algorithm in the experiment.

### 5.1.3 Classifiers Trained with Social3 Type Bots and Genuine Accounts

Table 5.3 displays the results obtained when using classifiers trained with examples of social3 type bots and legitimate accounts. The results for this classifiers are consistent with the ones obtained in Tables 5.1 and 5.2, where the higher performance results occur when the classifiers are used to discern between legitimate accounts and bot accounts of the same type. When classifiers are trained with social3 type bots, the classifications of the same types of bots led to an average AUC of 0.966. The standard deviation obtained for this case was 0.013. These

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	0.5	<b>0.989</b>	0.514	0.502	0.626	0.242
J48	0.68	<b>0.992</b>	0.596	0.835	0.776	0.175
Random Forest	0.502	<b>0.996</b>	0.508	0.511	0.629	0.245
Adaboost	0.502	<b>0.993</b>	0.518	0.589	0.651	0.231
Bagging	0.633	<b>0.993</b>	0.535	0.673	0.709	0.198
KNN	0.493	<b>0.99</b>	0.494	0.499	0.619	0.247
Logistic Regression	0.511	<b>0.987</b>	0.907	0.604	0.752	0.23
MLP	0.519	<b>0.989</b>	0.513	0.529	0.638	0.234
Naive Bayes	0.498	<b>0.99</b>	0.498	0.5	0.622	0.246
SVM	0.511	<b>0.986</b>	0.509	0.505	0.628	0.239
PBC4cip	0.998	<b>0.998</b>	0.998	0.998	0.998	0.00
Mean	0.577	0.991	0.599	0.613		
Standard Deviation	0.153	0.004	0.178	0.164		

Table 5.2: Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type social2 account examples. A bold typeface is used to convey the results of testing the same type of bots used in the training set.

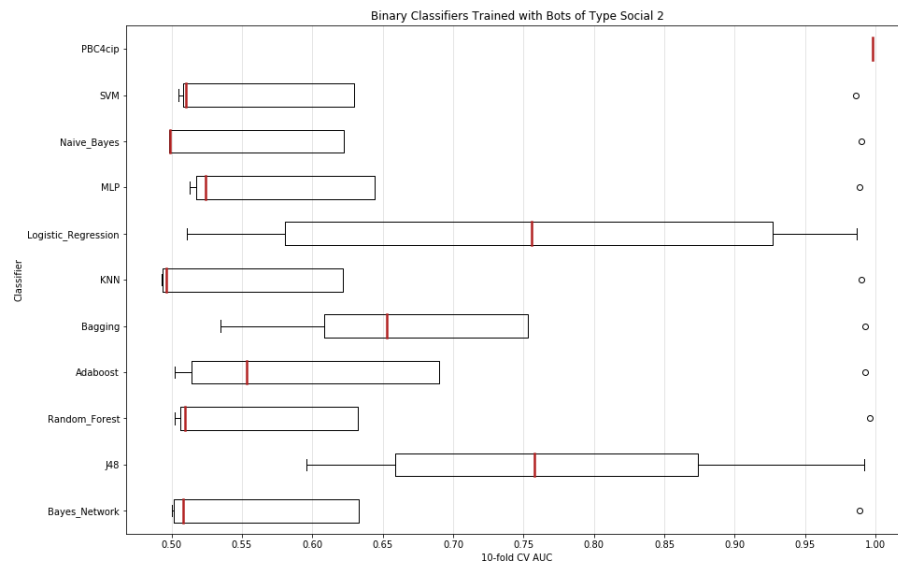


Figure 5.2: Performance results of the experiment set A when using social2 bots in training.

results indicate that a stable behavior from all classifiers is obtained when classifying bots of the same type as the ones used in training. The results presented in Table 5.1 suggest that the behavior of accounts of type social1 could be used to identify bots of type social3 for some classifiers correctly. Such is the case for KNN and Logistic Regression.

Nevertheless, while classifiers trained with bots of type social3 can also help discern bots of type social1, the AUC obtained is generally lower, shown with mean AUCs of 0.789. For classifiers such as SVM and Bayes Network, generalization capabilities are low, as they perform lower than other classifiers, leading to AUC values of 0.494 and 0.590, respectively. Moreover, the trained classifiers continue to show a low performance when discerning between legitimate users and bots of the traditional type, as the mean AUC is 0.769. Even a worse performance is shown when the classifiers attempt to distinguish between bots of type social2 and genuine users, where all the obtained AUC values are lower than 0.6, and the mean value is 0.559, with a standard deviation of 0.196.

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	0.59	0.599	<b>0.963</b>	0.774	0.732	0.176
J48	0.712	0.553	<b>0.97</b>	0.774	0.752	0.172
Random Forest	0.909	0.497	<b>0.966</b>	0.78	0.788	0.209
Adaboost	0.88	0.496	<b>0.968</b>	0.787	0.783	0.205
Bagging	0.722	0.497	<b>0.968</b>	0.756	0.736	0.193
KNN	0.91	0.491	<b>0.959</b>	0.621	0.745	0.226
Logistic Regression	0.971	0.984	<b>0.97</b>	0.741	0.917	0.117
MLP	0.887	0.496	<b>0.966</b>	0.695	0.761	0.21
Naive Bayes	0.812	0.497	<b>0.956</b>	0.643	0.727	0.2
SVM	0.494	0.484	<b>0.942</b>	0.891	0.703	0.248
PBC4cip	0.994	0.997	<b>0.996</b>	0.995	0.996	0.001
Mean	0.807	0.599	0.966	0.769		
Standard Deviation	0.16	0.196	0.013	0.105		

Table 5.3: Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type social3 account examples. A bold typeface is used to convey the results of testing the same type of bots used in the training set.

#### 5.1.4 Classifiers Trained with Traditional Type Bots and Genuine Accounts

The last round of binary classifiers experiments was done by training the algorithms with examples of legitimate and traditional bot accounts; 5.4 presents the results obtained. If the results obtained when using the same bot type in training and testing are compared, the second-best performance is obtained for the traditional bot type, with a mean AUC for all classifiers of 0.983, and a standard deviation of 0.026. The mean AUC, compared per bot type, obtained for traditional bots is higher by at least 0.42 points than the mean of other types of bots. It is true as well that, except for some cases, the classifiers trained with traditional type bots achieve lower values of AUC when detecting the other types of bots. The notable exception is PBC4cip, which always performs above .978 AUC. The mean AUC values obtained per bot type are 0.504 for social1 bots, 0.526 for social2 bots, and 0.719 for social3 bots. The

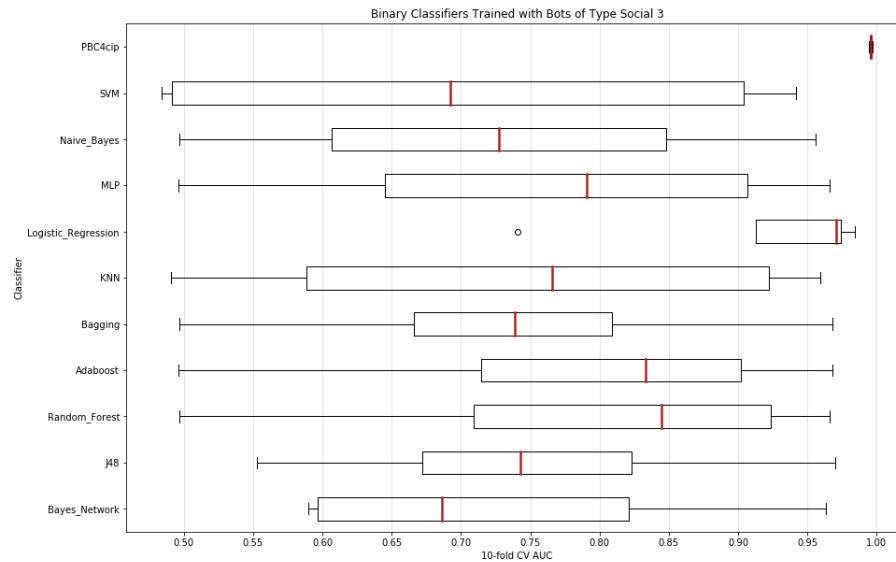


Figure 5.3: Performance results of the experiment set A when using social3 bots in training. Note that Logistic Regression yield the second-best performance, only outperformed by PBC4cip.

mean value for social3 equals the one of social2, nevertheless it also has the most prominent variation, as indicated by the standard deviation of 0.231. These results could signify that traditional bot accounts have a pattern of behavior that does not allow classifiers to easily discern other types of automated accounts more engaged in the social aspect of Twitter. This situation clearly reflects in figure 5.4, specially for Logistic Regression, Bagging, Adaboost, Random Forest, and J48.

