University of Brasília
Institute of Exact Sciences
Department of Computer Science

AgentBotSpotter: A Multi-agent System for Online Twitter Bot Detection

Jefferson Viana Fonseca Abreu

Thesis presented in partial fulfillment of the requirements for the degree of Master of Science in Informatics

Advisor
Prof. Dr. Célia Ghedini Ralha

Brazil
2021

University of Brasília

Institute of Exact Sciences
Department of Computer Science

AgentBotSpotter: A Multi-agent System for Online Twitter Bot Detection

Jefferson Viana Fonseca Abreu

Thesis presented in partial fulfillment of the requirements for the degree of Master of Science in Informatics

Brazil, Brasília, September 17 2021

Dedication

To the reader.

Acknowledgements

I thank everyone who somehow collaborated with the elaboration of this work. Special thanks to my dear mother Geni, who let me pursue the path to knowledge, on my own, supporting me whenever I needed it. I would not achieve this without you. Distinctive thanks too to My advisor Prof. Célia Ralha who shared her expertise and wisdom during this journey; My comrade Prof. João Gondim who, after many years working together, keeps surprising me with his mastery of research; My beloved partner Judi who always is by my side, advising, understanding, and caring; Prof. Jorge Fernandes who introduced me into the fascinating world of the online social networks; and to former Brazilian president Lula da Silva who democratized access to public and quality higher education, and filled my heart with hope.

“Science is the power of Man”

“Ikari Gendo” in the animated TV series Neon Genesis Evangelion by Hideaki Anno

Abstract

Online social networks like Twitter provide a novel channel to allow interaction between human beings. However, its success has attracted interest in attacking and exploiting through a wide range of unethical activities, such as malicious actions to manipulate users. One of the methods to carry out these abuses is the use of bots on Twitter. Such behavior needs investigation aiming to mitigate its effects. Recently, machine learning (ML) classifiers to distinguish between real and bot accounts have proven advances. In this work, we explored the technique by constructing a multi-agent system (MAS) capable of performing the Twitter bot detection autonomously. It is based on supervised classification with three ML algorithms and a reduced set of features. When tested offline, this MAS achieved good performance with an average of AUC equal to 0.9856 and a standard deviation of 0.0199. Using it for online bot detection on a Proof of Concept results suggest that 88.19% of bots detected were correctly labeled. On a comprehensive experiment, more than 780 thousand tweets were captured during five days. 1,597 tweets were from profiles classified as bots (678 unique profiles), indicating that the approach used is feasible and practical for the real-time bot detection problem.

Keywords: Twitter, bots, detection, on-line detection, agents, multi-agents system

Resumo

Redes sociais online como o Twitter fornecem um novo canal para permitir a interação entre seres humanos. Porém, seu sucesso atraiu o interesse em ataques e exploração por meio de uma ampla gama de atividades antiéticas, como ações maliciosas para manipular usuários. Um dos métodos para realizar esses abusos é o uso de robôs no Twitter. Tal comportamento necessita de investigação com o objetivo de diminuir seus efeitos. Recentemente, classificadores com aprendizado de máquina para distinguir entre contas reais e de robô têm avanços comprovados. Neste trabalho, explorou-se esta técnica através da construção de um sistema multiagentes capaz de fazer a detecção do robô do Twitter de forma autônoma. Este sistema é baseado na classificação supervisionada com três algoritmos de aprendizado de máquina e um conjunto reduzido de *features* descoberto e publicado em um trabalho anterior. Quando testado offline, este sistema multiagentes obteve bom desempenho com uma média de AUC igual a 0,9856 e um desvio padrão de 0,0199. Ao usá-lo para detecção de robôs online em uma prova de conceito, resultados sugerem que 88,19% dos robôs detectados foram rotulados corretamente. Em um experimento mais abrangente, mais de 780 mil tuítes foram capturados e analisados. Dentre esses, 1597 eram provenientes de perfis classificados como robôs (678 perfis únicos), indicando que a abordagem usada é viável e prática para o problema de detecção de robô em tempo real.

Palavras-chave: Twitter, robôs, detecção, detecção on-line, agentes, sistema multiagentes

Contents

List of Figures

List of Tables

List of Acronyms

ABS	Agent BotSpotter
ACL	Agent Communication Language
AI	Artificial Intelligence
AUC	Area Under the Curve
AutoML	Automated Machine Learning
CPU	Central Processing Unit
CSV	Comma Separated Value
FIPA	Foundation for Intelligent Physical Agents
GPU	Graphics Processing Unit
MAS	Multi-agent System
ML	Machine Learning
NB	Naïve Bayes
ocSVM	One-class Support Vector Machine
OSN	Online Social Network
PADE	Python Agent Development framework

PoC	Proof of Concept
RAM	Random Access Memory
RF	Random Forest
Sklearn	Scikit-Learn
SVM	Support Vector Machine
TPOT	Tree-based Pipeline Optimization Too

Chapter 1

Introduction

Online social networks (OSN) allow people to interact on a large scale. Among the most popular OSNs, we highlight Twitter. The relevance of this social media is so considerable that several notable people and heads of state adopt it as one of their primary media (Borges, 2020; Cruz, 2020; RedaçãoCorreio24horas, 2020). Nevertheless, a common problem on this social media is the presence of social bots (from now on, treated by bots) that can act like human users and perform malicious actions (Fonseca Abreu et al., 2020; Varol et al., 2017).

Consequently, on Twitter bot detection literature, there are good reviews with some focus on Machine Learning (ML) approaches (Alothali et al., 2018; Latah, 2020). Some works are using unsupervised ML algorithms to perform real-time clustering of profiles considered malicious (Chen and Subramanian, 2018; Miller et al., 2014). Other examples use approaches to improve their respective classifiers with newly extracted data in real-time (Al-Qurishi et al., 2018; Minnich et al., 2017). The works of Cresci et al. (2016); Rodríguez-Ruiz et al. (2020) perform offline classification using real data extracted from Twitter. But Varol et al. (2017) does online bot detection with an update published and deployed recently (Sayyadiharikandeh et al., 2020). However, the creation of good training data in the domain of Twitter bot detection is often difficult.

Another challenge is the definition of a feature set that facilitates bot classification. In Fonseca Abreu et al. (2020) there is the choice and definition of a reduced set of features that allows the detection of bots with results comparable to the state of the art in Twitter bot detection (Rodríguez-Ruiz et al., 2020). In such works, a higher number of features is used to describe the problem. Some of them have greater complexity when compared to the reduced feature set presented in (Fonseca Abreu et al., 2020). Defining the best feature set in ML approaches is vital since unlimited empirical experiments are costly and sometimes impossible to perform.

The literature review also includes some exciting works about political bots and their influence on general elections. An example is a study carried out by Fernquist et al. (2018), which demonstrates that bots acted in the 2018 Swedish general elections. Beğenilmiş and Uskudarli (2018) also successfully detected the performance of bots in the 2016 American elections, while Howard and Kollanyi (2016) conducted a study on the operation of bots during BREXIT in 2016. After the 2018 elections in Brazil, there was great prominence in electoral campaigns on the Internet, making OSNs a favorable channel for bot usage. As a result, the Brazilian national congress established a Joint Parliamentary Committee of Inquiry (*Comissão Parlamentar Mista de Inquérito - CPMI*) to investigate, among other issues, the use of bots to influence the results of the 2018 elections (da Mata, 2019). The mentioned use of bots to manipulate public opinion in elections and referenda indicates their potential to undermine democracies.

1.1 Problem description

Some individuals are willing to perform malicious actions seeking to gain undue advantage, with people being considered the weakest link in information security (Mitnick and Simon, 2003; Schneier, 2001). One of the methods used to carry out abuses is using bots. This problem consists of user profiles in online social networks that impersonate human beings through automated interactions. Bots, grouped as botnets, can act in a coordinated manner to enable more powerful and harmful attacks (Mazza et al., 2019). Examples of malicious actions using bots include spreading spam Chu et al. (2012); Stringhini et al. (2010); Tavares and Faisal (2013); Wang (2010), vectors for phishing Abreu et al. (2019, 2020); Shafahi et al. (2016), or the dissemination of fake news (Freitas et al., 2015).

Works like Abreu et al. (2019, 2020); Shafahi et al. (2016) show that bots can quickly be developed and act on Twitter performing malicious actions. In Chu et al. (2012) there is a discussion on why bot detection is not more ostensible. Credit is given to an alleged dependence that Twitter has on legitimate content generated by bots, for example, the profiles of news portals. However, the coexistence between humans and malicious bots negatively impacts the experience of platform users. Therefore, it is necessary to develop techniques that make bot detection more detailed to make Twitter a more friendly, safe, and reliable environment.

Additionally, an automatic solution that autonomously can detect bots on Twitter can help solve the problem. In this way, a multi-agent system (MAS) is an attractive approach since the performance of autonomous actions over a complex and mutable environment (like Twitter) is inherent to this kind of solution. The autonomous actions occurs because each agent can decide for itself to achieve its objectives Wooldridge (2009). The conducted literature review (April-May 2020) could not find works using MAS for Twitter bot detection what might indicate a research gap. Chapter 5 presents results involving the application of this approach for Twitter bot detection. The results show that when comparing to traditional ML techniques, a MAS using an ensemble of ML algorithms can reach relatively reasonable performance measures. To cite, an average of AUC 0,9856 with a standard deviation of 0,0199 against the best ML algorithm, RF with AUC 0,9878 and a standard deviation of 0,0170.

Moreover, the MAS approach allows the integration of several ML algorithms in the same architecture improving the robustness of the solution. For this task, a pattern for autonomous group decisions can be adopted, for example, a loan model or a poll. The previous standard (poll) was the one adopted on the solution as discussed in Chapter 4. According to Dietterich (2000), ensemble learning consists of selecting a set of different hypotheses to combine their predictions. Thus, the combination of ML algorithms results in an ensemble learning approach. Other desirable attributes for adopting the MAS approach are the agents' deliberative autonomy and the possibility of constructing a distributed approach (improving the system scalability).

The solution to the Twitter bots problem proposed in this work is called AgentBotSpotter (ABS). ABS is a MAS capable of detecting bots on an online stream of tweets. For this, three different agents were developed that cooperatively interact among them and with the environment. The agents capture a tweet stream based on pre-established keywords and decide online if any author is a bot. The decision consists of an ensemble of supervised ML techniques, including RF, NB, and SVM. The Twitter REST API provides the tweet stream.

1.2 Hypothesis and research question

The hypothesis of this work is the development of a MAS with ML algorithms to detect Twitter bots with an online flow of tweets. Online here means that the data source is a flow of tweets, and the detection decision takes place as close as possible to the timestamp of the tweet publication. Thus, the detection occurs in a period that the information is still useful to react against bots. Moreover, in previous works Abreu et al. (2019, 2020) the default bot detection response from the Twitter platform took from hours to days. Thus, a tool that performs the detection in less than hours is an improvement. Besides, the construction of the MAS requires an offline assessment once that is simple to calculate some performance measures offline before the online execution. Thus, the research questions (RQ) that this work intends to answer are:

- RQ1: Is it feasible to build a MAS to detect Twitter bots using an offline dataset to train the ML algorithms?

- RQ2: Can the MAS detect bots over an online flow of tweets in a period that the information is still useful to react against bots?

1.3 Goals, boundaries and contributions

The main objective of the research is to develop a MAS for the detection of bots on Twitter over an online flow of tweets. There are some secondary objectives as follows:

- empirically establishes, and validate a set of ML algorithms with features suitable for the classification of Twitter bots.
- develops agents with specific goals, such as:
  - agent to capture tweets.
  - agents to classify Twitter users using different ML algorithms.
  - referee agent to choose the best classification result of the classifier agents.
- integrates the aforementioned agents in a MAS, using an adequate communication and interaction protocol, applying an ensemble learning approach (Section 2.3.1).

Boundaries

This work contains the following boundaries:

- it only uses supervised ML algorithms;
- it uses profile features that are returned directly from Twitter, not doing any additional analysis, e.g., sentiment, time behavior, social graph, natural language processing;
- the MAS was implemented using the Python Agent DEvelopment framework (PADE);
- the online MAS depends upon Twitter's API response time to the requisitions.

Contributions

The contributions of this work are:

- initial results using a reduced set of five features based on a Twitter-specific user profile (achieving AUC greater than 0.9) as shown in Fonseca Abreu et al. (2020);
- validation of the reduced feature set presented in Fonseca Abreu et al. (2020) using TPOT tool (Automated machine learning - AutoML) (Olson and Moore, 2019);

- a MAS bot classifier with different agent types that combines three ML algorithms presenting good performance with the average of AUC equal to 0.9856 and standard deviation of 0.0199;
- proof of concept with the MAS capturing and classifying users provided by an online stream of s where 88.19% of bots detected were correctly labeled;
- a comprehensive online Twitter bot detection experiment with more than 780 thousand captured during five days using the word *vacina* (in Portuguese means vaccine) where 1,597 s were from profiles classified as bots (678 unique profiles); and
- the average time of detection was 14.428 seconds, and 99.02% of the detections occurred in less than 30 seconds, using reduced computing resources (off-the-shelf notebook).

Publications

Some results were published in scientific venues during the development of this work:

- J. V. Fonseca Abreu, C. Ghedini Ralha and J. J. Costa Gondim. A Multi-agent Approach for Online Twitter Bot Detection. *VI Jornadas Nacionales de Investigación en Ciberseguridad (JNIC)*, 2021 Live, doi: 10.18239/jornadas_2021.34.03.
- J. V. Fonseca Abreu, C. Ghedini Ralha and J. J. Costa Gondim, Twitter Bot Detection with Reduced Feature Set, *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2020, pp. 1-6, doi: 10.1109/ISI49825.2020.9280525.
- J. V. Fonseca Abreu, J. H. Cabral Fernandes, J. J. Costa Gondim, C. Ghedini Ralha. Bot Development for Social Engineering Attacks on Twitter. *CoRR abs/2007.11778*, 2020, arXiv:2007.11778v1.

1.4 Document presentation

The rest of the manuscript includes:

- In Chapter 2 an overview of the main fundamentals related to the work is presented, including concepts of Twitter, bots, intelligent agents, MAS, and ML;
- In Chapter 3 the related work is introduced according to a literature review based on two recent review articles by Alothali et al. (2018) and (Latah, 2020). The Appendix A includes the literature review table with all the studied works;

- In Chapter 4 the solution proposal is detailed with the design model, agent reasoning model, architecture, and implementation aspects for Twitter bot detection;
- In Chapter 5 the empirical validation of the proposed MAS is presented with experimental results and discussion;
- In Chapter 6 conclusion and a brief description of future work are presented.