### 5.1.5 Summary of Results of Experiment Set A

In this experiment set, the null hypothesis is that binary bot detection methods perform similarly, regardless of the type of bot used in the training phase. For all the cases presented, when the same type of bots was used in both datasets (training and testing), outstanding results were achieved; most AUC values are above 0.95, and standard deviations are low. This similarity in AUC and standard deviations means that almost all classifiers have similar classification performance. Regardless of these excellent results, the same classifiers would have low performance when used to discern between instances of legitimate user accounts and those belonging to a different type of bot than the one used in the training set. Figure 5.5 summarizes the difference in performance, per bot type, for all of the experiments carried in this section. For each box and whisker plot, it can be observed that the boy type with highest performance was the one used in the training phase. Moreover, Figure 5.6 shows the CD diagrams for each of the four experiments carried out in this section. Recall that groups whose difference is not statistically significance are joined with a thick line. In each CD diagram

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	0.503	0.501	0.942	<b>0.995</b>	0.735	0.27
J48	0.5	0.502	0.503	<b>1</b>	0.626	0.249
Random Forest	0.501	0.502	0.504	<b>0.999</b>	0.627	0.248
Adaboost	0.5	0.5	0.5	<b>0.999</b>	0.625	0.25
Bagging	0.5	0.502	0.503	<b>0.999</b>	0.626	0.249
KNN	0.489	0.48	0.928	<b>0.952</b>	0.712	0.263
Logistic Regression	0.501	0.502	0.511	<b>0.999</b>	0.628	0.247
MLP	0.536	0.561	0.933	<b>0.963</b>	0.748	0.231
Naive Bayes	0.523	0.729	0.929	<b>0.985</b>	0.792	0.21
SVM	0.489	0.481	0.939	<b>0.922</b>	0.708	0.257
PBC4cip	0.979	0.987	0.995	<b>1</b>	0.990	0.009
Mean	0.547	0.568	0.568	0.983		
Standard Deviation	0.144	0.156	0.231	0.026		

Table 5.4: Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type traditional account examples. A bold typeface is used to convey the results of testing the same type of bots used in the training set.

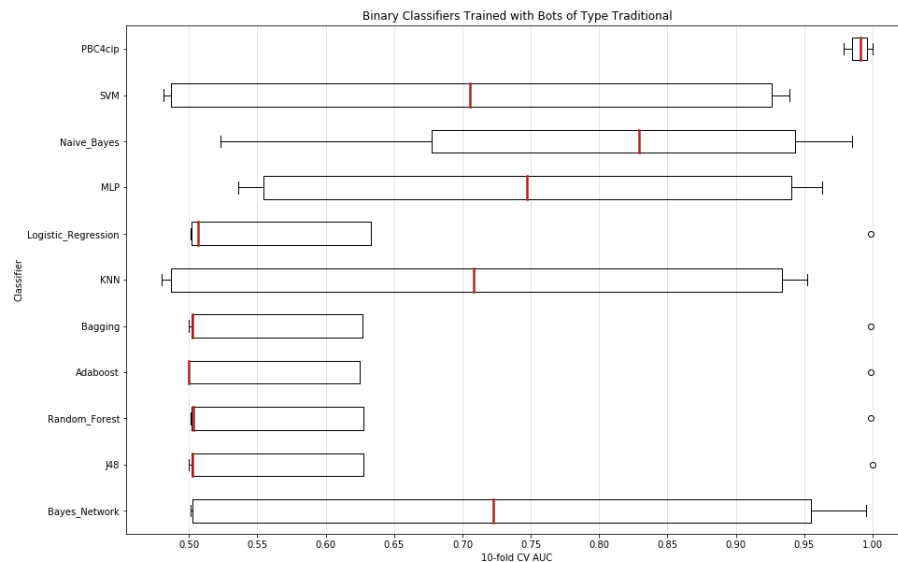


Figure 5.4: Performance results of the experiment set A when using traditional bots in training.

presented it can be seen that there is at least one bot type whose classification performance is statistically different from the other bot types. This evidence implies we can reject the null hypothesis. Thus, the initial hypothesis that different bot types express different behavior when

the set of features presented in Section 3.4.3 is used to characterize its behavior, is confirmed. Moreover, results indicate that most binary classification methods are not the best suited for automatic bot detection.

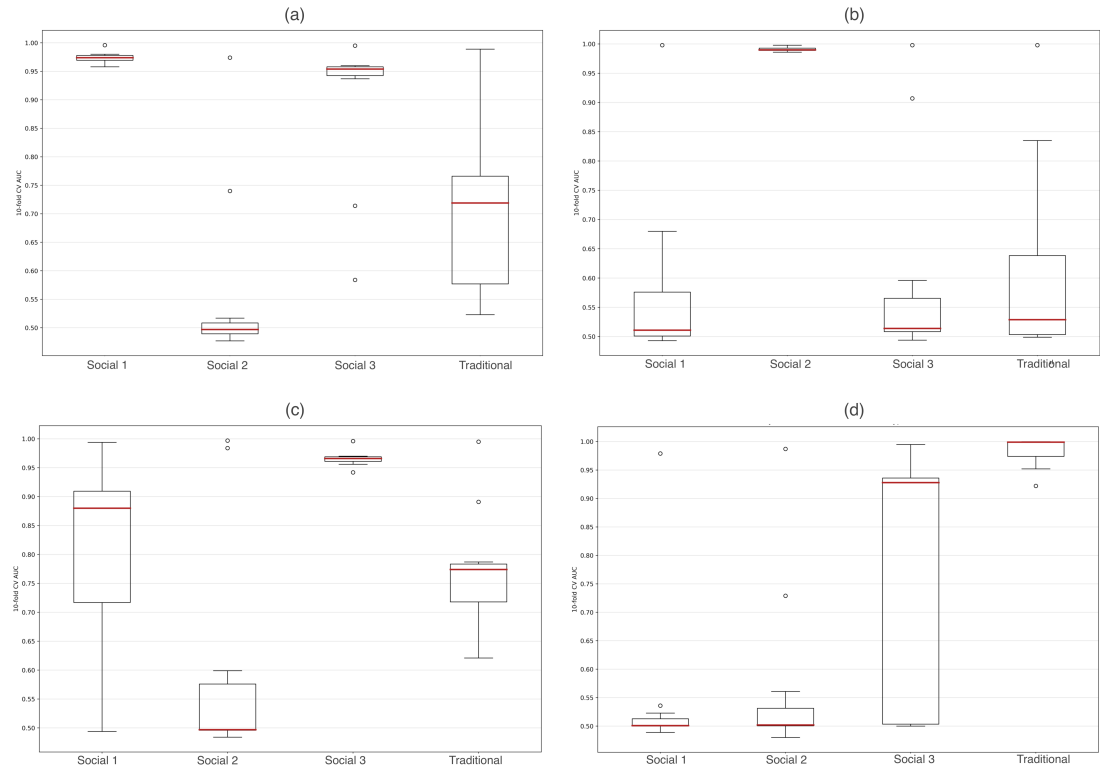


Figure 5.5: Box and whisker plots of the AUC performance, per bot type, for each table in Experiment Set A. The plots are labeled in order of appearance.

As presented in Section 3.3.1, bot's design is continuously evolving to avoid detection mechanisms, and different strategies for interaction with users are being developed. Given those facts, automatic bot detection mechanisms based on binary classifiers require to always keep up-to-date with different bot type account examples to continue being reliable. This experiment has shown that simple classifiers trained for a specific bot type do not allow to classify the other types correctly. On the other hand, methods based on contrast patterns, such as PBC4cip, have the potential to overcome the lack of examples of novel bots. In the next section, we show the results obtained when using classifiers that are trained with multiple bot types and tested with an unseen bot type.

## 5.2 Results of Experiment set B (Multi-class Classifiers)

Experiment set B deals with different models constructed from the same base classifiers presented in the previous section. The name *multi-class* was chosen to depict that bots of multiple



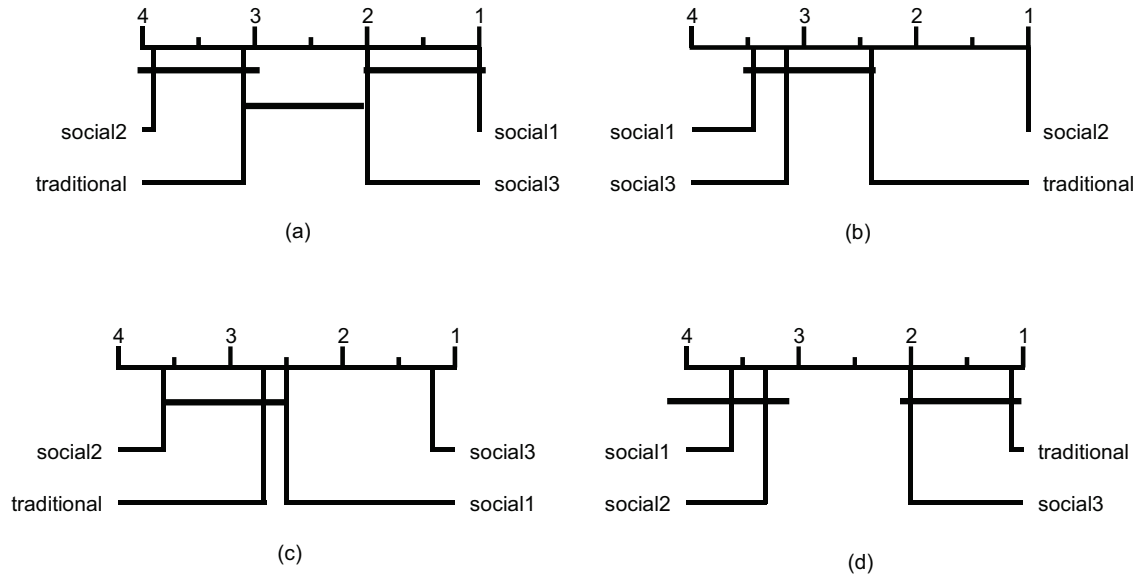


Figure 5.6: CD diagrams showing the significant statistical differences among the different types of bots. Each diagram corresponds to a table in experiment set A, in order of appearance.

types are used in the training phase. Nevertheless, all of these automated accounts are labeled under the same class: *bot*. In this experiment set, as well as the others, the classifiers are required to perform binary classification. In the following paragraphs, the results obtained when training the classifiers with 3 out of 4 types of bots and performing classification to the examples of the remaining bot type are analyzed. The experiments presented in this section simulate more closely how classifiers are used to detect bots. This simulation is done by imitating the strategy of most publicly available bot detection tools, which use a more diverse set of examples that encompasses multiple behaviors that could be used to discern between legitimate and bot accounts correctly. The null hypothesis for this experiment is that multi-class methods which are trained with three different bot types, and tested against a different one, yield a similar performance for each testing case.

The average AUC obtained by each classifier, when classifying a set containing a different type of bot than the ones used in training, is shown in Table 5.5. Analyzing the overall performance by bot type, it can be found that classifying bots of type social3 is more manageable for the different classifiers, as the average AUC obtained is 0.946. However, the mean value is 0.02 lower than the one obtained when using classifiers that are trained only with bots of type social 3, as shown in Table 5.3. On the other hand, the worst mean performance, an average AUC of 0.662, is obtained when the previously unseen bot is of type social2. These results corroborate the ones in Table 5.2, where the classifiers trained with bot examples of type social2 did not learn to discern the bot behavior of other types.

Table 5.5 shows that the second-highest performance for this set of experiments, an average AUC of 0.903, is obtained when using Logistic Regression. However, the Logistic Regression classifier has the disadvantage of performing poorly when the unseen bot examples are of type traditional; it obtains a 0.726 AUC in those cases. The third-best performance is an average AUC of 0.886, which is obtained when using the J48 classifier. Although it

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	0.578	0.539	0.957	0.774	0.712	0.193
J48	0.874	0.88	0.887	0.902	0.886	0.012
Random Forest	0.935	0.496	0.957	0.826	0.804	0.213
Adaboost	0.922	0.515	0.942	0.869	0.812	0.2
Bagging	0.928	0.499	0.954	0.933	0.829	0.22
KNN	0.893	0.537	0.941	0.609	0.745	0.202
Logistic Regression	0.956	0.973	0.957	0.726	0.903	0.118
MLP	0.858	0.717	0.953	0.729	0.814	0.112
Naïve Bayes	0.81	0.609	0.943	0.625	0.747	0.16
SVM	0.533	0.528	0.937	0.873	0.718	0.218
PBC4cip	<b>0.995</b>	<b>0.987</b>	<b>0.979</b>	<b>0.994</b>	0.989	0.007
Mean	0.844	0.662	0.946	0.805		
Standard Deviation	0.151	0.195	0.023	0.124		

Table 5.5: Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of different types. A bold typeface is used to convey the best AUC obtained by a classifier when testing a specific type of bot.

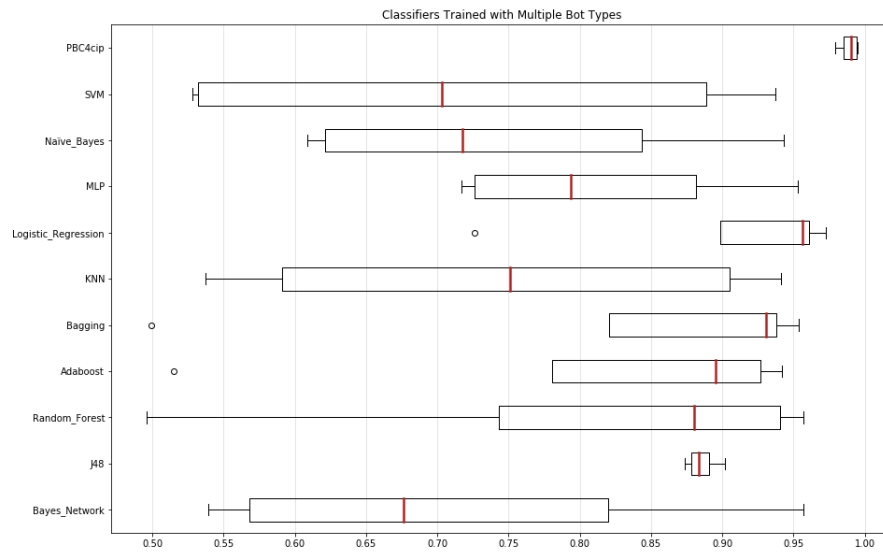


Figure 5.7: Performance results of the experiment set B.

does not have the best mean performance, J48 has a standard deviation of 0.012, which is one order of magnitude lower than the other classifiers. This standard deviation suggests that J48 and PBC4cip are the most stable classifiers. Additionally, both classifiers perform well regardless of the unseen type of bot in the testing dataset. It can also be seen in Table 5.5 that

all classifiers, except for PBC4cip and J48, have experiments where the performance is drastically lower. The box and whiskers plot shown in Figure 5.7 reflects the stability of PBC4cip and J48, as the boxes and tails are the smallest of all the classifiers. CD diagrams presented in Figure 5.8 are in accordance with the information presented by the box and whiskers plot. Further, the statistical tests show there are groups of bot types whose performance is statistically similar. Therefore, there is no evidence to reject the null hypothesis. In the next section, we show the results of using one-class classifiers, which do not rely on examples of the different bot types but use only genuine user examples instead.

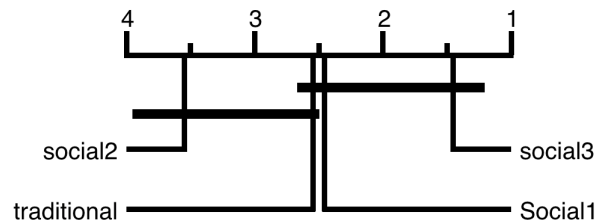


Figure 5.8: CD diagram of the performance in experiment set B. In this experiments, classifiers are trained with three types of bots and tested with the remaining one.

### 5.3 Results of Experiment set C (One-class Classifiers)

One class classifiers have the particularity of only requiring only legitimate user accounts in their training phase. Therefore, the training dataset created for the previous experiments can be used without further modifications. The current section presents the performance of the selected one-class classifiers when trained with legitimate accounts, and tested with datasets that contain legitimate and bot account examples of only one type. The null hypothesis for experiment set C is that the performance of one-class classifiers remains stable when tested against a dataset containing genuine and bot activity, regardless of the bot type used.

The average AUC performance obtained when using one-class methods to classify the testing datasets containing different bot examples is shown in Table 5.6. The highest AUC obtained by a classifier for each type of bot is depicted in bold numbers. Regarding the classifiers used, the original implementations of BTPM and BRM use Mahalanobis distance, nevertheless it was found that using Euclidean distance increase their performance in the bot detection context. Thus, in this experiment set we include both version of the classifiers. In this experiment set it was observed that Näive Bayes has better performance when detecting bots of type social1 and Bagging-TPminer (Euclidean) when detecting bots of type social2 and social3. OCKRA shows a higher performance when detecting traditional bots. Moreover, Bagging-TPminer (Euclidean) has a performance above 0.89 AUC for all bot types. If the performance is examined per classifier instead of per type of bot type, it can be observed that Bagging-TPMiner (Euclidean) has the highest performance from all the one-class classifiers,

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
OCKRA	0.843	0.906	0.924	<b>0.916</b>	0.897	0.037
BTPM (E)	0.904	<b>0.952</b>	<b>0.934</b>	0.893	0.921	0.027
BTPM (M)	0.793	0.905	0.840	0.899	0.859	0.053
BRM (E)	0.724	0.807	0.791	0.892	0.803	0.069
BRM (M)	0.656	0.828	0.776	0.892	0.788	0.100
ocSVM	0.871	0.893	0.851	0.898	0.878	0.021
Naive Bayes	<b>0.995</b>	0.791	0.849	0.883	0.88	0.086
Isolation Forest	0.811	0.777	0.829	0.898	0.829	0.051
Mean	0.825	0.857	0.849	0.896		
Standard Deviation	0.105	0.065	0.056	0.009		

Table 5.6: Results of testing different types of bots when classifiers have used a training dataset containing legitimate users examples. A bold typeface is used to convey the highest performance attained by a classifier per testing dataset. BTPM and BRM where used with two different measures: Euclidean(E) and Mahalanobis(M)

with an average AUC of 0.921. Additionally, the standards deviations of Bagging-TPMiner (Euclidean) are lower than 0.1, which indicates this classifier is stable regardless of the type of bot to be detected.