Chapter 2

Background

In this chapter, concepts of Twitter and bots (Section 2.1) are presented based on texts available in the Twitter help center, where there are explanatory texts about the existing functionalities (Twitter Inc., 2020b). Agent and MAS (Section 2.2), and ML (Section 2.3) are presented based on the content of the book of Russell and Norvig (2010) that presents a modern approach to Artificial Intelligence (AI).

2.1 Twitter and bots

Twitter consists of a microblog, where users post short messages (from now on referred to as s) in their respective profiles (Fazil and Abulaish, 2018). For Ruz et al. (2020) Twitter is a fast-growing online platform where people can create, post, update, and read short text messages called s.

Rodríguez-Ruiz et al. (2020) defines Twitter as a micro-blogging social network that allows posting short messages of up to 280 characters called s. Twitter's users can interact with one another by replying to s, mentioning other users in their s, or reposting another user's message, an action called reing. Furthermore, users can follow each other to keep up with the s posted by other users. The social platform allows accessing its services through a web page, mobile applications, and an application programming interface (API) for all registered users.

According to Twitter Inc. (2020b), the s have a length of up to 280 textual characters. These may contain mentions of other profiles by typing '@' and the username to be mentioned. Other multimedia elements are also possible, such as videos, images, polls, geolocation, among others (Gui et al., 2019). Each has some attributes that allow its identification within the platform. Examples of attributes are identifier number, author profile, creation timestamp, and body text. Similar to the , a profile also has several attributes that allow its characterization: username, number of followers, number of friends, s produced by this account, among others.

Twitter profiles establish one-way relationships following each other. That is, a x profile that follows a y profile does not imply that y follows x (Chu et al., 2012). In Twitter Inc. (2020b), the interaction between profiles occurs through the writing of s mentioning each other or through direct messages that are a private chat. Each profile has a timeline, which is the main element of the social network, that shows the s from the accounts that the user follows and sponsored content. If a profile violates the rules of the platform, there are mechanisms for making complaints. After the complaint, the platform investigates whether the profile violates the rules and applies punishments, ranging from excluding s to banning the user from Twitter.

According to Twitter Inc. (2020a), Twitter has a REST API (hereafter called API), where it is possible to consume platform resources in a similar way to the application. Through the API, it is possible to retrieve the attributes of profiles and s transparently. The attributes of s and user profiles are essential in detecting bots, as this occurs based on these or other attributes derived from those provided by Twitter. As an example of metrics derived from combinations of attributes, we can mention: user activity frequency information Mazza et al. (2019); Varol et al. (2017); content information Fernquist et al. (2018); Stringhini et al. (2010); Subrahmanian et al. (2016); and information on relationships between profiles (Chu et al., 2012; Fazil and Abulaish, 2018).

As presented in Chapter 1, the existence of Twitter bots may represent a problem for users of the platform, so it is necessary to understand this problem to be able to solve it. It is essential to highlight that terms and conditions provided by Twitter allow for automation and bots, but it prohibits the use of bots for malicious purposes (Twitter Inc., 2020b). There are many different definitions of bots in the literature. Many studies in the literature in this field focus merely on distinguishing between legitimate users and malicious social bots (Dickerson et al., 2014; Echeverria et al., 2017; Varol et al., 2017).

In Gorwa and Guilbeault (2020) a typology of bots is presented. The topology suggests six different types of bots: web robots (crawlers and scrapers), chatbots (human-computer dialog system which operates through natural language via text or speech), spambots (bots that post on online comment sections and spread advertisements or malware on social media platforms), social bots (various forms of automation that operate on social media platforms), sock puppets and trolls (fake identities used to interact with ordinary users on social networks), and cyborgs and hybrid accounts (a combination of automation and human curation).

According to Latah (2020), social bots can be categorized into different fine-grained classes. Chu et al. (2012) defined a bot as an automated account used for spam purposes and a cyborg as a bot-assisted human or human-assisted bot. Tavares and Faisal (2013) categorized Twitter accounts into three groups (i) personal, (ii) managed, and (iii) bot-controlled accounts based on their time intervals between s. Even though they may be involved in malicious activities, no real users' accounts (such as human spammers, trolls, managed accounts) are defined as social bots. Stringhini et al. (2010) also defined different spambots displayers, braggers, posters, and whisperers.

In Howard and Kollanyi (2016) bots have been defined as executable software that automates the interaction between a user and content or other users. Fernquist et al. (2018) adopts this definition, but with a more comprehensive interpretation, the researchers consider any bot-like mechanical behavior not necessarily prevenient from a software program. Consequently, their definition may include sock puppets and trolls too. Similarly to Fernquist et al. (2018) this project proposal (Chapter 4) also uses only supervised learning methods, so the bots detected by it are closer to their examples. Therefore, the definition adopted here is the same as (Fernquist et al., 2018): executable software that automates the interaction between a user and content or other users or fake identities used to interact with ordinary users on social networks (not necessarily a software program).

2.2 Agent and MAS

According to Poole et al. (1997) computational intelligence is the area of study that deals with the development of intelligent agent design. Russell and Norvig (2010) defines an agent as something that acts. However, a computational agent does more like: operate under autonomous control, perceive its environment, persist for an extended period, adapt to changes, and be capable of creating and achieving goals. As all computer programs have the purpose of processing something, we can differentiate an agent by the ability to perceive its environment with its sensors and act on it using its actuators, as illustrated in Figure 2.1.

According to Wooldridge (2009) and Weiss (2013), an MAS is composed of a set of agents capable of interacting with the environment and with each other, cooperatively or competitively, in search of achieving one or more individual goals or collective. The agents of a MAS can play different roles to achieve their goals, whether through the independent execution of actions, adapting to changes in the environment, or interacting with other agents. A MAS project needs system modeling to define the behavior and reasoning of each agent. Also, the communication and interaction protocol for the group of agents needs to be defined, and which tools will be used in the implementation of the system. In this section, concepts of agents and an overview of MAS techniques are presented.

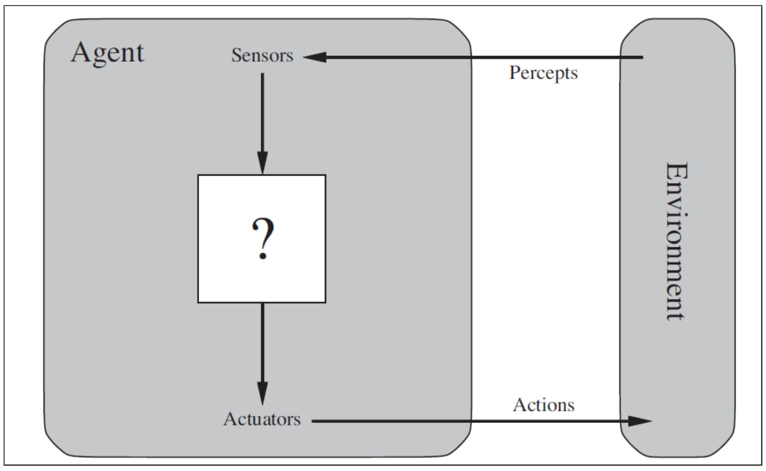


Figure (2.1) Schematic representation of a computational agent (Russell and Norvig, 2010).

2.2.1 Agent reasoning

In Russell and Norvig (2010), a rational agent needs to perform actions seeking to achieve the best possible result with the set of available information. Rationally acting may involve dealing with uncertainties, logical inferences, and reflexes. It is possible to classify agents according to their behavior and performance in the environment at various levels of complexity. These levels are simple reflex agent, model-based reflex agent, goal-based agent, utility-based agent, and agent with learning, from the simplest to the most complex. However, it is possible to extend all types of agents using ML.

The rationality of the simple reflex agents is the simplest of all categories. Their actions are responses mapped directly onto the perceptions received in a conditional flow. It resembles instinctual thinking, like the body's reflexes, like a person who closes eyes when a powerful light turns on. There is no static memory for state registration, only conditionals for the action chosen from the perception.

The model-based reflex agents still have actions mapped directly to perceptions in a conditional flow but consider the agent's current state in their response. Its best action search function considers an internal state machine for the return. The future state is saved according to action and the previous state.

The goal-based agent has advanced rationality. Agents of this type consider, among the universe of actions to be taken, the best in the pursuit of completing one or several objectives. In addition to considering states, its action-seeking function considers how actions can modify the external environment. The agent chooses an action using a vector of possibilities. Furthermore, the goals can be updated according to changes in the environment, the goals achieved, and the agent's state.

In addition to considering objectives and considering the best action to take, the utility-based agent can consider other variables unrelated to the environment it is in, called utilities. This kind of agent answer may be the most economical action, for example, or it may have some other influence on the agent's reasoning. Very similar to the goal-oriented structure, except for the agent's utility, which influences the chosen action and the updating of goals.

The agent with learning characteristics is responsible for making improvements, and the performance element, responsible for selecting external actions. The performance element was previously thought of as the complete agent: it receives insights and decides on actions. The learning element uses feedback from the critic about how the agent is working and determines how the performance element should be modified to work better in the future.

2.2.2 MAS design

An essential element in the design of intelligent agents is the correct characterization of the environment. According to Russell and Norvig (2010); Weiss (2013); Wooldridge (2009) the environment must be characterized according to six essential aspects:

- Observable - the fraction of the agent's environment available for use, considering the perception sensor(s). This aspect is defined as totally or partially.
- Deterministic - if the following environment states are determined by the current state or by the result of the agent's action in the environment. Otherwise, it is non-deterministic or stochastic.
- Episodic - if it is possible to divide each agent's experience (including perception and action). In this case, the next episode does not depend on previous episodes (e.g., assembly line). Otherwise, the environment is sequential (e.g., chess game).
- Static - if the environment does not change while the agent is deliberating on what to do. Otherwise, it is dynamic. If the environment does not change over time, but the agent's performance can change, there is a semi-dynamic environment.
- Discrete - if it is possible to represent the characteristics of the environment employing a finite set of elements. Otherwise, it is continuous (e.g., an autonomous taxi driver is continuous in time and space).
- Multi-agent - when there are several agents in the environment, otherwise there is a single agent.

For an adequate definition of the rational agent's task environment, the PEAS (Performance measure, Environment, Actuators, Sensors) was defined (Russell and Norvig, 2010). The PEAS is considered a pre-design phase, where each agent must be specified by the aspects of performance measurement, environment, perceptions, and actions in the most detailed way. Different agent tasks do not have a fixed performance measure. The designer is responsible for determining a performance measure to evaluate the agent's behavior in a specific environment. For each sequence of rational agent perceptions, the designer must select an action that will maximize the defined performance measure. This measure is built through evidence provided by: the sequence of perceptions and any internal knowledge of the agent.

2.2.3 Agent modeling

In the literature of agents, there are many approaches to modeling agent-oriented software systems, but this task requires specific modeling tools and techniques, unlike the Unified Modeling Language (UML). Agent-oriented modeling can analyze agents' mental states and beliefs rather than machine states; plans and actions rather than procedures; methods of communication, negotiation, interaction, and social capability rather than input functionalities and exit; goals, desires, and other aspects. Despite this, the UML can contribute to the diagram of phases related to implementing a MAS through class and sequence diagrams, among other diagrams (Bresciani et al., 2004). In this work, we applied the modeling methodology called Tropos to design agent-oriented software systems.

According to Bresciani et al. (2004), Tropos is based on two key ideas. First, the notion of agent and all related mentalistic notions (for instance, goals and plans) are used in all phases of software development, from early analysis down to the actual implementation. Second, Tropos also covers the early phases of requirements analysis, thus allowing for a deeper understanding of the environment where the software must operate and interactions between software and human agents. five phases divides the development of a project using Tropos: initial requirements, late requirements, architectural design, detailed design, and implementation.

In the initial requirements diagram, domain stakeholders are identified and modeled as actors. These, in turn, have dependencies to achieve their goals and may (or not) share resources. It is possible to declare the why, what, and how of system features at this stage. The previous model is extended at the late requirements diagram, including new actors representing the system and probable dependencies with the other actors already identified in the environment. Such dependencies will allow defining the functional and non-functional requirements of the future system.

The architectural design diagram represents a global view of the system by expanding the final requirements diagram and displaying more details. It defines the system's overall architecture in terms of subsystems, interconnected through data and control flows. In addition, it is also possible to observe a mapping of system actors to a set of software agents, each characterized by specific resources. In the detailed design diagram, the idea is to specify the agents' resources and interactions. For a better level of detail, ideally, the implementation platform has already been chosen. The implementation platform must be considered to perform a detailed design that will map directly to code.

For the initial modeling phases, the language i* (Intentional STrategic ActorRelationships modeling - iStar) is adopted. This methodology has a focus on the intentional (why?), social (who?), and strategic (how?) dimensions of the software (Dalpiaz et al., 2016). Tropos allows modeling the functionality of an application based on objectives through diagrams.

As illustrated in Figure 2.2, each of the diagrams in Tropos has visual components used to represent graphically: actor, agent, role, goal, soft-goal, task/plan, and resource (Pimentel and Castro, 2018). The standard circle is used for generic actors without specialization. Circles also represent the agents, but a straight line-shaped detail in its upper base indicates an intelligent agent. When the circle has a curved line at the bottom, it represents roles. A Role is an actor who has some kind of specialization in the environment in which it operates.

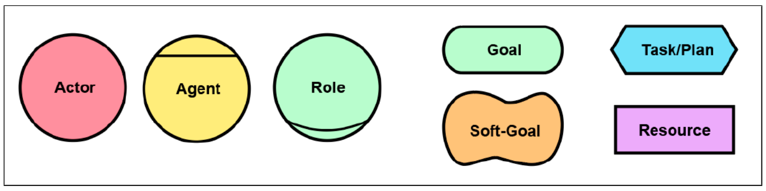


Figure (2.2) Tropos visual components used to represent actor, agent, role, goal, soft-goal, task/plan, and resource.

The Goal element represents an objective or sub-goal that the actor or system agent must achieve. On the other hand, Soft-Goal represents a capability or the ability that an actor or agent has to define, choose, and execute a plan to fulfill a goal. It can also represent issues related to quality and performance. The Task/Plan element represents the actions to be performed by the actor or agent to achieve their goals. The resource represents an entity's information that the actor or agent needs to perform a task. In other words, the resource can be generated or used by the elements involved to achieve objectives.

Usually, these elements are related to each other through interconnections. Each connection represents the current relationship level between the elements, as shown in Figure 2.3. According to Dalpiaz et al. (2016), when two elements (e.g., task or goal) are linked to a goal with an association of type *And*, it indicates that both must be executed to reach the goal. Unlike when using the *Or* association, it is enough that one of the elements is performed or satisfied.

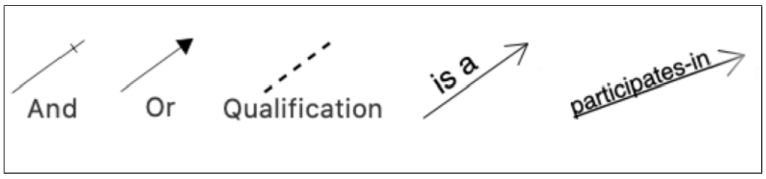


Figure (2.3) Tropos interconnections.

Qualification serves to connect Soft-Goals to the elements to which it refers. Graphically a T-shaped arrowhead indicates AND refinement, a solid arrow pointing to the parent element represents the OR refinement, and a dashed line represents a Qualification. The association 'is a', represents that an actor is a generalization/specialization of other actors or generalized roles of this one. On the other hand, participates-in represents an association between two actors, except generalization/specialization.

2.2.4 Agent development tools

It is possible to develop MAS by writing its entire structure manually using a programming language. Although, this is not necessary, as there are already several tools to support the development of MAS. These tools already have standards, classes, communication structures, and other artifacts ready to be used, enabling the creation of intelligent agent networks in various applications, allowing the designer to focus on problem resolution.

The Java Agent Development Framework (JADE) is a set of tools for developing MAS that implements the architecture proposed by the Foundation for Intelligent Physical Agents (FIPA) to define the interaction protocol in the system (Bellifemine et al., 2007). JADE is a middleware for developing and executing intelligent agents in Java. It haves all the necessary infrastructure for the creation, communication, and administration of agents is available in classes ready to be used or extended and customized. JADE also provides a platform for executing the developed agents. The platform also has support for distributed environments, as long as each machine belonging to the scenario has an active JADE platform. JADE's tools are also an agent administration and debugging module, making the work easier for the programmer and the designer to develop the MAS.

JACK Intelligent Agents is an agent framework designed as a set of lightweight components with high performance and strong data typing. It is a business development environment oriented to agents and was developed to provide extensions to the Java language, inserting object-oriented elements. Furthermore, it was designed as a set of strongly typed components with low computational cost, providing high performance. Although JACK has a consolidated belief-desire-intentions (BDI) approach and a complex graphical interface, it is possible to run it on machines without a sizeable computational capacity (Howden et al., 2001).

The PADE is a framework for developing, running, and managing MAS in distributed computing environments written in Python language (Melo et al., 2019). PADE is open software, licensed under the terms of the MIT license, and uses the Twisted project libraries to implement the communication between the network nodes. PADE is developed considering the requirements for the automation system and offers the following features:

- abstraction for implementing agents using Object Orientation;
- agent execution environment entirely in Python code;
- module for the implementation of the protocols defined by FIPA;
- module for construction and processing of messages in the FIPA-ACL (Agent Communication Language -ACL) standard. In addition, allowing the sending of serialized objects as the content of ACL messages.

PADE provides the implementation of cyclical and temporal behaviors, database interaction, cross-platform, easy to install and configure, among other features (Melo et al., 2019).

2.3 Machine learning

According to Alpaydin (2020); Russell and Norvig (2010), ML is the programming of computers to optimize a performance criterion using example data or previous experiences. There are several ML methods in the literature. Inductive learning occurs when the program generates a rule from specific input-output pairs. It is also possible to make software entities perform deductive learning by deriving new rules from a general rule. There is also reinforcement learning, where the agent receives rewards for learning an optimal (or almost optimal) policy for the environment without needing any prior information.

Also, according to Alpaydin (2020), inductive methods can be divided into unsupervised, semi-supervised, and supervised. The semi-supervised method consists of designing a general rule from a data set that is partially labeled. That means some data have the information of the output expected by the algorithm linked to them. Unsupervised learning algorithms generate knowledge from completely unlabeled input data.

In Section 2.3.1, supervised learning methods are detailed since this is the central technique employed. This choice occurred based on the literature review conducted (Chapter 3), which showed that the most used to detect bots on Twitter are RF, SVM, and NB supervised ML algorithms. Also, supervised learning allows the production of data output from the previous experience, allowing to improve the classifier.

2.3.1 Supervised learning

Supervised learning occurs when using, as a starting point, a set of labeled training data (input-output pairs), and from this one seeks to design a general rule that works to classify unlabeled data. According to Russell and Norvig (2010), a task can be described by:

> Given a training dataset N with input and output pairs $(x_1, y_1), (x_2, y_2)...(x_n, y_n)$, where each y_1 was generated by a function $y = f(x)$ unknown, we want to find the function h that approximates the function f true.

The values x_j and y_j are not necessarily numeric values. The h function is called a hypothesis. Learning is searching in the space of possible hypotheses for the one that will perform well. For measuring the accuracy of a hypothesis, a set of test examples (*test set*) is provided to the learning algorithm, which must be distinct from the training set. We say that a hypothesis achieves generalization if it can predict with a reasonable degree of accuracy the output values y for inputs x, which are not part of the training set. Sometimes the f function is stochastic and is not strictly a x function. In this case, a conditional probability distribution $P(y|x)$ is needed to be learned.

When y is a number (e.g., tomorrow's temperature), the learning problem is named a regression. Technically, solving a regression problem is finding a conditional expectation or mean value of y. The probability that we find precisely the correct real-valued number for y is close to zero. The classification problem aims to make a selection of entities following common characteristics. When a finite set of values forms the output y, the learning problem is called classification. Classification refers to discrete values while regression, continuous values. There are a few different types of classification.

However, this work covers only two: one-class and multi-class. According to Rodríguez-Ruiz et al. (2020), one-class classification is a branch of supervised classification, where the training set contains only examples of a class. By contrast, multi-class classification occurs when the training set x_j contains more than one class data. As algorithms of a class can classify input data as belonging to the class or not belonging to the class, such algorithms can also be called binary classifiers. Among the various supervised learning algorithms in the literature, this section will detail three: Random Forest(RF), Support Vector Machine (SVM), and Naïve Bayes (NB).

Random Forest

According to Alpaydin (2020); Russell and Norvig (2010), the inductive learning algorithm called decision tree (DT) is one of the most simplistic and efficient. This algorithm consists of a hierarchical data structure that implements the divide-and-conquer strategy.

A decision tree represents a function that inputs an array of attribute values and returns a single output (called a decision). It is an efficient non-parametric method used for classification or regression.

Each internal node of the tree corresponds to a test of a feature. The edges represent the possible paths that the algorithm takes according to the feature value. The sheets represent the possible decisions when traversed each path. Figure 2.4 illustrates a DT using ten features (Alternative, Bar, Fri/Sat, Hungry, Customers, Price, Raining, Reserve, Type, Estimated Wait). It is possible to follow a tree path and decide when to wait for a restaurant vacancy. Note that the tree ignores the features 'Price' and 'Type'. These were considered little explanatory by the algorithm due to the examples contained in the training set.

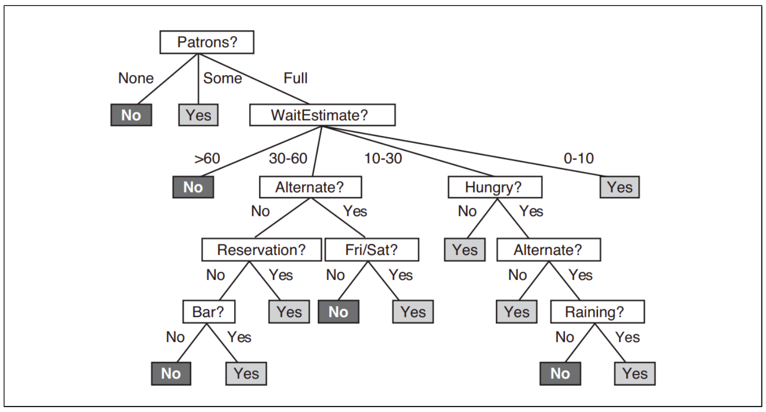


Figure (2.4) DT example for the restaurant waiting list problem (Russell and Norvig, 2013).

At Mitchell (1997) a basic algorithm for decision tree learning is presented. This algorithm learns decision trees by constructing them top-down, beginning with the question "which attribute should be tested at the root of the tree?" Each instance attribute is evaluated using a statistical test to determine how well it alone classifies the training examples. The best attribute is selected and used at the root node of the tree. A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node (i.e., down the branch corresponding to the example's value for this attribute). The entire process is repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree. The algorithm is presented in Listing 2.1, where *Examples* are the training examples, *Target_attribute* is the attribute whose value is to be predicted by the tree, and *Attributes* is a list of other attributes that the learned decision tree may test.

```
def tree_construction(Examples, Target_attribute, Attributes):
    Create a Root node for the tree
    if every bit of the Examples are positive:
        return the single-node tree Root, with label =+
    if every bit of the Examples are negative:
        return the single-node tree Root, with label =-
    if Attributes is empty:
        return the single-node tree Root, with
            label = most common value of Target_attribute in
            Examples

    A = the attribute from Attributes that best classifies Examples
    the decision attribute for Root = A
    for each possible value vi, of A:
        Add a new tree branch below Root,
            corresponding to the test A = vi
        Let Examples_vi, be the subset of Examples
            that have value vi for A
        if Examples_vi, is empty Then:
            below this new branch add a leaf node with
                label = most common value of Target_attribute in
                Examples
        else:
            below this new branch add the subtree
                tree_construction( Examples_vi, Target_attribute,
                Attributes - {A})
    return Root
```

Listing (2.1) The DT learning algorithm pseudocode (Mitchell, 1997).

The DT algorithm has some negative points, for example, the existence of different trees for the same problem. According to Russell and Norvig (2010), for a set of m features there are more than 2^{2^m} distinct trees. Also, DT can lose generalization when dealing with too many features or with a very different case from the training set. Seeking to solve some of the problems of the DT algorithm, Ho (1995) conceived the RF algorithm.

According to Breiman (2001), the RF algorithm has a combination of decision tree predictors. RF is a classifier consisting of a collection of tree-structured classifiers $h(x, \Theta_k)$, $k = 1, ...$, where Θ_k are independent random vectors, identically distributed, and each tree casts one vote for the most popular class in the x entry. According to Ho (1995) there are many ways to construct different trees, but an arbitrarily introduced difference does not necessarily give helpful trees. The trees required are 100% accurate on training data and yet have different generalization errors.

Randomization has been a powerful tool for introducing differences in classifiers. Previously it has been used to initialize training algorithms with different configurations that eventually yield different classifiers. The use of randomization in selecting feature vector components is merely a convenient way to explore the possibilities. A decision tree is constructed in each selected subspace using the entire training set. In this way, each tree depends on values of an independently sampled random vector, with the same distribution for all trees in the forest. Thus, it is possible to achieve a more consistent prediction for entirely new data or a large set of features, decreasing the risk of overfitting (Ho, 1995).

Support Vector Machine

The SVM is one of the most popular kernel-based ML algorithms, known as an off-the-shelf solution. According to Russell and Norvig (2010), it is an algorithm that is worth using when dealing with an application domain in which there is no prior knowledge or details. In Alpaydin (2020), SVM is an example of a kernel machine. This classifier applies a maximum margin method allowing the model to be explained as a sum of the influences of a subset of the training instances. According to Bishop (2006), the SVM approaches the problem through the concept of the margin. The margin is defined to be the smallest distance between the decision boundary and any of the samples. In SVM the decision boundary is chosen to be the one for which the margin is maximized.

At Theodoridis and Koutroumbas (2008) kernel methods owe their name to the use of kernel functions. These functions enable them to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates.

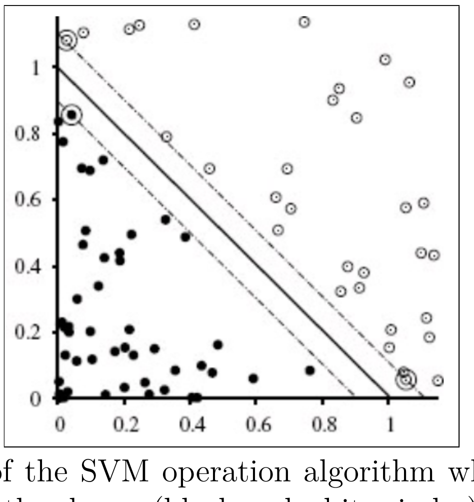


Figure (2.5) Illustration of the SVM operation algorithm where the continuous line is the hyperplane that divides the classes (black and white circles). The vector support (dots with large circles) is the closest example to the separator (Russell and Norvig, 2010).

The SVM algorithm treats the training data as points in space. The homogeneous examples are divided, and starting from this representation, it calculates a hyperplane that divides the data into classes. In this way, it is possible to check which quadrant new examples fit. Figure 2.5 illustrates how the algorithm operates the classes division. According to Russell and Norvig (2010), SVM has three properties that make it attractive. The construction of a maximum margin separator increases generalization. traditionally, the separator is defined as the set of points $\{x : w{\cdot}x{+}b = 0\}$. It is possible search the space of w and b with the descent down the gradient to find the the parameters that maximize the margin and at the same time correctly sort all the examples. The kernel trick allows data incorporation in a higher dimension accepting an expansion of the hypothesis space. Moreover, it is not a parametric algorithm making the methodology flexible to represent complex functions.