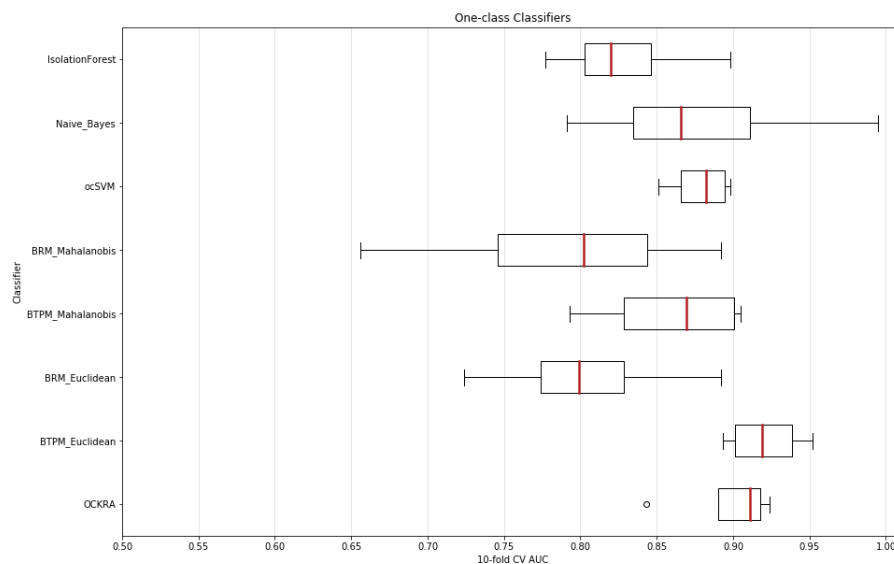


Figure 5.9: Performance results of the experiment set C.

As with the previous experiment sets, statistical tests were performed to observe if the bot type used in the testing set directly influence the performance of the classifiers. Figure

5.10 shows a CD diagram that compares the performances obtained for each bot type. The CD diagrams calculated for experiment set C show that there is no statistically significant difference among the bot types. Thus, the null hypothesis fails to be rejected.

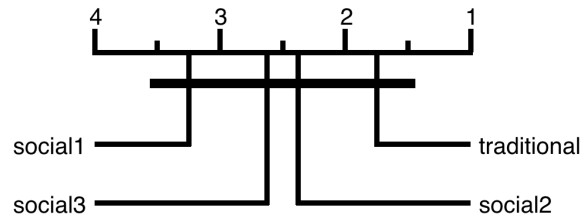


Figure 5.10: CD diagram of one-class classifiers’ AUC performance for Twitter bot detection.

## 5.4 Results Comparison

The experimental setup chosen for this work has the advantage of providing directly comparable results. This advantage is possible because the training and testing datasets are the same in all the experiments, the only difference is that, at the training phase, one-class classifiers ignore the examples of bot accounts. The following analysis was divided in two part to have a better understanding of the difference between one-class and binary classifier. The first part deals with such comparison, and will no include the performance of PBC4cip. On the second part analysis, PBC4cip is compared with the other four classifiers, and conclusions are made. In the first part, the best overall classifiers were chosen by comparing the two best-ranked classifiers of the multi-class and one-class approaches. Recall the criteria used for the classifier’s comparison is the performance in terms of AUC.

Table 5.7 contrasts the AUC performance of the two-best binary and one-class classifiers. In the first column of the table, the elements with bold typeface letters indicate that for the corresponding row, the training and testing datasets contain bots from the same type. Additionally, in the columns indicating the name of the classifier, for each row, the best AUC performance obtained by a given classifier has a bold typeface format. Table 5.7 gives several insights. First, we observe that, when the training set contains the same type of bot account examples as the testing set, the binary classifiers perform better than the others. Among the four compared classifiers, J48 obtained the best performance in 4 out of 16 cases, specifically in the experiments where the training and testing bots are of the same type. Nevertheless, in most of the cases where the training and testing datasets contain examples of different types of bots, one-class classifiers perform better. This happening is true for OCKRA and BPM, which obtained better performance than the others in 3 out of 16 cases, and in 6 out of 16 cases, respectively. In the three remaining cases, a binary classifier, Logistic Regression, performed better than one-class classifiers when classifying different types of bots. The performance of this binary classifier is superior to the others when trained with bots of type social3 and used

Bots in Train/Test	J48	LogReg	OCKRA	BTPM	PBC4cip
<b>s1/s1</b>	<b>0.977</b>	0.969	0.843	0.904	<b>0.996</b>
s1/s2	0.517	0.74	0.906	<b>0.952</b>	<b>0.974</b>
s1/s3	<b>0.957</b>	<b>0.957</b>	0.924	0.934	<b>0.995</b>
s1/t	0.733	0.588	<b>0.916</b>	0.893	<b>0.989</b>
s2/s1	0.68	0.511	0.843	<b>0.904</b>	<b>0.998</b>
<b>s2/s2</b>	<b>0.992</b>	0.987	0.906	0.952	<b>0.007</b>
s2/s3	0.596	0.907	0.924	<b>0.934</b>	<b>0.998</b>
s2/t	0.835	0.604	<b>0.916</b>	0.893	<b>0.998</b>
s3/s1	0.712	<b>0.971</b>	0.843	0.904	<b>0.994</b>
s3/s2	0.553	<b>0.984</b>	0.906	0.952	<b>0.997</b>
<b>s3/s3</b>	<b>0.97</b>	<b>0.97</b>	0.924	0.934	<b>0.996</b>
s3/t	0.774	0.741	<b>0.916</b>	0.893	<b>0.995</b>
t/s1	0.5	0.501	0.843	<b>0.904</b>	<b>0.979</b>
t/s2	0.502	0.502	0.906	<b>0.952</b>	<b>0.987</b>
t/s3	0.503	0.511	0.924	<b>0.934</b>	<b>0.995</b>
<b>t/t</b>	<b>1</b>	0.999	0.916	0.893	<b>1</b>
Mean	0.738	0.778	0.897	0.921	<b>0.993</b>
Standard Deviation	0.196	0.210	0.033	0.024	<b>0.007</b>

Table 5.7: Comparison of binary and one-class classifiers’ performance. In each row, bold typeface numbers are used to convey the highest AUC performance obtained for the given training/testing set combination. PBC4cip is the best-performing classifier, nevertheless, it is marked with gray bold typeface, as it is analyzed separately at first.

to discern between legitimate users and bots of types social1 and social2. Also, it is the best when using social1 type bots for the training phase, and used to discern between legitimate user accounts and bot accounts of type social3. As mentioned before, binary classifiers perform better in the scenario where the training set contains social1 bots, and the testing set contains social3 bots. This higher performance maybe because social1 and social3 have some related characteristics, which some classifiers can capture. Thus, examples of either type of bot could be used to detect the other one.

Regarding the average performance per classifier, for all cases, it can be observed in Table 5.7 that Bagging-TPMiner achieves the highest mean performance of all the classifiers, with an average AUC value of 0.921. The second-best performing method is OCKRA, with an average AUC value of 0.897. In general, the one-class classifiers perform at least 0.1 higher than the better performing binary classifiers. Furthermore, the standard deviations for

the one-class approach results are lower than 0.034, while for binary classifiers, the standard deviations are higher than 0.19. Therefore, the performance of one-class classifiers is more stable than the one exhibited by most binary classifiers.

The analysis so far has excluded PBC4cip's performance in order to gain a better understanding of the difference between most binary classifiers and one-class methods. As shown by table 5.7, the performance of PBC4cip exceeds the ones yielded by other classifiers, both in terms of mean AUC and standard deviation. These results show that the contrast pattern-based algorithm is more stable than one class classifiers, and also does not suffer from lack of generalization most binary classifiers have.

### 5.4.1 Statistical Differences Among Approaches

Figure 5.11 shows a CD diagram that compares the performance that the two best-performing binary and two best-performing one class classifiers. This CD diagram confirms that PBC4cip is the best performing algorithm, as it is ranked very close to one. Moreover, the performance of PBC4cip has a statistically significant difference compared with the other classifiers. This contrast pater-based algorithm is more stable than one class classifiers, and also does not suffer from lack of generalization most binary classifiers have.

As for the other 3 classifiers, BTM is the highest ranked, followed by OCKRA and Logistic Regression. Regardless of the different positions in the diagram, the statistical tests show that are no statistically significant difference between the performance they obtained. This does not should obscure the fact that each classifier has different properties that make their more suitable for particular situations.

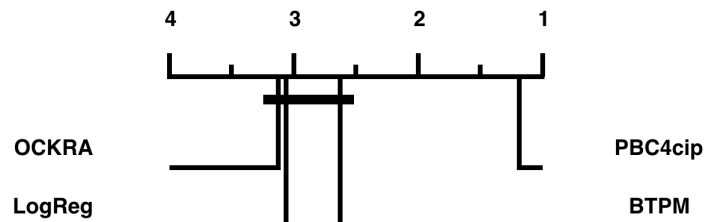


Figure 5.11: CD diagram of the two best multi-class and one-class classifiers used in the Twitter bot detection experiments.