Naïve Bayes

The NB is a supervised algorithm that formulates the learning task as a probabilistic inference process (Russell and Norvig, 2010). The NB uses Bayes' theorem (Equation 2.1) to calculate the influence that each feature exerts on the hypothesis isolatedly. Since the algorithm does not consider possible correlations between features, it is called naïve. The probability of a class C with values of features $x_1, \ldots, x_n$ is calculated according to Equation 2.2. The NB algorithm learns and adapts well to large problems, with datasets bigger than the number of features. Furthermore, no search is needed to find the H_{ML} (naive Bayes hypothesis of maximum probability). Finally, the NB algorithm does not present difficulties with noisy or missing data, as it can provide probabilistic predictions when necessary.

$$P(cause|effect) = \frac{P(effect|cause)P(cause)}{P(effect)} \tag{2.1}$$

$$P(C|x_1, ..., x_n) = \alpha P(C) \prod_i P(x_i|C) \tag{2.2}$$

Ensemble learning

According to Russell and Norvig (2010), Dietterich (2000), ensemble learning consists of selecting a set of different hypotheses to combine their predictions. An example is what happens in the RF algorithm, which generates different decision trees that carry out a vote to choose the best classification for a set of input data. For example, considering a set of hypotheses (function $h = 5$) and assuming that their predictions are combined using simple majority voting. For this set, to misclassify an example, at least three of the five hypotheses must be incorrect. Consequently, the use of diverse hypotheses will probably improve the approximation results of the f function. Dietterich (2000) states that three problems contribute to the adoption of ensemble learning:

1. Statistical - arises when the learning algorithm looks for a larger hypothesis space than the available training data. In this case, there may be several different hypotheses that provide the same precision in the training data, and the learning algorithm must choose one of them. There is a risk that the chosen hypothesis does not predict future data points well. In this case, a simple vote of equally good classifier algorithms can reduce this risk.
2. Computational - arises when the learning algorithm cannot guarantee to find the best hypothesis within the hypothesis space. As with the statistical problem, a weighted combination of several different local minima can reduce the risk of choosing the wrong local minima.
3. Representational - arises when the hypothesis space (function h) does not contain any hypotheses that are good approximations of the true f function. In some cases, a weighted sum of hypotheses expands the space of possible represented functions. Therefore, by giving a weighted vote of hypotheses, the learning algorithm may form a more accurate approximation for f.

As per Russell and Norvig (2010) each hypothesis $h(k)$ in the hypothesis set k has an associated error p. Assuming that the errors made by each hypothesis are independent, if p is small, the probability of a large number of incorrect classifications occurring is also small. Therefore, if the hypotheses are minimally different, reducing the correlation between their errors, learning by grouping becomes more effective.

Furthermore, the hypothesis set is a generic way of expanding the hypothesis space. Figure 2.6 illustrates the greater expressive power obtained by cluster learning. Adopting three linear threshold hypotheses, each one ranks the example positively on the unhatched side and ranking as positive any example ranked positively by the three hypotheses. The resulting triangular region is a hypothesis that the original hypothesis space cannot express.

2.3.2 Feature selection

According to Bishop (2006), a feature is an individual measurable property or characteristic of a phenomenon observed. Features describe object attributes: a number, a text, a date, and a boolean, among others. One of the problems in designing a classifier using ML is the selection of features. Guyon and Elisseeff (2003) cite that the goal of selecting features is threefold: (i) to improve the prediction performance of predictors; (ii) provide faster and more cost-effective predictors; and (iii) provide a better understanding of the underlying process that generated the data.

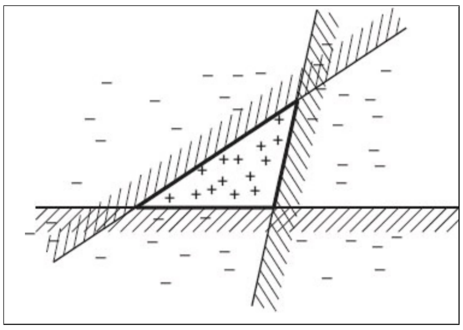


Figure (2.6) Illustration of the greater expressive power obtained by ensemble learning (Russell and Norvig, 2010).

There are several methods in the literature for selecting features, which aim to reduce the number of input variables to those believed to be the most useful for a given model to predict the response variable. Selecting features is primarily focused on removing uninformative or redundant predictors from the model. It is a way of reducing the dimensionality of the model, which will reduce the processing time of the AM algorithm.

According to Chandrashekar and Sahin (2014), the features selection methods fit into three large groups:

1. Filter - use sorting techniques as the main criteria for selecting features by ordering. The features are ranked following a chosen criteria, and those below an established threshold are discarded.

2. Wrapper - uses predictor as a black box and predictor performance as the objective function to evaluate the subset of features. It consists of creating subsets of features and evaluating the impact of their use on the classifier. In this way, it is possible to observe the interactions between the features.

3. Embedded - aim to reduce computation time spent to reclassify different subsets as in wrapper methods. The primary approach is to incorporate the selection of features as part of the training process.

According to Menze et al. (2009); Nembrini et al. (2018), the RF algorithm performs an implicit selection of features, using a small subset of "strong variables" for classification. This strategy makes performance superior when dealing with high-dimensional data. This implicit selection can be viewed through the "Gini importance" and can is a general indicator of relevance for features. Commonly, a splitting criterion in decision trees is the Gini index. Also often, the "Gini importance" is associated with the importance of the impurity corresponding to the index. This feature importance score is a by-product of training the RF classifier.

Also in Menze et al. (2009), at each node t within the T trees of the RF, the ideal division is sought using the Gini impurity $i(t)$. This measures how well a potential split is separating the samples of the two classes at this particular node t. Let $p_k = \frac{n_k}{n}$ be the fraction of the n_k samples of a class $k = \{0.1\}$, out of the total of n samples in node t, the Gini impurity $i(t)$ is calculated according to Equation 2.3.

$$i(t) = 1 - p_1^2 - p_0^2 \tag{2.3}$$

The decay Δi results from dividing and sending the samples to two subnodes t_l and t_r (with the respective fractions of $p_l = \frac{n_l}{n}$ sample and $p_r = \frac{n_r}{n}$) by a limit t_θ on variable θ is defined as shown in Equation 2.4.

$$\Delta i(t) = i(t) - p_l i(t_l) - p_r i(t_r) \tag{2.4}$$

In an exhaustive search over all θ variables available in the node (RF restricts this search to a random subset of the available features) and over the possible limits t_θ, the pair $\{\theta, t_\theta\}$ which leads to a maximum Δi is determined. The decrease in Gini impurity resulting from this optimal division $\Delta i_\theta(t, T)$ is recorded and accumulated for all t nodes in all T trees in the forest, individually for all θ variables, according to Equation 2.5.

$$I_G(\theta) = \sum_T \sum_t \Delta i_\theta(t, T) \tag{2.5}$$

The Gini importance I_G (Equation 2.5) indicates how often a division selects a feature θ. As well as how great was its discriminative value for the investigated classification problem. Usually, the calculus of RF's feature importance intends to validate the features chosen empirically (e.g., Feature set validation experiment at Chapter 4). This work applies an empirically generated feature selection that is similar to the filter-based approach. However, the aforementioned feature selection was empirically validated through the feature selection method embedded in an AutoML tool, as described in Chapter 5.

2.3.3 Automated machine learning

AutoML refers to the process of studying a traditional machine learning model development pipeline to segment it into modules and automate each of those to accelerate workflow(Chauhan et al., 2020). According to Truong et al. (2019), there are three main stages in an AutoML pipeline: (*i*) data preprocessing and feature selection; (*ii*) model selection, hyperparameter optimization, and architecture search; and (*iii*) the model interpretation and prediction analysis. The different steps are usually interactively optimized, meaning that the chosen preprocessing may interfere in the model selection step, influencing the model interpretation. The data preprocessing and feature selection is one of the most time-consuming steps in ML pipelines, Munson (2012) estimates that 50% to 80% of the data mining time employed here. In addition, this step influences directly the ML performance (Bilalli et al., 2018). Thus, this aspect is important to AutoML but not always present in the tools (Truong et al., 2019).

Although several AutoML tools are available, we selected three open-source ones in this work: Auto-Sklearn (Feurer et al., 2020), Tree-based Pipeline Optimization Tool (TPOT) (Olson and Moore, 2019), and Auto-Keras (Jin et al., 2019). Each one performs the classification step with a different approach, including meta-learning, genetic programming, and architecture search. We will briefly describe them to provide a better understanding of how the tools work.

Auto-Sklearn

Developed at the University of Freiburg, Germany, in 2014, Auto-Sklearn was build based on the Python library Scikit-learn (Pedregosa et al., 2011). This project sees the AutoML as the Combined Model Selection, and Hyperparameter optimization (CASH) problem (Feurer et al., 2015). A complete solution for the CASH problem tries to find the best ML pipeline for raw data in the shortest time possible (Tuggener et al., 2019). The tool first calculates a set of meta-features from a dataset provided by the user to be used as the model selection to perform the task. The Auto-Sklearn will then extract meta-features and apply an ML framework where it uses the Bayesian optimization to adjust the hyperparameters of the selected algorithm. Finally, it ensembles the used algorithms in the process. Thus, instead of discarding the computation needed to perform the hyperparameter optimization, it will enhance the result and avoid overfitting. Recently, Version 2.0 was released, which provides significant improvements in the model selection, changes the meta-feature approach to a portfolio (a set of configurations that covers diverse datasets), and automated the policy selection of the AutoML. Those combinations improved the speed and performance compared to the previous version (Feurer et al., 2020).

TPOT

Developed in 2015 at the University of Pennsylvania, USA, it uses Scikit-learn to implement the ML algorithms as Auto-Sklearn. TPOT includes a variety of ML pipelines, including preprocessing, feature selection, and model selection. Unlike the other approaches, this solution treats each step in the pipeline as an operator. The tool treats each operator as a primitive of the generic programming (GP) and uses them to create

GP trees (Olson and Moore, 2019). The TPOT is a flexible system being able to generate complex and straightforward ML pipelines.

Auto-Keras

Built at the University of Texas A&M, USA, in 2017 and perhaps is the most different of the previous solutions. Auto-Keras was implemented over the Keras and Tensorflow library, aiming to find the best neural network for each specific problem. Auto-Keras does not implement the preprocessing steps of the AutoML pipeline. To create the model, it first performs an architectural search. When founding a possible architecture, the AutoML will implement the neural network as a graph and train it. Auto-Keras will repeat this process until a good solution is found (Jin et al., 2019).

2.3.4 Performance measures

According to Tharwat (2018), several applications in the most varied fields of science uses classification algorithms. As such, there are several ways to evaluate these algorithms. These include accuracy, recall, false-positive rate, false-negative rate, precision, area under the receiver operating characteristic curve (AUC), F1-score. Furthermore, the results of learning algorithms need to be carefully evaluated and analyzed. The analysis must be correctly interpreted to assess the performance of different learning algorithms.

Through the literature review carried out (Chapter 3), we found the most used evaluation metrics in the works for the classification of bots on Twitter. As a result, we found that the most used metrics are accuracy, recall, and F1-score. These metrics were adopted in this work, aiming to facilitate the comparison of results with other work. With the same objective, AUC was incorporated into the set of evaluation metrics, as it is the primary metric adopted by the state-of-the-art base work (Rodríguez-Ruiz et al., 2020).

According to Sokolova et al. (2006); Tharwat (2018), accuracy is one of the most used measures to measure classification performance. Accuracy calculates the percentage of correctly classified instances as the ratio of correctly classified samples to the total number of samples. Equation 2.6 presents the calculation, where TP is the true-positive rate, TN is the true-negative rate, FP is the false-positive rate, and FN is the false-negative rate.

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

According to Powers (2011); Sokolova et al. (2006), recall (also known as sensitivity) is the ratio between correctly ranked examples (true-positives - TP) and incorrectly ranked examples (false-negatives - FN), as described in Equation 2.7. The recall is the proportion of true-positive cases correctly predicted.

$$\text{recall} = \frac{TP}{TP + FN} \tag{2.7}$$

According to Tharwat (2018), F1-score represents the harmonic mean between precision (Equation 2.8) and recall (Equation 2.7), as described in equation 2.9 . The value of the F1-score ranges from zero to one, where values closer to one indicate high ranking performance. According to Sokolova et al. (2006), this measure tends to benefit algorithms with higher sensitivity and is more challenging for algorithms with higher specificity.

$$\text{precision} = \frac{TP}{FP + FP} \tag{2.8}$$

$$\text{F1-score} = 2 * \frac{\text{precision} + \text{recall}}{\text{precision} * \text{recall}} \tag{2.9}$$

In Tharwat (2018), a two-dimensional graph describes the receiver operating characteristic curve (ROC). Where recall (sensitivity) represents the y axis, and the false-positives rate, or inverse of specificity, (Equation 2.10) describes the axis x. The false-positive rate measures the balance between benefits (true-negative, TN) and costs (false-positive, FP).

$$\text{false-positive rate} = \frac{FP}{FP + TN} \tag{2.10}$$

Furthermore, in Tharwat (2018), comparing different classifiers using the ROC curve is not easy because no scalar value represents the expected performance. Therefore, the area under the curve (AUC) is a numeric metric used to represent the ROC. The AUC score is always bounded between zero and one, where values closer to one represent better results. Considering that there is no perfect classifier ($AUC = 1$), realistic values are said to have $AUC > 0.5$, as lower values would be worse than a random choice. Figure 2.7, qdapted from `http://mbaskills.in/what-is-auc-roc-curve/`, illustrates the concepts of AUC (in grey), ROC (in green), and the dashed line represents a hypothetical ROC with AUC = 0.5.

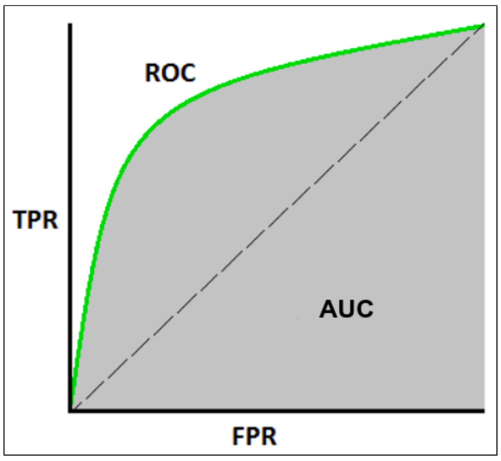


Figure (2.7)   ROC and AUC illustration.

Chapter 3

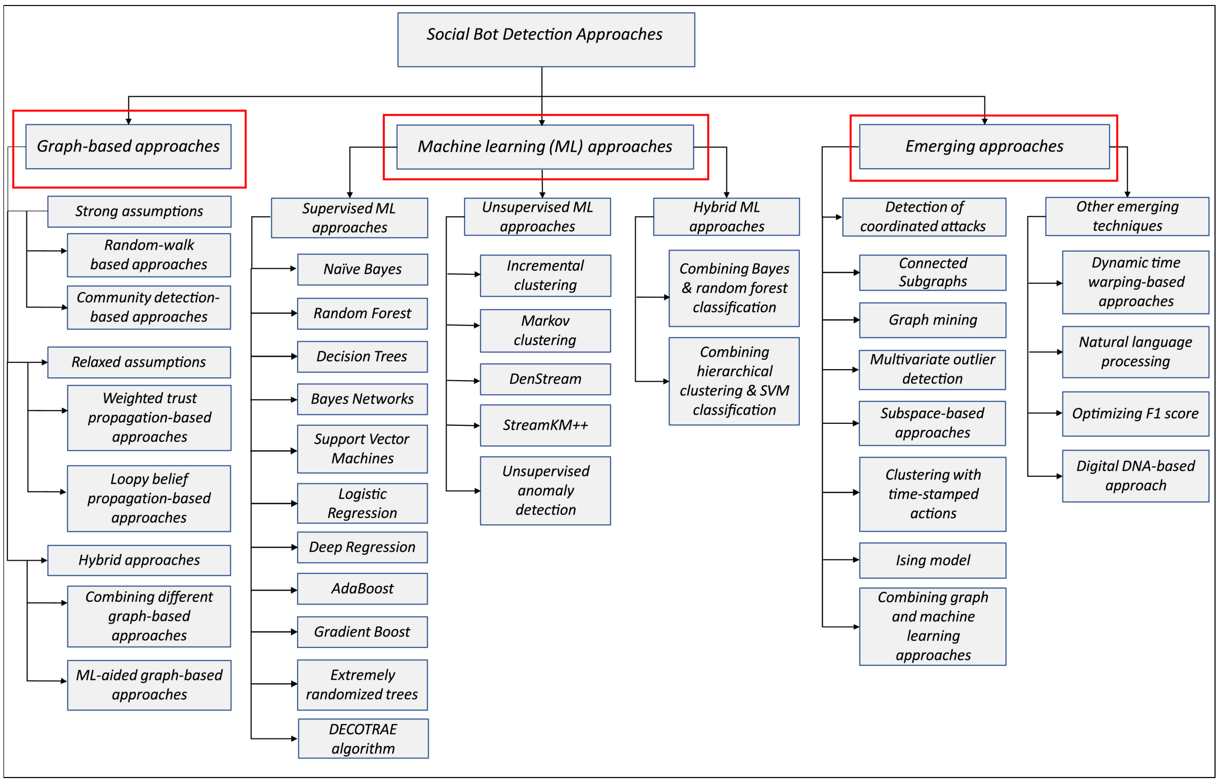
Literature Review

This section presents the literature review results related to the theme of Twitter bot detection. The reviews conducted by Latah (2020) were taken as a starting point, all works referenced were read. In addition to these, a search was performed in databases Scopus, Scielo, Web of Science, and Google scholar, aiming to update the review. The review occurred between April and May of 2020 and found 23 different works. This chapter discusses the works read in the literature review. After the discussion, is presented a comparison between the works considered most related to the proposal through Table 3.1. The works in this table used real datasets extracted from Twitter to validate the proposed classification models. Moreover, in the appendix A exists a table that summarizes all works studied. The review was conducted through three main research questions (Q) to detect the main techniques applied:

- Q1: What are the techniques presented in the literature for Twitter bot detection?
- Q2: What are the main ML methods and the most used algorithms for Twitter bot detection?
- Q3: What are the best ML algorithms employed in Twitter bot detection, considering the accuracy results?

Published literature reviews are interesting for the identification of related work. In this sense, Alothali et al. (2018) presents a wide range of research in the field of bot detection with three classes of work: (i) graph-based approaches; (ii) crowdsourcing - uses groups of humans collaborating to classify the profiles manually; and (iii) AI approaches.

Another review by Latah (2020) was found, which involves research results with social benefits without a focus on Twitter, but presents results with more sophisticated, capable bots of hiding the detection mechanisms. The author identified similar detection areas of Alothali et al. (2018) but replaced the category of crowdsourcing with emerging approaches to encompass techniques that do not fit into the other two categories. Figure 3.1 presents a taxonomy proposed by Latah (2020), including different classes and subclasses presented in the article. The main groups of approaches are marked with a red line. On the taxonomy, the Graph-based approaches use social graphs to detect bots, focusing on detecting fake accounts. The ML-based are those who apply an ML technique to perform the bot detection. Furthermore, the emerging approaches are those proposed for detecting coordinated social bot attacks and enhancing the previous detection approaches. All works discussed in this chapter of the literature review belong to the ML-based class on the Latah (2020) taxonomy. The following paragraphs contain a discussion of the studied works.


Social Bot Detection Approaches
Graph-based approaches
Machine learning (ML) approaches
Emerging approaches
Strong assumptions
Random-walk based approaches
Community detection-based approaches
Relaxed assumptions
Weighted trust propagation-based approaches
Loopy belief propagation-based approaches
Hybrid approaches
Combining different graph-based approaches
ML-aided graph-based approaches
Supervised ML approaches
Naïve Bayes
Random Forest
Decision Trees
Bayes Networks
Support Vector Machines
Logistic Regression
Deep Regression
AdaBoost
Gradient Boost
Extremely randomized trees
DECOTRAE algorithm
Unsupervised ML approaches
Incremental clustering
Markov clustering
DenStream
StreamKM++
Unsupervised anomaly detection
Hybrid ML approaches
Combining Bayes & random forest classification
Combining hierarchical clustering & SVM classification
Detection of coordinated attacks
Connected Subgraphs
Graph mining
Multivariate outlier detection
Subspace-based approaches
Clustering with time-stamped actions
Ising model
Combining graph and machine learning approaches
Other emerging techniques
Dynamic time warping-based approaches
Natural language processing
Optimizing F1 score
Digital DNA-based approach
Figure (3.1) Taxonomy proposed by (Latah, 2020).

The work of Lee et al. (2011) focuses on composing a bot-only dataset. According to the actors, the name of the technique used is social honeypots. This technique consists of Twitter profiles that do not post anything attractive to human users, only attracting bot interactions. This article uses this technique to build a unique database of accounts that are bots on Twitter. After that, a clustering algorithm (Expectation-Maximization (EM)) was run on the obtained database. The result was the discovery of 4 classes of bots (Duplicate Spammers, Duplicate @ Spammers, Malicious Promoters, Friend Infiltrators). The researchers also trained a classifier to identify which accounts were bots. Among 30 classifiers tested, the best approach was RF with 98.42% accuracy and 0.984 F1-score. It was possible to leverage the results using this technique using boosting (with an increase in accuracy to 98.62% and F1-score to 0.986) or bagging (increased accuracy to 98.57 % and from F1-score to 0.986). The classifier using RF and boosting resulted in higher accuracy. The main contribution of this work is the creation of a technique to assemble a dataset of Twitter users who are bots.

In Mazza et al. (2019), a classifier for the detection of bots whose main activity is focused on retweeting is presented. The proposed methodology consisted of taking all the retweets performed during a time window chosen by the researchers and analyzing the pattern of retweets for each account that performed this action during the period. A data visualization technique was developed, called ReTweet-Tweet (RTT), which found four retweet patterns, one from human users and three from bots. The same classes that the visualization had defined were found when designing the Retweet-Buster (RTBust) classifier through the unsupervised Long Short-Term Memory (LSTM) algorithm. With few features they obtained a F1-score $= 0.87$, higher than all the works used for the comparison Cresci et al. (2015); Liu et al. (2019); Varol et al. (2017) (all with F1-score) ≤ 0.76). Also, the RTBust found two completely new botnets (they were not in the dataset used). Following the line of unsupervised algorithms, we also have the works of Chen and Subramanian (2018); Miller et al. (2014) that perform a real-time clustering of profiles considered malicious, and Al-Qurishi et al. (2018); Minnich et al. (2017) whose approaches can improve their respective classifiers with new data extracted in real-time.

Wang (2010) deals with bot classification work with a focus on spambots. It collects the data through two different methods: through the Twitter API and with the aid of a web crawler that searched the 20 most recent profile tweets that did not authorize the extraction through the API. The final dataset had 25,847 profiles, around 500,000 tweets, and around 49 million relationships between profiles. A set of 500 profiles was used to validate the models, which the researchers manually annotated in two categories: spam and not spam. Data from the platform were also collected that were marked by other users as spam (tweets mentioning the profile @spam and the reported account). After all the effort, the validation set had about 3% spam profiles. The classifiers DT, Neural Networks (NN), SVM, and NB, based on attributes based on the social graph and the content of the tweets. The classifier using NB had a better overall performance with recall $= 0.917$ (other techniques obtained ≤ 0.417) and F1-score $= 0.917$ (other classifiers obtained ≤ 0.588). NN and SVM achieved precision $= 1$ (versus 0.917 for NB) but were worse in F1-score precision and recall.

Yang et al. (2013) also computes graph-based features, and these are among the most robust according to the authors' evaluation. Still following the idea of the social graph but avoiding the high computational cost involved in its construction/analysis, the authors Fazil and Abulaish (2018) propose the so-called features based on the community. These consist of analyzing the followers of a profile, as it is an attribute that the account has very little control over. The use of this type of feature proved to be effective for the detection of bots.

Chu et al. (2012) develops a classifier that separates profiles into three classes: human, bot, or cyborg. The cyborg class is defined by either a human profile assisted by bots or a bot profile that a human assists. An interesting discussion about the dilemma in the classification of bots by Twitter was presented. Considering that bots help maintain the platform based on posting valuable and helpful content to users. The classifier here is composed of four parts: a component that measures the entropy that detects the pattern of the temporal behavior of a given profile on Twitter. This component detects spam (checks the content of tweets), a component that extracts some properties from the profile (such as the number of URLs posted), and the decision-making module. The study involved 500,000 different profiles, and this framework achieved a true-positive rate of 96%, demonstrating that this method is applicable for detecting bots. In Dickerson et al. (2014) there is evidence that feeling analysis can improve classifiers, as humans express feelings in a more complex way than bots.

In Stringhini et al. (2010), the authors used the honey profiles technique (same social honeypots technique as Lee et al. (2011)) to extract data about bots in three different social networks: Twitter, Facebook, and Myspace. The researchers defined six features for the classifier: rate FF (in the context of the Twitter ratio between followers and followed by the profile), rate of URLs, the similarity between messages, criteria of choice of friends, number of messages sent, and number of friends (in Twitter the number of profiles followed was considered). The criteria for choosing friends consisted of checking if the bot uses a list of names to choose the accounts followed. This feature was not used on Twitter because it had very similar values for human profiles and bots, suggesting that this strategy is not used on the platform. In Twitter, cross-validation estimated the rate of false positives at 2.5% and false negatives at 3% in the training set with 500 profiles. After that, the classifier was tested in real-time, reporting to Twitter the identified bots. Around 75% of the reported profiles were banned.

In Tavares and Faisal (2013) there is the development of work focused on temporal features of actions performed in the OSN. The researchers developed an application that collects information from Twitter through the Twitter API. This application and the collected data are available in a public repository on GitHub[1]. In this article, profiles are grouped into three classes: personal, company-managed, and bots. Two different classi-

[1]the authors declare that the link is http://www.faisallab.com/TRM

fiers were designed, one classifying the profiles into personal and managed by companies and another classifying them with all classes. The temporal attributes considered here were the tweet's timestamp and the time elapsed between 2 tweets. Four features were conceived based on the attributes: marginal distribution of the delay among tweets, the marginal distribution of the tweets timestamps, joint distribution of the two attributes assuming independence, and joint distribution of the two attributes not assuming independence. The first classifier classifies profiles between human and company-administered profiles with 83.1% correctness. The second categorizes profiles within the three proposed classes with a correctness of 73.1%.

In Al-Qurishi et al. (2018); Gao et al. (2012) although it is not the principal focus of research, there is also analysis of temporal features, which serves to demonstrate that this device can be used in detection with reasonable performance measures. Oentaryo et al. (2016) developed with the primary objective of analyzing beneficial and harmful bots. The authors mention that most works in this area deal only with malicious bots. They categorize bots into three classes: transmission, consumption, and spam. With a very diverse set of features, classifiers were tested using four different ML techniques NB, RF, SVM, and Logistic Regression (LR). The researchers found behavior patterns of the different types of bots that were the research targets. As a result, they obtained a classifier that is better than a random choice. However, it is less efficient than other works that focus only on malicious bots. Meanwhile, in Ahmed and Abulaish (2013) there is an attempt to design features that serve for classification in both Twitter and Facebook.

In Rodríguez-Ruiz et al. (2020) the idea of classifiers of one-class (binary) trained only with data from legitimate users, aiming to obtain a tool capable of detecting any bot. Ten different algorithms were tested: BayesianNetworks (BN), J48, RF, adaptive boosting (AdaBoost), Bagging, K-Nearest Neighbors (KNN), LR, multilayer perceptron (MLP), NB, and SVM. According to the authors, these one-class algorithms were chosen because they are the most used in the works found in the literature review carried out by them. The one-class algorithms tested were BaggingTPMiner (BTPM), BaggingRandomMiner (BRM), OneClass K-means with randomly designed feature algorithm (OCKRA), SVM of a class (ocSVM), and NB. The classifiers were trained and tested offline with the datasets public from the works of (Cresci et al., 2017; Yang et al., 2013). As a result, there is a consistent detection of different types without requiring any prior information reaching $AUC > 0.89$.

Alharthi et al. (2019) maps the spread groups of Arab spam. These accounts are responsible for spreading unwanted content (among them misinformation), which lowers the quality of RSO content. The API of Twitter was employed to collect the data. Then semi-supervised classification algorithms were used: Labelpropagation (LP) and

Labelspreading (LS), which were trained and tested offline with collected data, using 16 features. The article does not mention whether the datasets used are public. Both algorithms achieved $AUC = 0.9$, 88% precision, 91% recall, 88% specificity, and 91% accuracy. In addition, the work showed that profiles that disseminate spam have similar behavior to botnets, organizing themselves into groups, where each profile has a specific activity to perform.