### 5.4.2 Generalization Capabilities

When comparing the performance results of multi-class classifiers shown in Table 5.5 with the results of the one-class classifiers presented in Table 5.6, it can be observed that, unlike most multi-class classifiers, the performance of one-class classifiers is more stable. The values of the standard deviations show this stability, which means that one-class classifiers have similar performance when classifying all types of bots. In contrast, with a multi-class approach, there

is at least one type of bot where each classifier has considerably lower performance than the others. Moreover, the standard deviations of most one-class classifiers are lower in one order of magnitude than those of most multi-class classifiers. The notable exception is J48, from the multi-class classifiers, which has a 0.012 standard deviation. However, its performance of 0.886 average AUC, is lower than that of Bagging-TPMiner, which is 0.921 average AUC, with a standard deviation of 0.027. The results obtained in the second set of experiments show that multi-class classifiers have an acceptable performance when the types of bots used in the training set express similar behavior to the bots used in the testing phase. This adequate performance is due to some limited generalization capabilities. However, binary and multi-class methods also have common disadvantages; for bots that exhibit different behavior than the ones used to train, the performance depends on the classifier, and generally, the rate of false negatives increases. On the other side, one-class classifiers have lower performance than specialized multi-class classifiers. Nevertheless, one-class methods can correctly discern between legitimate and bot accounts without requiring any example of what constitutes a bot. Also, these algorithms have more stable behavior, as shown by the lower standard deviations. PBC4cip, which makes use of contrast patterns, is resilient to the difference of bot types used in the test set; its performance is excellent in every one of the experiments, and it shows a statistically significant difference with respect to the other binary and one-class classifiers. Further, PBC4cip has proven to behave similarly regardless of the type of bot and number of types of bots used in the training set, and robust when tested against bot type it has not previously encountered.

## 5.5 Summary

Three experiments were carried out to analyze the difference and similarities in the performances of binary, multi-class, and one-class classifiers. As explained in Section 4.1, the experimental setup allows for a direct comparison of the results. From figure 5.6, it can be observed that binary classifiers do not have a statistical difference among their AUC performance. A recurrent pattern found when comparing the performance of both methods is that binary classifiers achieved the highest performance when detecting the type of bot that was used in the training phase. This pattern also holds when multiple types of bots are used in the training set. However, binary and multi-class methods also have common disadvantages; for bots that exhibit different behavior than the one used to train, the performance depends on the classifier, leading to an increase in the false-negative rate for most cases. As for the one-class classifiers, they perform a little lower than specialized multi-class classifiers. Nevertheless, one-class methods can correctly discern between legitimate and bot accounts without requiring any example of what constitutes a bot. Moreover, the one-class methods tend to have more stable behavior, as shown by their lower standard deviations. PBC4cip has performed almost excellent in every different experiment set, which was reflected in the CD diagrams and standard deviations in the tables. Thus, suggesting that contrast pattern-based classifiers might outperform other methods in the bot detection context.



# Chapter 6

## Conclusions

In the last decade, online social networks have increased in popularity. By the end of 2018, the total number of accounts registered in these networks was approximately 2.65 billion [108]. This vast amount of users imply that an essential part of the world's population is continuously exposed to the content posted in such networks. There exists a relevant information flow in social networks, although most of the participants in social platforms do not know each other personally. Moreover, researches have shown that there exist central nodes (accounts) from which the information disseminates [85, 3]. Unfortunately, in online social networks such as Twitter, there are automated accounts, known as bots, that have been used for malicious purposes. The misuses of bots go from distributing spam and malware, to attempting to influence people's opinion during political campaigns.

Different Twitter bot detection methods [82, 129, 37, 56, 31] have been developed to uncover malicious bots and stop them from causing harm in society. Bot detection methods are based on the premise that humans and bot accounts behave differently. The behavior of a given account can be modeled using a series of selected features that give information about account use. These features go from the number of followers and friends to the ratio between unique hashtag count and tweet count, for example. In the Twitter bot detection literature, the features used to characterize accounts vary greatly across authors. As for the measures selected used to evaluate the performance of each method, most authors report the performance achieved by their proposals using Accuracy, F1, or AUC. Although they make use of the same measures, those methods are still not directly comparable due to differences in their methodology. Nevertheless, there exists common ground among bot detection methods; most of them are based on supervised machine learning approaches [111] and have proven to be highly effective when detecting bot types that are similar to those used to train their algorithms.

Worryingly, malicious bots have changed their behavior over time to bypass existing detection mechanisms. Bot content generation [51] and behavior emulation [64] have gotten more sophisticated with the use of artificial intelligence tools. Also, coordinated accounts of bots have been devised to make less visible the intentions of the individual accounts. Experiments have shown that when the types of bots trying to be detected are different from those used in the training phase of the algorithms, state-of-the-art classifiers' performance decreases [129].

The results of the experiments designed in this thesis are consistent with those in the

bot detection literature. These results show that Twitter bots can be correctly detected using a supervised classification approach. The highest performance obtained, when discerning between bots and legitimate users, exceeds 0.95 AUC. Nevertheless, we have shown that binary classifiers yield a low performance when predicting the class of a bot account whose type is different from the ones that were used to train the classifier. The highest losses in performance are close to 0.4 average AUC.

On the other hand, multi-class methods have shown a more stable performance than binary classifiers. Also, they have a higher mean AUC performance. Even though multi-classifiers are trained with examples of more than one type of bots, they cannot discern between bot and legitimate accounts in all experiments. To overcome this limitation, we propose to approach Twitter bot detection as an anomaly detection problem. This approach involves the use of one-class classifiers, which are methods that only require examples of one class to learn a classification model.

Our experiment results show that using only genuine users examples, the best performing classifiers, for each type of bot, achieved a score above 0.91 average AUC. Moreover, the difference in performance among the three types of algorithms was statistically significant. Therefore the hypothesis that one-class classifiers can overcome the limitations of currently existing methods can be accepted. Nevertheless, it is not the only way to do it. A contrast pattern-based classifier, PBC4cip, excelled in every experiment set, yielding outputs close to 1 AUC. These findings support previous research of pattern-based classifiers made in the bot detection context [82]. The results of the experiments designed for this thesis further provide evidence that these methods are reliable and robust in the Twitter bot detection context, at an account level. An additional advantage of one-class methods relies on the proposed feature vector being approximately 92 times smaller than the one used in the so-far best-performing methods. This difference in size translates to a simpler and faster retrieval of the proposed vector.

Given the results obtained in the first and second experiment sets and the continuous change in bot behavior, it is reasonable to think that nowadays, there are new types of bots navigating Twitter. These bots exhibit behaviors different from those seen previously and, therefore, currently avoid automated detection methods. The existence of such undercover bots represents a threat for social network users, as it can result in successful amplification of harming messages, malware dissemination campaigns that extend for relatively long periods, or in any other malicious activity.

On the other hand, the performance shown by one-class classifiers and PBC4cip, suggest that its underlying mechanism could be used to detect anomalous patterns in the behavior of Twitter. After a careful analysis, we can conclude that one-class classifiers can complement existing approaches, helping with the detection of new bots, while state-of-the-art classifiers are better when focused on previously known bot types. Thus, a more robust automatic bot detection system could be constructed by using binary classifiers together with one-class methods. Such a system could discern correctly known bots from legitimate accounts, and also be able to find possible automated accounts with more certainty. An important consideration is that a successful implementation of an automated bot detection system by using one-class and binary classifiers would need to include updating mechanisms that take into account changes in the legitimate behaviors, such as those that result by changes in the social platform. On the other hand, contrast pattern-based classifiers are powerful detection methods with great

explanatory capacity. Having the possibility to extract the bot behavior patterns can bring useful insight for OSN's administrators, which can modify the rule to reduce the misuse of these popular platforms.

## 6.1 Future Work

The work in the bot detection domain is far from being over. This present work has proven the effectiveness of a novel approach for detecting bots; nevertheless, there are elements of this approach that could be optimized. One example of these elements is the feature vector used to characterize the accounts' behavior. Even though the proposed vector was carefully designed by studying the findings in previous studies, it could be refined to provide a better characterization by extensively testing different feature combinations. Additionally, more contrast pattern-based methods can be compared and combined to further push the upper bound in bot detection performance.

### 6.1.1 Hybrid approach

The approach proposed in this thesis does not intend to replace existing bot detection methods. Instead, they are intended to complement existing methods. We encourage the development of systems that combine both multi-class and one-class methods into a single detection mechanism. Such a hybrid approach could be robust when detecting known bot types, as existing methods are. Additionally, these systems would benefit from the one-class approach by having the capacity of detecting novel bot types more effectively than traditional methods can. The development of hybrid approaches is an open area of research. To the best of the author's knowledge, there are still no efforts done in the direction of mixed detection systems. Botnet detection is an open area of research, as current methods for botnet detection have the limitation of relying on checking only a single feature. This limitation may allow for different types of botnets to pass undetected. Nonetheless, a first step to solve this problem may be taken by designing hybrid detection methods, or ensembles of distinct types of classifiers, that evaluate different aspects of the accounts' behavior at the same time.

### 6.1.2 Proactive approaches

Most bot detection mechanisms have arisen from the need to stop active malicious accounts, but a recent study has proposed a proactive approach for bot detection [39]. Adopting a proactive approach means to devise manners in which bots could evade actual detection systems, to then come up with methods that could detect these more *advanced* bots. Moreover, the experiments presented in [39] focus on *evolving* only the posting patterns of the bots. A more general proactive approach could deal not only with posting patterns but also with the type of content posted and even to the manners in which coordinated groups of malicious bots behave. This anticipation strategy has the potential of uncovering *advanced* bots that currently infiltrate social networks. A proactive mindset by no way implies this persecution is soon to be over, but instead, it is just another step towards developing more robust systems that eventually will be outsmarted by more sophisticated bots. As an educated guess, it is very likely

that this arms race continue as long as online social networks exist, and that the advancement of the malicious bot side still be strongly influenced by the extent to which the administrators of these networks allow automation.