In Beğenilmiş and Uskudarli (2018) an organized behavior classifier was built-in Twitter. The period chosen to collect the tweets was the 2016 US elections. The authors believe that there occurred a high number of bots and fake news dissemination. They trained three different classifiers (RF, LR, and SVM) and tested them to classify accounts between organized or organic, political or non-political, pro-Trump, or pro-Hillary. The RF algorithm showed the best performance with average accuracy and F1-score greater than 95% in each category. The source code and the datasets used are available in a public repository at Github[2]. The main contribution of this work is the conception of a basic model for the detection of organized behavior in Twitter.

The work of Fernquist et al. (2018) occurred during Sweden's general election and intended to classify Twitter bots. The researchers trained a classifier with the RF algorithm and tested it offline to filter which bots produced tweets during the experimentation period. Three different public datasets were used, combined with 140 account attributes: (Cresci et al., 2015; Gilani et al., 2017; Varol et al., 2017). The main objective of the work is not to design a new or more efficient bot detection technique, but it designed a good classifier using ML techniques. The classifier result was compared with five other works: (Ahmed and Abulaish, 2013; Cresci et al., 2016; Davis et al., 2016; Miller et al., 2014; Yang et al., 2013). The results showed a recall rate of 0.976, which was higher than all five works compared (the second-highest was Cresci et al. (2016) with 0.972).

In Varol et al. (2017) a report is presented about a classifier 'bot or not' (currently, this classifier is known as botometer). The RF classifier with more than a thousand different features was used to perform the classification in this work. The researchers used two datasets: a dataset containing only bots obtained through a honeypot (article Lee et al. (2011)) and a dataset manually annotated by them. The article dataset is a mix of the previously cited datasets, and it is available to the community. Researchers also maintain a web tool that can classify any profile on Twitter in real-time. AUC gives the implementation accuracy measure here with a score of 0.95. It is possible to note that the user and content metadata of the Tweet are the most important attributes to find out if the account is a bot. Furthermore, this demonstrates that software-controlled accounts can be grouped according to their intentions or *modus Operandi.*

[2]`https://github.com/Meddre5911/DirenajToolkitService`

At Sayyadiharikandeh et al. (2020) occurs an improvement of Varol et al. (2017) by the usage of an ensemble of specialized classifiers (ESC). The ESC can better generalize, leading to an average improvement of 56% in F1-score for unseen accounts across datasets. Furthermore, novel bot behaviors are learned with fewer labeled examples during retraining. The new technique was deployed in the newest version of Botometer, a popular tool to detect social bots in the wild, with a cross-validation AUC of 0.99. Each specialized classifier utilizes RF, and the ensemble can detect bots generally achieving good cross-domain performance. The authors affirm that the proposed approach generalizes better than a monolithic classifier and is more robust to mislabeled training examples.

Synthesis of the literature review

Considering the literature review, the research questions were answered as follows:

- Q1: What are the techniques presented in the literature for Twitter bot detection? According to Alothali et al. (2018), the main approaches are based on graph algorithms, crowdsourcing and AI algorithms. According to Latah (2020), the categories include graphs, ML, and emerging approaches.
- Q2: What are the main ML methods and the most used algorithms for Twitter bot detection? The main ML methods and algorithms used to detect bots on Twitter include the supervised ML algorithms RF, SVM, and NB, as shown in Figure 3.2.
- Q3: What are the best ML algorithms employed in Twitter bot detection, considering the accuracy results? RF with 98% (Lee et al., 2011) and 96% (Al-Qurishi et al., 2018); DenStream and StreamKM++ with 97% (Miller et al., 2014), Iterative regression with 96% (Al-Qurishi et al., 2018).

This chapter reported the literature review performed to assist the proposal design. The review was conducted based on three research questions about bot detection. The works read were resumed on Table 3.1 and expanded on Table A.1 (Appendix A). Chapter 4 contains a description of the proposed solution.

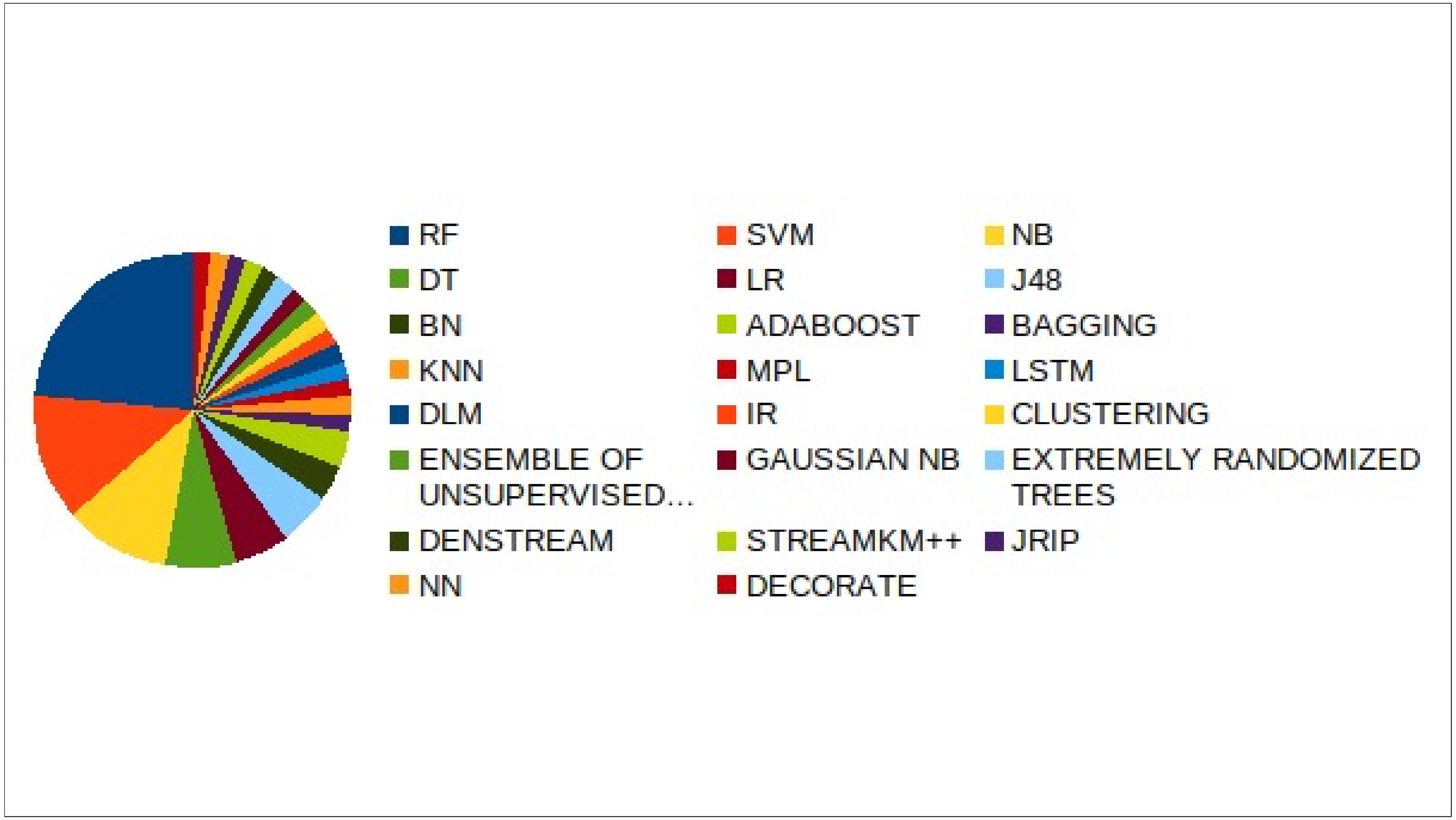


Figure (3.2) Main ML methods and algorithms from related work.

Table (3.1) Comparison between the main related work found on the literature review.

Reference	**ML algorithm**	**Application domain**	**Methodology**	**Dataset**
Rodríguez-Ruiz et al. (2020)	BN, J48, RF, Adaboost, Bagging, KNN, LR, MLP, NB, SVM, BTPM, BRM, OCKRA, ocSVM	general	offline classification with supervised method	public
Sayyadiharikandeh et al. (2020)	ensemble of RF	general	online supervised classification	public
Alharthi et al. (2019)	LP, LS	spam	offline classification with semi-supervised method	not informed
Beğenilmiş and Uskudarli (2018)	RF, LR, SVM	election campaign	offline classification with supervised method	public
Fernquist et al. (2018)	RF	election campaign	offline classification with supervised method	public
Varol et al. (2017)	RF	general	offline classification with supervised method, and classifier execution with online data	public

Chapter 4

Proposal description

The main goal of the research is to develop a MAS that automatically detects bots for Twitter. As discussed in Section 1.1 of Chapter 1, this approach has desirable attributes for solving this kind of problem. For this purpose, the ABS was developed, a MAS capable of detecting bots on an online stream of tweets. For this, three different agents were developed that cooperatively interact with them and the environment. The agents capture a tweet stream based on pre-established keywords and decide online if any author is a bot. The Twitter REST API provides the tweet stream. The decision consists of a combination of supervised ML techniques, including RF, NB, and SVM. These are the most used algorithms, according to the literature review. The decision was to use the most common algorithms to compare the results with the existing algorithms in the literature.

The MAS design used the Tropos methodology. The choice of this methodology is motivated mainly by the fact that it covers several different phases of the MAS project, which allows a greater understanding of the system and a more refined set of requirements. This section reports the development stages performed during the proposal development. The ABS performance was measured offline using the same method used in Fonseca Abreu et al. (2020)). For this, the tweets stream was replaced by the dataset provided by Cresci et al. (2017), including several files in the CSV format containing information related to user profiles and tweets. Each file contains annotations with the type of profile produced in the present data. The profiles adopted here are Genuine accounts, Social spambots#1, and Traditional spambots#1.

4.1 Design model

In the pre-project phase of the MAS design, we defined the PEAS (see Section 2.2). In this work, the environment that the ABS will interact is Twitter. It consists of a microblog where users post short messages (tweets) in their respective profiles (Fazil and Abulaish, 2018). According to Twitter Inc. (2020b), the tweets are up to 280 text characters long and may contain references to other profiles by typing @ and the username to be mentioned. Other multimedia elements are also possible, such as videos, images, surveys, geolocation, among others (Gui et al., 2019). Thus, the environment shared by agents in the ABS can be defined as:

- partially observable: impossible to view the complete set of all tweets;
- non-deterministic: it is not possible to determine the contents of the tweet in advance;
- episodic: tweets can be treated loosely connected in episodes;
- dynamic: new tweets may appear at any time;
- continuous: tweets may appear in a continuous spectrum of values; and
- multi-agent: more than one agent can perceive and act in the environment.

We defined three types of agents in the ABS. The first type is the collector, which is responsible for identifying the profiles making tweets according to a pre-established domain, saving the respective data in a monitoring database. As soon as data is identified, the collector forwards it to the classifying agents. In turn, the classifiers perform the classification of the profiles and send the results to the referee. The training of the classifiers is carried out offline, making its use faster during the execution of the system. Each classifying agent is implemented with a different ML algorithm to classify the same profile into different classes. The researchers based the choice of the approach using supervised ML algorithms on the work Latah (2020), where it was possible to glimpse the most widely used algorithms for the Twiter bot application domain.

According to the results of the classifying agents, the referee agent is responsible for making the final profile classification through a simple majority vote. In other words, the referee joins the decisions resulting in an ensemble learning approach and records these decisions in a comma-separated values (CSV) file. The experiment is only interrupted by the researchers' decision. The PEAS of the three agent types is available in Table 4.1. Since all agents share the environment, it is not included in the PEAS table.

During the MAS design project, we built four different models referring to development phases. The models designed include the early requirements, late requirements, architectural design, and detailed design. Due to the adoption of the Tropos Bresciani et al. (2004) methodology, some models were built using the language i* (iStar) Dalpiaz et al. (2016), and the online tool piStar implemented by (Pimentel and Castro, 2018).

Table (4.1) PEAS for the three agents: collector, classifier and referee.

Agent Type	**Performance Measure**	**Actuators**	**Sensors**
Colletor	Number of collected tweets	Monitor tweets according to key-words, collect information about tweets, record information in database, send data to classifiers.	Verifies the existence of new tweets in the tweet flow chosen
Classifier	Number of executed classifications	Receive tweet data from collectors, extract the necessary features, classify the user who posted the tweet.	Verifies the existence of new data from tweets collected by collectors
Referee	Number of accounts classified as bots	Count *votes* and decide about the classification.	Verifies the existence of new classification from classifier agents

iStar modeling

The early requirements diagram shown in Figure 4.1 identifies the domain stakeholders and defines them as actors: Twitter and ABS. It is the most simplistic diagram, and it addresses how dependencies and exchanges of resources between entities take place. It is possible to see that Twitter provides the flow of tweets, and the ABS produces the bot detection.

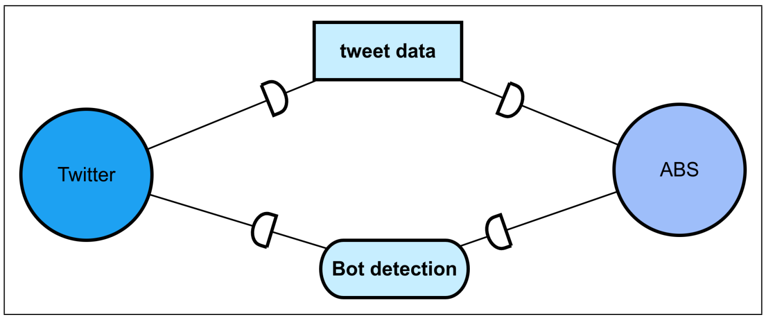


Figure (4.1) Early requirements diagram.

Regarding the late requirements diagram (Figure 4.2), it is possible to see an expanded model containing a higher detailing of the exchanges of resources between the actors. Because of these details, it is possible to see how the actors share messages and resources.

The ABS's architectural project diagram shows the relationships between actors, agents, tasks, resources, and objectives. In addition to all the actors involved in the environment, it is possible to observe the agents, their tasks (representing the functional requirements) and desires (non-functional requirements), and the dependencies and relationships to the other actors involved in the process. Figure 4.3 illustrates a version of the architectural project diagram, with the details of each agent collapsed.