# Bibliography

- [1] ABOKHODAIR, N., YOO, D., AND McDONALD, D. W. Dissecting a social botnet. *Proceedings of the 18th Association for Computing Machinery Conference on Computer Supported Cooperative Work Social Computing - CSCW '15* (2015).
- [2] ACKERMANN, M. R., MÄRTENS, M., RAUPACH, C., SWIERKOT, K., LAMMERSEN, C., AND SOHLER, C. Streamkm++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics* 17 (2012), 2.4:2.1–2.4:2.30.
- [3] ADALI, S., ESCRIVA, R., GOLDBERG, M. K., HAYVANOVYCH, M., MAGDON-ISMAIL, M., SZYMANSKI, B. K., WALLACE, W. A., AND WILLIAMS, G. Measuring behavioral trust in social networks. In *2010 IEEE International Conference on Intelligence and Security Informatics* (2010), pp. 150–152.
- [4] AHMED, F., AND ABULAISH, M. A generic statistical approach for spam detection in online social networks. *Computer Communications* 36, 10 (2013), 1120 – 1129.
- [5] AIELLO, L. M., DEPLANO, M., SCHIFANELLA, R., AND RUFFO, G. People are strange when you're a stranger: Impact and influence of bots on social networks. vol. abs/1407.8134.
- [6] ALIZA ROSEN. Tweeting made easier, 2017. Blog. [https://blog.twitter.com/official/en\\_us/topics/product/2017/tweetingmadeeasier.html](https://blog.twitter.com/official/en_us/topics/product/2017/tweetingmadeeasier.html), Last accessed on 2019-08-15.
- [7] ALLEM, J.-P., AND FERRARA, E. The importance of debiasing social media data to better understand e-cigarette-related attitudes and behaviors. *Journal of Medical Internet Research* 18, 8 (2016), e219.
- [8] ALLEM, J.-P., AND FERRARA, E. Could social bots pose a threat to public health? *American Journal of Public Health* 108, 8 (2018), 1005–1006. PMID: 29995482.
- [9] ALLEM, J.-P., FERRARA, E., UPPU, S. P., CRUZ, T. B., AND UNGER, J. B. E-cigarette surveillance with social media data: Social bots, emerging topics, and trends. *Journal of Medical Internet Research Public Health Surveill* 3, 4 (2017), e98.
- [10] ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.

- [11] AMANCIO, D. R., COMIN, C. H., CASANOVA, D., TRAVIESO, G., BRUNO, O. M., RODRIGUES, F. A., AND DA FONTOURA COSTA, L. A systematic comparison of supervised classifiers. *Public Library of Science ONE* 9, 4 (2014), 1–14.
- [12] ARLOT, S., AND CELISSE, A. A survey of cross-validation procedures for model selection. *Statistics Surveys* 4 (2010), 40–79.
- [13] ARTHUR, D., AND VASSILVITSKII, S. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 2007), SODA '07, Society for Industrial and Applied Mathematics, pp. 1027–1035.
- [14] BARTKOWIAK, A. Anomaly, novelty, one-class classification: A short introduction. *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)* (2010), 1–6.
- [15] BATISTA, G. E. A. P. A., PRATI, R. C., AND MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. *Association for Computing Machinery. Special Interest Group on Knowledge Discovery and Data Mining. Explorations Newsletter* 6, 1 (2004), 20–29.
- [16] BEKKAR, M., DJEMA, H., AND ALITOCHE, T. Evaluation measures for models assessment over imbalanced data sets. *Journal of Information Engineering and Applications* 3 (2013), 27–38.
- [17] BEN-GAL, I. *Bayesian Networks*. American Cancer Society, 2008.
- [18] BERGER, J. M., AND MORGAN, J. The isis twitter census: defining and describing the population of isis supporters on twitter.
- [19] BERRAR, D. Cross-validation. In *Encyclopedia of Bioinformatics and Computational Biology*, S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach, Eds. Academic Press, Oxford, 2019, pp. 542 – 545.
- [20] BESSI, A., COLETTI, M., DAVIDESCU, G. A., SCALA, A., CALDARELLI, G., AND QUATTROCIOCCI, W. Science vs conspiracy: Collective narratives in the age of misinformation. *Public Library of Science ONE* 10, 2 (2015), 1–17.
- [21] BESSI, A., AND FERRARA, E. Social bots distort the 2016 u.s. presidential election online discussion. *First Monday* 21 (2016), 1–14.
- [22] BISHOP, C. M. Novelty detection and neural network validation. *Institution of Electrical Engineers - Vision, Image and Signal Processing* 141, 4 (1994), 217–222.
- [23] BOSER, B. E., GUYON, I. M., AND VAPNIK, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (New York, NY, USA, 1992), Computational Learning Theory '92, Association for Computing Machinery, pp. 144–152.

- [24] BOUGHORBEL, S., JARRAY, F., AND EL-ANBARI, M. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *Public Library of Science ONE* 12, 6 (2017), 1–17.
- [25] BREIMAN, L. Bagging predictors. *Machine Learning* 24, 2 (1996), 123–140.
- [26] BREIMAN, L. Random forests. *Machine Learning* 45, 1 (2001), 5–32.
- [27] BRONIATOWSKI, D., M JAMISON, A., QI, S., ALKULAIB, L., CHEN, T., BENTON, A., QUINN, S., AND DREDZE, M. Weaponized health communication: Twitter bots and russian trolls amplify the vaccine debate. *American journal of public health* 108 (2018), e1–e7.
- [28] BURGESS, C. J. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2, 2 (1998), 121–167.
- [29] CAMIÑA, J. B., MEDINA-PÉREZ, M. A., MONROY, R., LOYOLA-GONZÁLEZ, O., VILLANUEVA, L. A., AND GURROLA, L. C. Bagging-randomminer: A one-class classifier for file access-based masquerade detection. *Machine Vision Applications* 30, 5 (2019), 959–974.
- [30] CAMPBELL, C., AND YING, Y. *Learning with Support Vector Machines*. Morgan & Claypool Publishers, 2011.
- [31] CHAVOSHI, N., HAMOONI, H., AND MUEEN, A. Identifying correlated bots in twitter. In *Social Informatics* (Cham, 2016), E. Spiro and Y.-Y. Ahn, Eds., Springer International Publishing, pp. 14–21.
- [32] CHEN, Z., AND SUBRAMANIAN, D. An unsupervised approach to detect spam campaigns that use botnets on twitter. *Computing Research Repository abs/1804.05232* (2018).
- [33] CHU, Z., GIANVECCHIO, S., WANG, H., AND JAJODIA, S. Who is tweeting on twitter: Human, bot, or cyborg? In *Proceedings of the 26th Annual Computer Security Applications Conference* (New York, NY, USA, 2010), ACSAC '10, Association for Computing Machinery, pp. 21–30.
- [34] CHU, Z., GIANVECCHIO, S., WANG, H., AND JAJODIA, S. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *Institute of Electrical and Electronics Engineers Transactions on Dependable and Secure Computing* 9, 6 (2012), 811–824.
- [35] CLARK, E. M., JONES, C. A., WILLIAMS, J. R., KURTI, A. N., NOROTSKY, M. C., DANFORTH, C. M., AND DODDS, P. S. Vaporous marketing: Uncovering pervasive electronic cigarette advertisements on twitter. *Public Library of Science ONE* 11, 7 (2016), 1–14.
- [36] CRESCI, S., DI PIETRO, R., PETROCCHI, M., SPOGNARDI, A., AND TESCONI, M. Fame for sale: Efficient detection of fake twitter followers. *Decision Support Systems* 80 (2015), 56–71.

- [37] CRESCI, S., DI PIETRO, R., PETROCCHI, M., SPOGNARDI, A., AND TESCONI, M. Dna-inspired online behavioral modeling and its application to spambot detection. *Institute of Electrical and Electronics Engineers Intelligent Systems* 31, 5 (2016), 58–64.
- [38] CRESCI, S., DI PIETRO, R., PETROCCHI, M., SPOGNARDI, A., AND TESCONI, M. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion* (Republic and Canton of Geneva, Switzerland, 2017), WWW '17 Companion, International World Wide Web Conferences Steering Committee, pp. 963–972.
- [39] CRESCI, S., PETROCCHI, M., SPOGNARDI, A., AND TOGNAZZI, S. Better safe than sorry: An adversarial approach to improve social bot detection. 47–56.
- [40] DAVIS, C. A., VAROL, O., FERRARA, E., FLAMMINI, A., AND MENCZER, F. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web* (2016), pp. 273–274.
- [41] DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7 (2006), 1–30.
- [42] DI MARTINO, M., DECIA, F., MOLINELLI, J., AND FERNÁNDEZ, A. Improving electric fraud detection using class imbalance strategies. In *ICPRAM (2)* (2012), pp. 135–141.
- [43] DICKERSON, J. P., KAGAN, V., AND SUBRAHMANIAN, V. S. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *Proceedings of the 2014 Institute of Electrical and Electronics Engineers/Association for Computing Machinery International Conference on Advances in Social Networks Analysis and Mining* (Piscataway, NJ, USA, 2014), ASONAM '14, Institute of Electrical and Electronics Engineers Press, pp. 620–627.
- [44] DONG, G., AND BAILEY, J. *Contrast Data Mining: Concepts, Algorithms, and Applications*, 1st ed. Chapman & Hall/CRC, 2012.
- [45] FAWCETT, T. An introduction to roc analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.
- [46] FAWCETT, T., AND PROVOST, F. Activity monitoring: Noticing interesting changes in behavior. 53–62.
- [47] FERRARA, E. Disinformation and social bot operations in the run up to the 2017 french presidential election. *Computing Research Repository abs/1707.00086* (2017).
- [48] FERRARA, E., VAROL, O., DAVIS, C., MENCZER, F., AND FLAMMINI, A. The rise of social bots. *Communications of the Association for Computing Machinery* 59, 7 (2016), 96–104.
- [49] FERRARA, E., WANG, W.-Q., VAROL, O., FLAMMINI, A., AND GALSTYAN, A. Predicting online extremism, content adopters, and interaction reciprocity. *Social Informatics* (2016), 22–39.



- [50] FOODY, G. M., AND ARORA, M. K. An evaluation of some factors affecting the accuracy of classification by an artificial neural network. *International Journal of Remote Sensing* 18, 4 (1997), 799–810.
- [51] FREITAS, C., BENEVENUTO, F., GHOSH, S., AND VELOSO, A. Reverse engineering socialbot infiltration strategies in twitter. 25–32.
- [52] FREUND, Y., AND SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55, 1 (1997), 119 – 139.
- [53] GARCIA, S., AND HERRERA, F. An extension on ”statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. *Journal of Machine Learning Research* 9 (2008), 2677–2694.
- [54] GEISSER, S. The predictive sample reuse method with applications. *Journal of the American Statistical Association* 70, 350 (1975), 320–328.
- [55] GHOSH, S., VISWANATH, B., KOOTI, F., SHARMA, N. K., KORLAM, G., BENEVENUTO, F., GANGULY, N., AND GUMMADI, K. P. Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st International Conference on World Wide Web* (New York, NY, USA, 2012), WWW ’12, ACM, pp. 61–70.
- [56] GILANI, Z., KOCHMAR, E., AND CROWCROFT, J. Classification of twitter accounts into automated agents and human users. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017* (New York, NY, USA, 2017), ASONAM ’17, ACM, pp. 489–496.
- [57] GRIMME, C., ASSENMACHER, D., AND ADAM, L. Changing perspectives: Is it sufficient to detect social bots? In *Social Computing and Social Media. User Experience and Behavior* (Cham, 2018), G. Meiselwitz, Ed., Springer International Publishing, pp. 445–461.
- [58] GUPTA, A., LAMBA, H., AND KUMARAGURU, P. \$1.00 per rt bostonmarathon pray-forboston: Analyzing fake content on twitter. pp. 1–12.
- [59] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The weka data mining software: An update. *Special Interest Group on Knowledge Discovery and Data Mining Explorations Newsletter* 11, 1 (2009), 10–18.
- [60] HAUSTEIN, S., BOWMAN, T. D., HOLMBERG, K., TSOU, A., SUGIMOTO, C. R., AND LARIVIÈRE, V. Tweets as impact indicators: Examining the implications of automated ”bot” accounts on twitter. *Journal of the Association for Information Science and Technology* 67, 1 (2016), 232–238.
- [61] HINTON, G. E. Connectionist learning procedures. *Artificial Intelligence* 40, 1 (1989), 185 – 234.

- [62] HO, T. K. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1* (Washington, DC, USA, 1995), ICDAR '95, IEEE Computer Society, pp. 278–282.
- [63] HODGE, V. J., AND AUSTIN, J. A survey of outlier detection methodologies. *Artificial Intelligence Review* 22, 2 (2004), 85–126.
- [64] HWANG, T., PEARCE, I., AND NANIS, M. Socialbots: Voices from the fronts. *Interactions* 19, 2 (2012), 38–45.
- [65] INUWA-DUTSE, I., LIPTROTT, M., AND KORKONTZELOS, I. Detection of spam-posting accounts on twitter. *Neurocomputing* 315 (2018), 496 – 511.
- [66] JAPKOWICZ, N., MYERS, C., AND GLUCK, M. A novelty detection approach to classification. 518–523.
- [67] JHA, S., TAN, K., AND MAXION, R. A. Markov chains, classifiers, and intrusion detection. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations* (Washington, DC, USA, 2001), Computer Security Foundations Workshop '01, Institute of Electrical and Electronics Engineers Computer Society, pp. 206–.
- [68] JIN HUANG, AND LING, C. X. Using auc and accuracy in evaluating learning algorithms. *Institute of Electrical and Electronics Engineers Transactions on Knowledge and Data Engineering* 17, 3 (2005), 299–310.
- [69] JUN, Y., MENG, R., AND JOHAR, G. V. Perceived social presence reduces fact-checking. *Proceedings of the National Academy of Sciences* 114, 23 (2017), 5976–5981.
- [70] KAPLAN, A. M., AND HAENLEIN, M. The early bird catches the news: Nine things you should know about micro-blogging. *Business Horizons* 54, 2 (2011), 105 – 113.
- [71] KAUR, G., AND CHHABRA, A. Improved j48 classification algorithm for the prediction of diabetes. *International Journal of Computer Applications* 98, 22 (2014), 13–17.
- [72] KELLER, F., SCHOCH, D., STIER, S., AND YANG, J. How to manipulate social media: Analyzing political astroturfing using ground truth data from south korea, 2017.
- [73] KOTSIANTIS, S. B. Supervised machine learning: A review of classification techniques. 3–24.
- [74] KWAK, H., LEE, C., PARK, H., AND MOON, S. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web* (New York, NY, USA, 2010), WWW '10, Association for Computing Machinery, pp. 591–600.
- [75] LARSON, S. C. The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology* 22, 1 (1931), 45–55.

- [76] LEE, K., CAVERLEE, J., AND WEBB, S. Uncovering social spammers: Social honeypots + machine learning. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2010), SIGIR '10, ACM, pp. 435–442.
- [77] LEE, K., EOFF, B. D., AND CAVERLEE, J. Seven months with the devils: a long-term study of content polluters on twitter. In *In Association for the Advancement of Artificial Intelligence Int'l Conference on Weblogs and Social Media (ICWSM)* (2011).
- [78] LIPTON, Z. C., ELKAN, C., AND NARYANASWAMY, B. Optimal thresholding of classifiers to maximize f1 measure. In *Machine Learning and Knowledge Discovery in Databases* (Berlin, Heidelberg, 2014), T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, Eds., Springer Berlin Heidelberg, pp. 225–239.
- [79] LIU, F. T., TING, K. M., AND ZHOU, Z.-H. Isolation forest. In *Proceedings of the 2008 Eighth International Conference on Data Mining* (Washington, DC, USA, 2008), International Conference on Data Mining '08, Institute of Electrical and Electronics Engineers Computer Society, pp. 413–422.
- [80] LOKOT, T., AND DIAKOPOULOS, N. News bots. *Digital Journalism* 4, 6 (2016), 682–699.
- [81] LONGLEY, P., CHESHIRE, J., AND SINGLETON, A. *Consumer data research*. UCL Press, 2018.
- [82] LOYOLA-GONZÁLEZ, O., MONROY, R., RODRÍGUEZ, J., LÓPEZ-CUEVAS, A., AND MATA-SÁNCHEZ, J. I. Contrast pattern-based classification for bot detection on twitter. *Institute of Electrical and Electronics Engineers Access* 7 (2019), 45800–45817.
- [83] LOYOLA-GONZÁLEZ, O., MEDINA-PÉREZ, M., MARTÍNEZ-TRINIDAD, J. F., CARRASCO-OCHOA, J., MONROY, R., AND GARCÍA-BORROTO, M. Pbc4cip: A new contrast pattern-based classifier for class imbalance problems. *Knowledge-Based Systems* 115 (2017), 100–109.
- [84] M. J. LAZER, D., BAUM, M., BENKLER, Y., J. BERINSKY, A., M. GREENHILL, K., MENCZER, F., J. METZGER, M., NYHAN, B., PENNYCOOK, G., ROTHSCHILD, D., SCHUDSON, M., SLOMAN, S., SUNSTEIN, C., A. THORSON, E., J. WATTS, D., AND L. ZITTRAIN, J. The science of fake news. *Science* 359 (2018), 1094–1096.
- [85] MAHYAR, H., HASHEMINEZHAD, R., GHALEBI, E., NAZEMIAN, A., GROSU, R., MOVAGHAR, A., AND RABIEE, H. R. Identifying central nodes for information flow in social networks using compressive sensing. *Social Network Analysis and Mining* 8, 1 (2018), 33.
- [86] MARTINEZ-ROMO, J., AND ARAUJO, L. Detecting malicious tweets in trending topics using a statistical analysis of language. *Expert Systems with Applications* 40, 8 (2013), 2992–3000.