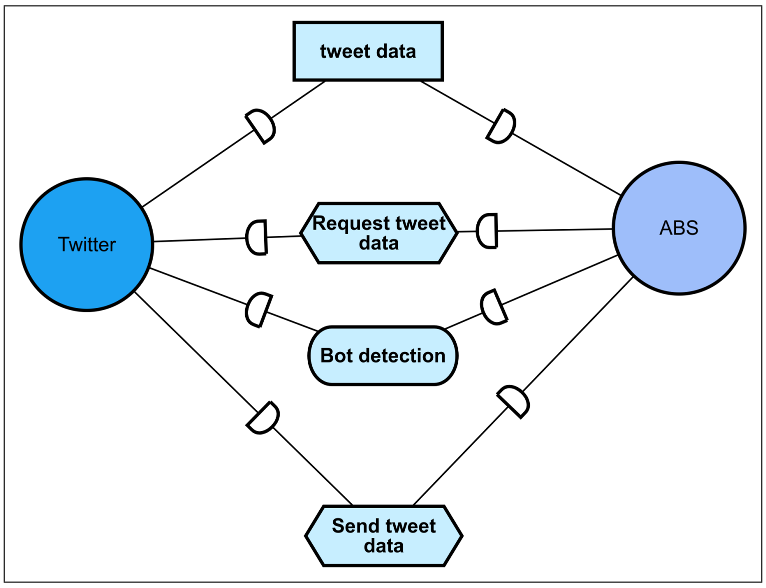


Figure (4.2) Late requirements diagram.

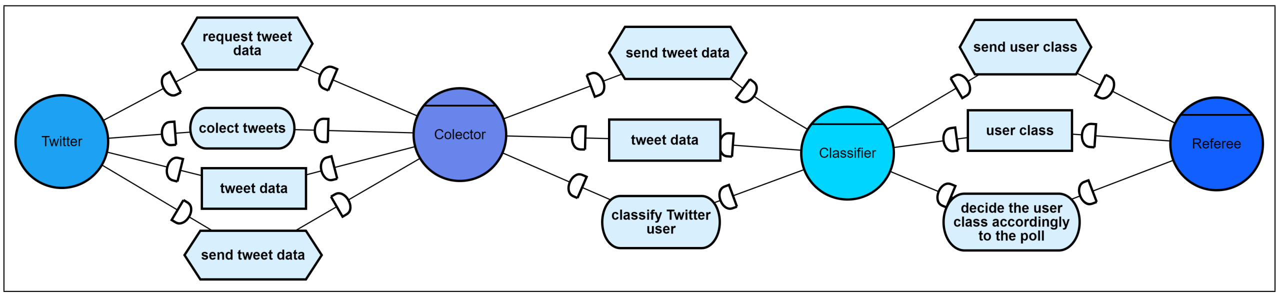


Figure (4.3) Architectural project diagram.

Detailed design

This phase aims to specify the resources and interactions between agents. It is considered that the implementation platform has already been chosen to allow for a better level of detail. For this, UML diagrams of sequence of events and activities were used. At this point, the detailing of the model will allow the mapping of diagrams directly to the code.

Figure 4.4 shows the exchange of messages between the System user, Twitter, and the ABS agents: collector, classifiers, and referee.

A diagram of activities was prepared for each agent: collector, classifiers, and referee (Figure 4.5). The diagram presents the flow of activities that each kind of agent must execute to conclude their respective objectives: Collect tweets - Collector; Classify the author of tweets - Classifier and Recommend Monitoring - Referee.

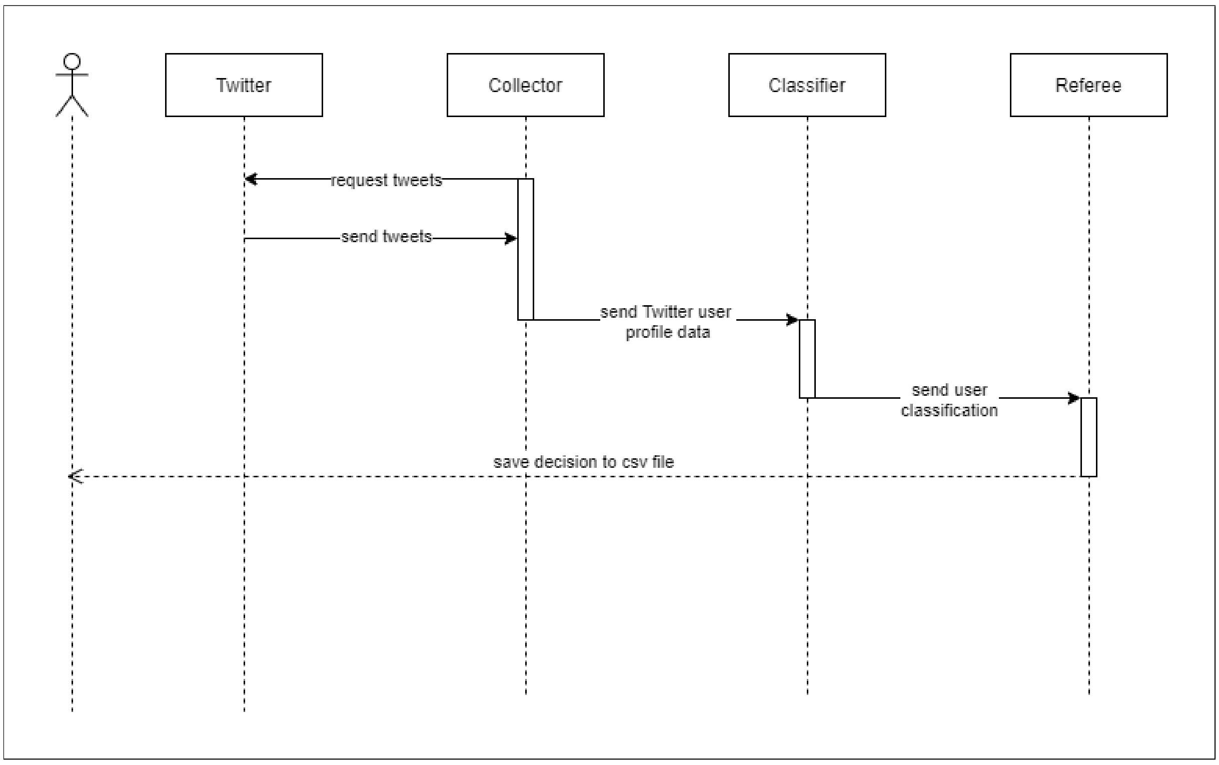


Figure (4.4) Diagram of sequence of events.

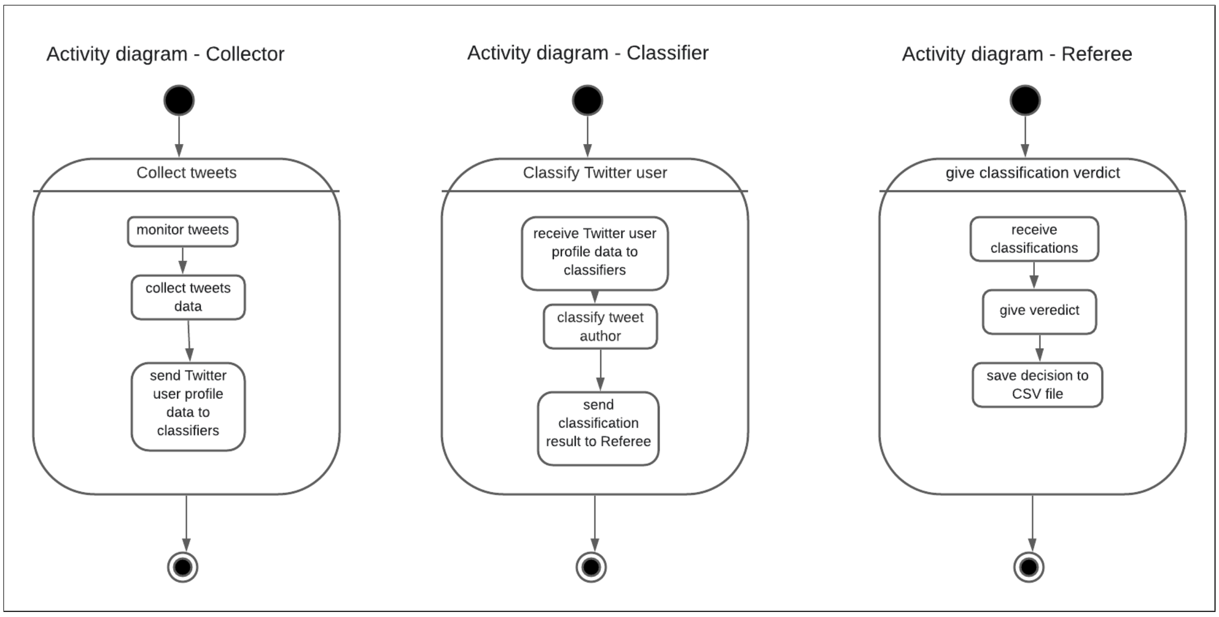

Activity diagram - Collector
Collect tweets
monitor tweets
collect tweets data
send Twitter user profile data to classifiers
Activity diagram - Classifier
Classify Twitter user
receive Twitter user profile data to classifiers
classify tweet author
send classification result to Referee
Activity diagram - Referee
give classification verdict
receive classifications
give veredict
save decision to CSV file

Figure (4.5) Diagram of activities.

4.2 Agent reasoning model

We defined three types of intelligent agents for the ABS. Each type has different properties and reasoning models. The reasoning model of each agent was chosen according to the actions that they would perform as presented in the PEAS definition of Table 4.1.

The first type of agent is the collector with the simple reflex reasoning model. This is the simplest type of reasoning without a sophisticated strategy, but the choice of action that the agent will perform is based on the perception-action mapping. We chose this reasoning model because the activity is trivial. The agent is responsible for accessing Twitter to collect tweets based on pre-defined keywords and passing the tweet data to the classifying agents.

We defined the classifier agent as an agent with learning. This agent is responsible for receiving data from the collector agent, applying a feature selection routine, and performing a data classification using an ML algorithm. This classification will be the agent's *vote* to assist the referee's decision. In the first ABS implementation, three classifying agents were built using the most common algorithms in this application domain: RF, SVM, and NB. The same classifiers previously used in Fonseca Abreu et al. (2020) intending to allow comparison with the ABS results.

Finally, the referee is a goal-based agent. This kind of agent considers the best action to achieve their objectives among a universe of possible actions to be taken. The referee is responsible for counting the classifier agents' *votes* referring to a Twitter profile. The referee's decision will allow reporting the profile and monitor whether Twitter bans the profile has.

4.3 Architecture

The ABS architecture is based on the reactive agent model that is horizontally layered as presented in (Mora Lizán and Rizo-Maestre, 7 05). The architecture layers connect the sensors (input) and the actuators (output) directly. There are three layers and m possible actions suggested by each layer resulting in m^3 possible interactions. The three layers are divided by well-defined activities, and each has homogeneous agents. The activities include tweets capture, the tweets' authors classification, and arbitration. In other words, the architecture is horizontalized distributed into competency modules. Note that the architecture is naturally prepared to accommodate the online detection since it establishes a direct pipeline for processing and classifying tweets as the agents capture them.

One of the facts supporting the architecture choice is the high fault tolerance inherent in the model, failures that can occur at any level. As the ABS aims to deal with a dynamic environment, this property is attractive. Besides, the layered scheme is a good choice when several agents with different capabilities interact in the same environment. A problem in this architectural model is the bottleneck introduced by the communication between ABS and the environment (Twitter) since there is a unique communication point (the collector agent). However, the positive points mentioned compensate for the problem cited, and the proposed architecture does not have many layers, which helps to smooth the bottleneck. Also, the proposed architecture does not have many layers, which helps to soften the bottleneck. In Figure 4.6 we present the proposed ABS architecture. It is possible to visualize the agent types and how they interact with each other.

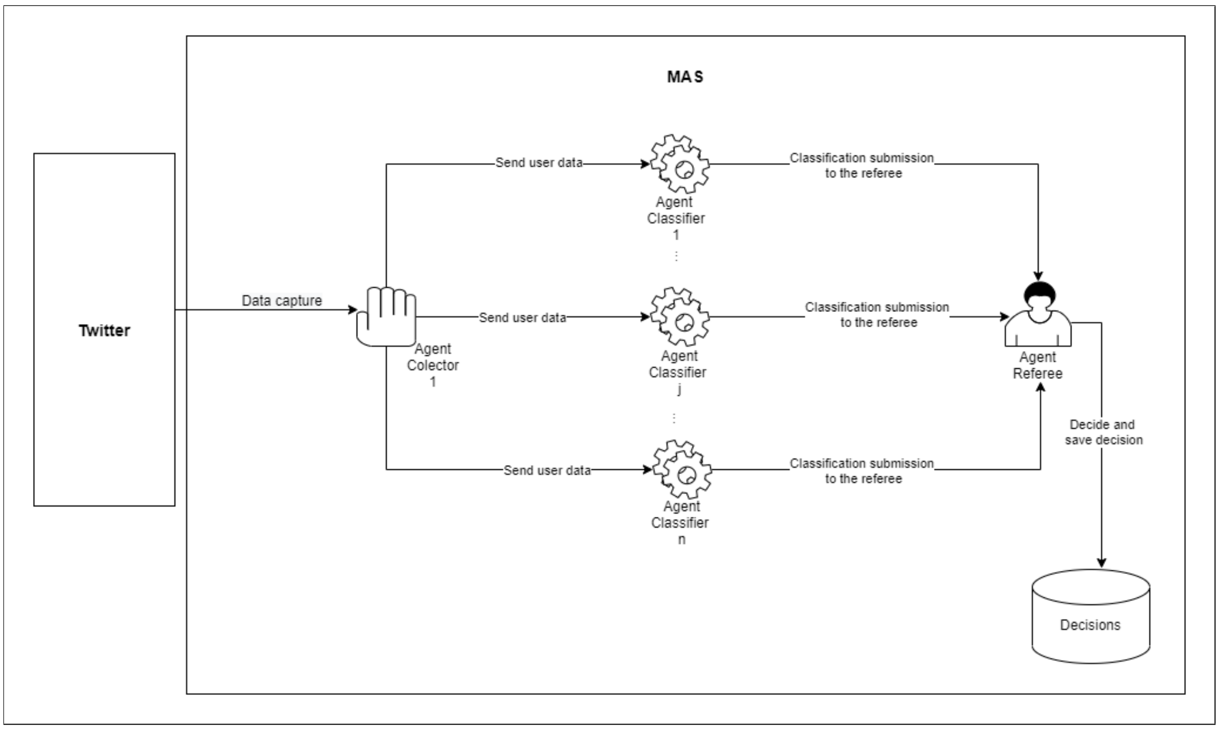


Figure (4.6) The ABS horizontally layered architecture.