- [87] MEDINA-PÉREZ, M. A., MONROY, R., CAMIÑA, J. B., AND GARCÍA-BORROTO, M. Bagging-tpminer: a classifier ensemble for masquerader detection based on typical objects. *Soft Computing* 21, 3 (2017), 557–569.
- [88] MILLER, Z., DICKINSON, B., DEITRICK, W., HU, W., AND WANG, A. H. Twitter spammer detection using data stream clustering. *Information Sciences* 260 (2014), 64 – 73.
- [89] MOYA, M. M., KOCH, M. W., AND HOSTETLER, L. D. One-class classifier networks for target recognition applications. 24–43.
- [90] NAMIOT, D. Twitter as a transport layer platform. In *2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT)* (2015), Institute of Electrical and Electronics Engineers, pp. 46–51.
- [91] NISBET, R., MINER, G., AND YALE, K. Chapter 11 - model evaluation and enhancement. In *Handbook of Statistical Analysis and Data Mining Applications (Second Edition)*, R. Nisbet, G. Miner, and K. Yale, Eds., second edition ed. Academic Press, Boston, 2018, pp. 215 – 233.
- [92] OLIVERI, P. Class-modelling in food analytical chemistry: Development, sampling, optimisation and validation issues – a tutorial. *Analytica Chimica Acta* 982 (2017), 9 – 19.
- [93] OPITZ, D., AND MACLIN, R. Popular ensemble methods: An empirical study. *Journal Artificial Intelligence Research* 11, 1 (1999), 169–198.
- [94] POLIKAR, R. Ensemble based systems in decision making. *Institute of Electrical and Electronics Engineers Circuits and Systems Magazine* 6, 3 (2006), 21–45.
- [95] PROVOST, F., AND FAWCETT, T. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (1997), Knowledge Discovery and Data Mining '97, Association for the Advancement of Artificial Intelligence Press, pp. 43–48.
- [96] QUINLAN, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [97] RIPLEY, B. D., AND HJORT, N. L. *Pattern Recognition and Neural Networks*, 1st ed. Cambridge University Press, New York, NY, USA, 1995.
- [98] RITTER, G., AND GALLEGOS, M. T. Outliers in statistical pattern recognition and an application to automatic chromosome classification. *Pattern Recognition Letters* 18, 6 (1997), 525–539.
- [99] RODRÍGUEZ, J., BARRERA-ANIMAS, A., TREJO, L., MEDINA-PÉREZ, M., AND MONROY, R. Ensemble of one-class classifiers for personal risk detection based on wearable sensor data. *Sensors* 16, 10 (2016), 1619.

- [100] RODRÍGUEZ-RUIZ, J., MATA-SÁNCHEZ, J. I., MONROY, R., LOYOLA-GONZÁLEZ, O., AND LÓPEZ-CUEVAS, A. A one-class classification approach for bot detection on twitter. Submitted to *Computers & Security*. Under review.
- [101] RODRÍGUEZ-RUIZ, J., MONROY, R., MEDINA-PÉREZ, M. A., LOYOLA-GONZÁLEZ, O., AND CERVANTES, B. Cluster validation in clustering-based one-class classification. *Expert Systems*, e12475. e12475 10.1111/exsy.12475.
- [102] ROKACH, L. Ensemble-based classifiers. *Artificial Intelligence Review* 33, 1-2 (2010), 1–39.
- [103] ROUSSEEUW, P. J., AND LEROY, A. M. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., New York, NY, USA, 1987.
- [104] RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. Neurocomputing: Foundations of research. Massachusetts Institute of Technology Press, Cambridge, MA, USA, 1988, ch. Learning Representations by Back-propagating Errors, pp. 696–699.
- [105] SHAO, C., HUI, P.-M., WANG, L., JIANG, X., FLAMMINI, A., MENCZER, F., AND CIAMPAGLIA, G. L. Anatomy of an online misinformation network. *Public Library of Science ONE* 13, 4 (2018), e0196087.
- [106] SOKOLOVA, M., JAPKOWICZ, N., AND SZPAKOWICZ, S. Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. In *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence* (Berlin, Heidelberg, 2006), AI'06, Springer-Verlag, pp. 1015–1021.
- [107] SOKOLOVA, M., AND LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing and Management* 45, 4 (2009), 427–437.
- [108] STATISTA. Number of social network users worldwide from 2010 to 2021 (in billions). (2019) <https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/>, Last accessed on 2019 08 15.
- [109] STELLA, M., FERRARA, E., AND DE DOMENICO, M. Bots increase exposure to negative and inflammatory content in online social systems. *Proceedings of the National Academy of Sciences* 115, 49 (2018), 12435–12440.
- [110] STRINGHINI, G., KRUEGEL, C., AND VIGNA, G. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference* (New York, NY, USA, 2010), ACSAC '10, Association for Computing Machinery, pp. 1–9.
- [111] SUBRAHMANIAN, V. S., AZARIA, A., DURST, S., KAGAN, V., GALSTYAN, A., LERMAN, K., ZHU, L., FERRARA, E., FLAMMINI, A., AND MENCZER, F. The darpa twitter bot challenge. *Computer* 49, 6 (2016), 38–46.

- [112] SUGIMOTO, C. R., WORK, S., LARIVIÈRE, V., AND HAUSTEIN, S. Scholarly use of social media and altmetrics: A review of the literature. *Journal of the Association for Information Science and Technology* 68, 9 (2017), 2037–2062.
- [113] TAX, D. *One-class classification: concept-learning in the absence of counter-examples*. PhD thesis, The Netherlands: University of Delft, 2001.
- [114] TAX, D. M. J. *One-class classification: Concept learning in the absence of counter-examples*. PhD thesis, Technische Universiteit Delft, 2001.
- [115] TWITTER, INC. Getting Started. Help Center. (2017) <https://help.twitter.com/en/twitter-guide>, Last accessed on 2019 08 15.
- [116] TWITTER, INC. Financial Releases. Q2 2019 Letter to Shareholders. Investor Relations. (2019).
- [117] TWITTER, INC. About Twitter’s APIs. Help Center. (2019) <https://help.twitter.com/en/rules-and-policies/twitter-api>, Last accessed on 2019 08 15.
- [118] TWITTER, INC. Automation rules. Help Center. (2017) <https://help.twitter.com/en/rules-and-policies/twitter-automation>, Last accessed on 2019 08 15.
- [119] VAN DONGEN, S. Graph clustering via a discrete uncoupling process. *Society for Industrial and Applied Mathematics Journal on Matrix Analysis and Applications* 30, 1 (2008), 121–141.
- [120] VAPNIK, V. N. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg, 1995.
- [121] VAROL, O., FERRARA, E., DAVIS, C. A., MENCZER, F., AND FLAMMINI, A. Online human-bot interactions: Detection, estimation, and characterization. In *International AAAI Conference on Web and Social Media* (2017), Association for the Advancement of Artificial Intelligence, Association for the Advancement of Artificial Intelligence, p. 280–289.
- [122] WALD, R., KHOSHGOFTAAR, T. M., NAPOLITANO, A., AND SUMNER, C. Predicting susceptibility to social bots on twitter. In *2013 IEEE 14th International Conference on Information Reuse Integration (IRI)* (2013), pp. 6–13.
- [123] WAN, L., NG, W. K., DANG, X. H., YU, P. S., AND ZHANG, K. Density-based clustering of data streams at multiple resolutions. *Association for Computing Machinery Transactions on Knowledge Discovery from Data* 3, 3 (2009), 14:1–14:28.
- [124] WANG, A. H. Detecting spam bots in online social networking sites: A machine learning approach. In *Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy* (Berlin, Heidelberg, 2010), DBSec’10, Springer-Verlag, pp. 335–342.

- [125] WANG, B., ZUBIAGA, A., LIAKATA, M., AND PROCTER, R. Making the most of tweet-inherent features for social spam detection on twitter. *Computing Research Repository abs/1503.07405* (2015).
- [126] WANG, G., MOHANLAL, M., WILSON, C., WANG, X., METZGER, M. J., ZHENG, H., AND ZHAO, B. Y. Social turing tests: Crowdsourcing sybil detection. *Computing Research Repository abs/1205.3856* (2012).
- [127] WANG, K., AND STOLFO, S. J. One-class training for masquerade detection. In *3rd IEEE Conference Data Mining Workshop on Data Mining for Computer Security* (2003).
- [128] YANG, C., HARKREADER, R., AND GU, G. Empirical evaluation and new design for fighting evolving twitter spammers. *Institute of Electrical and Electronics Engineers Transactions on Information Forensics and Security* 8, 8 (2013), 1280–1293.
- [129] YANG, K.-C., VAROL, O., DAVIS, C. A., FERRARA, E., FLAMMINI, A., AND MENCZER, F. Arming the public with artificial intelligence to counter social bots. *Human Behavior and Emerging Technologies* 1, 1 (2019), 48–61.
- [130] YU, H.-F., HUANG, F.-L., AND LIN, C.-J. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning* 85, 1 (2011), 41–75.
- [131] ZHANG, H. The optimality of naive bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004)* (2004), V. Barr and Z. Markov, Eds., Association for the Advancement of Artificial Intelligence Press.
- [132] ZHANG, J., ZHANG, R., ZHANG, Y., AND YAN, G. The rise of social botnets: Attacks and countermeasures. *Institute of Electrical and Electronics Engineers Transactions on Dependable and Secure Computing* 15, 6 (2018), 1068–1082.
- [133] ZHU, J., ROSSET, S., ZOU, H., AND HASTIE, T. Multi-class adaboost. *Statistics and its interface* 2 (2006).