4.4 Implementation

After the description of the artifacts, the implementation of the ABS was carried out. We used the language Python, version 3, and the framework for agent-oriented development called Python Agent DEvelopment framework (PADE) developed by (Melo et al., 2019). The source code is open source and is available in a public repository of the research group at GitLab [1]. The three proposed agents were developed, and the interactions between them occur using communication protocols standardized by the FIPA (O'Brien and Nicol, 1998).

[1] https://gitlab.com/InfoKnow/SocialNetwork/jeffersonabreu-twitterbotdetection

Agent comunication protocols

The communication between the collector and the classifier agents occurs through the FIPA Agent Communication Language (ACL) with an interaction protocol called subscribe (Karzan and Erdogan, 2013). In this protocol, the agent's subscribers perform a subscription to an agent publisher, and after that, all subscribers receive notifications sent by the publisher. The role of subscribers is performed by the classifiers agents, while the collector agent acts as a publisher. It is believed that this interaction protocol implements the desired behavior for communication between classifiers and the collector through a simple message exchange using the FIPA inform performative. At the Listing 4.1 exists the source code used by the collector for doing the distribution of the profiles info. Moreover, At the Listing 4.2 is the source code used by the classifiers for doing the subscription to the collector and informing the classification result to the referee.

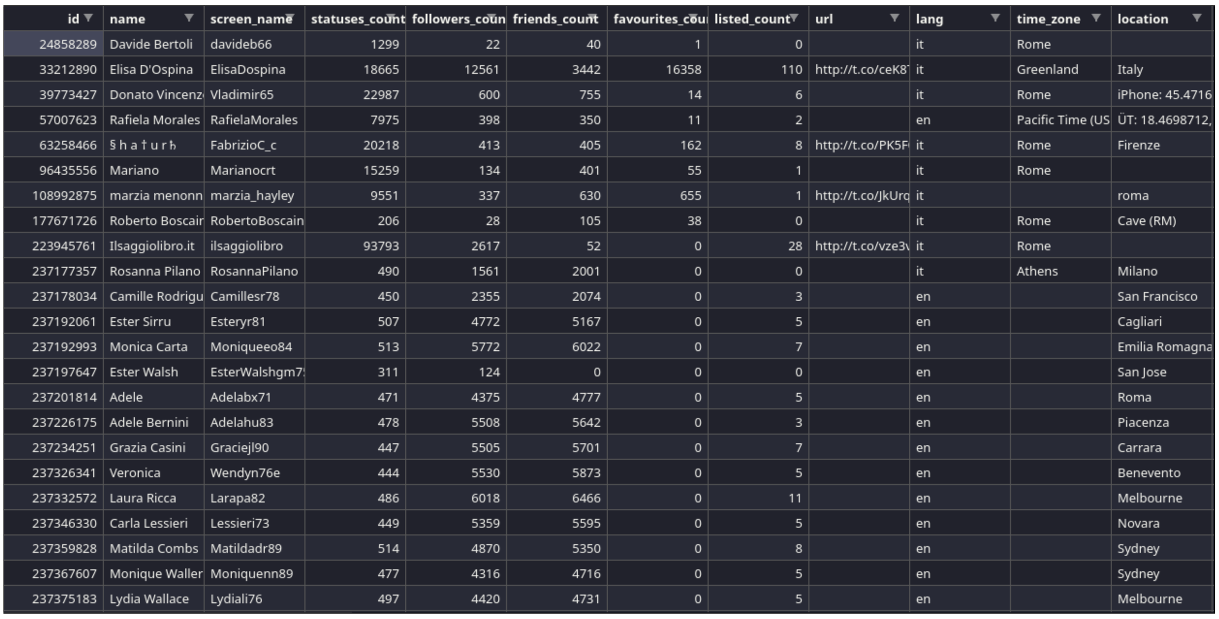
```python
class PublisherProtocolColector(FipaSubscribeProtocol):
    def __init__(self, agent):
        super(PublisherProtocolColector, self).__init__(
            agent, message=None, is_initiator=False)

    def handle_subscribe(self, message):
        self.register(message.sender)
        display_message(self.agent.aid.name, '{} from {}'.format(
            message.content, message.sender.name))
        resposta = message.create_reply()
        resposta.set_performative(ACLMessage.AGREE)
        resposta.set_content('Subscribe message accepted')
        self.agent.send(resposta)
        self.agent.registered = self.agent.registered + 1

        if self.agent.registered == 3:
            self.agent.timed.ready = True

    def handle_cancel(self, message):
        self.deregister(message.sender)
        display_message(self.agent.aid.name, message.content)

    def notify(self, message):
        super(PublisherProtocolColector, self).notify(message)
```

Listing (4.1) Example of the class used by the collector for doing the publishing of the profiles info

```python
class SubscriberProtocolClassifier(FipaSubscribeProtocol):

    def __init__(self, agent, message):
        super(SubscriberProtocolClassifier, self).__init__(
            agent, message, is_initiator=True)

    def handle_agree(self, message):
        display_message(self.agent.aid.name, str(
            message.content))

        message = ACLMessage(ACLMessage.INFORM)
        message.set_protocol(ACLMessage.FIPA_REQUEST_PROTOCOL)
        message.add_receiver(agent_colector.aid)
        message.set_content('ready')
        self.agent.send(message)

    def handle_inform(self, message):
        x = ast.literal_eval(str(message.content))
        classy = self.agent.classify(x['user']['id'], x['user']['statuses_count'], x['user']['followers_count'],
                                     x['user']['friends_count'], x['user']['favourites_count'], x['user']['listed_count'])

        display_message(self.agent.aid.name, 'class of ' +
                        str(x['user']['id']) + ' is ' + str(classy))

        message = ACLMessage(ACLMessage.INFORM)
        message.set_protocol(ACLMessage.FIPA_REQUEST_PROTOCOL)
        message.add_receiver(agent_referee.aid)
        message.set_content(str(x['user']['id']) +
                            '  ' + str(classy[0]) + '  ' + str(classy[1]) + '  ' + str(x['created_at']))
        self.agent.send(message)
```
Listing (4.2) Example of the class used by the classifiers for doing the subscription, and sending the class to the referee

Training and testing of classifiers

For training and testing the ABS, the dataset used is provided by (Cresci et al., 2017). It is composed of several files in the CSV format containing information related to user profiles and tweets. Each file contains the type of profile produced by the present data. Figure 4.7 shows a sample of data from the dataset Social1, with 12 out of 40 columns available on the user characteristics CSV file.



id	name	screen_name	statuses_count	followers_coun	friends_count	favourites_cou	listed_count	url	lang	time_zone	location
24858289	Davide Bertoli	davideb66	1299	22	40	1	0		it	Rome	
33212890	Elisa D'Ospina	ElisaDospina	18665	12561	3442	16358	110	http://t.co/ceK8	it	Greenland	Italy
39773427	Donato Vincenz	Vladimir65	22987	600	755	14	6		it	Rome	iPhone: 45.4716
57007623	Rafiela Morales	RafielaMorales	7975	398	350	11	2		en	Pacific Time (US	ÜT: 18.4698712,
63258466	§ h a † u r ♄	FabrizioC_c	20218	413	405	162	8	http://t.co/PK5F	it	Rome	Firenze
96435556	Mariano	Marianocrt	15259	134	401	55	1		it	Rome	
108992875	marzia menonn	marzia_hayley	9551	337	630	655	1	http://t.co/JkUrq	it		roma
177671726	Roberto Boscair	RobertoBoscain	206	28	105	38	0		it	Rome	Cave (RM)
223945761	Ilsaggiolibro.it	ilsaggiolibro	93793	2617	52	0	28	http://t.co/vze3v	it	Rome	
237177357	Rosanna Pilano	RosannaPilano	490	1561	2001	0	0		it	Athens	Milano
237178034	Camille Rodrigu	Camillesr78	450	2355	2074	0	3		en		San Francisco
237192061	Ester Sirru	Esteryr81	507	4772	5167	0	5		en		Cagliari
237192993	Monica Carta	Moniqueeo84	513	5772	6022	0	7		en		Emilia Romagna
237197647	Ester Walsh	EsterWalshgm7	311	124	0	0	0		en		San Jose
237201814	Adele	Adelabx71	471	4375	4777	0	5		en		Roma
237226175	Adele Bernini	Adelahu83	478	5508	5642	0	3		en		Piacenza
237234251	Grazia Casini	Graciejl90	447	5505	5701	0	7		en		Carrara
237326341	Veronica	Wendyn76e	444	5530	5873	0	5		en		Benevento
237332572	Laura Ricca	Larapa82	486	6018	6466	0	11		en		Melbourne
237346330	Carla Lessieri	Lessieri73	449	5359	5595	0	5		en		Novara
237359828	Matilda Combs	Matildadr89	514	4870	5350	0	8		en		Sydney
237367607	Monique Waller	Moniquenn89	477	4316	4716	0	5		en		Sydney
237375183	Lydia Wallace	Lydiali76	497	4420	4731	0	5		en		Melbourne

Figure (4.7) Sample of dataset Social1.

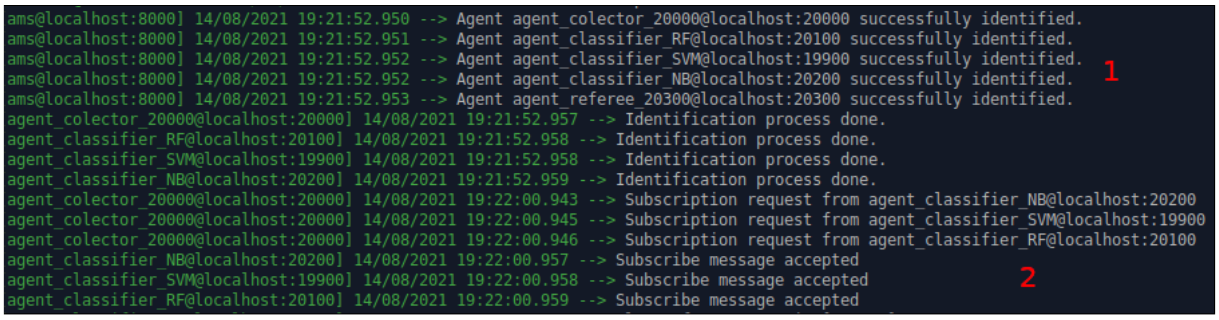
The types of profiles used in this work are the same as those of Fonseca Abreu et al. (2020), including:

- Social spambots#1 (Social1): includes accounts that retweeted the candidate for mayor of Rome in the 2014 Italian elections. One of the candidates hired a marketing company that used 991 bots to run the campaign.
- Genuine accounts: $3,474$ accounts were classified as legitimate using the following method: researchers asked a simple question in natural language for profiles chosen at random in the OSN. After this step, human beings analyzed the responses to define whether the account is legitimate.
- Traditional spambots#1 (Traditional): is the same dataset provided by Yang et al. (2013), which contains information on $1,000$ bots that have tweeted with malicious links.

The classification algorithms used are the same as those of Fonseca Abreu et al. (2020), trained with 90% of the dataset Genuine accounts combined with 90% of social spambots#1 xor Traditional spambots #1, serialized with the help of the pickle library (Foundation, 2021; Lutz, 2013). The serialization is essential to avoid training the three classifiers at every ABS execution. The features set adopted here is the same as Fonseca Abreu et al. (2020), it includes the number of tweets ($statuses\_count$), number of followers ($followers\_count$), number of friends ($friends\_count$), number of likes given by the account ($favorites\_count$), and number of lists that the account is included ($listed\_count$).

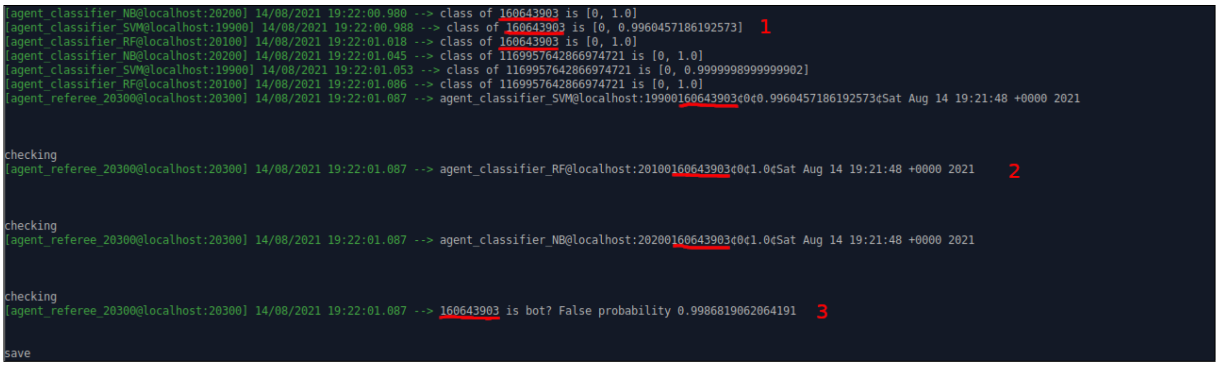
Execution

For illustrating the execution of the ABS the Figures 4.8 and 4.9 presents some logs from the system. Figure 4.8, is shown: (1) the moment when PADE identifies of all agents; (2) the subscription of the classifiers to the collector, and the respective acceptance. Figure 4.9, shows the classification process focusing on the user 160643903, where: (1) is the result of the classification from each classifier, represented by the tuple (class, probability of being human); (2) is the dispatch of the classification data to the referee; and (3) the decision of the referee.



```
ams@localhost:8000] 14/08/2021 19:21:52.950 --> Agent agent_colector_20000@localhost:20000 successfully identified.
ams@localhost:8000] 14/08/2021 19:21:52.951 --> Agent agent_classifier_RF@localhost:20100 successfully identified.
ams@localhost:8000] 14/08/2021 19:21:52.952 --> Agent agent_classifier_SVM@localhost:19900 successfully identified.
ams@localhost:8000] 14/08/2021 19:21:52.952 --> Agent agent_classifier_NB@localhost:20200 successfully identified.
ams@localhost:8000] 14/08/2021 19:21:52.953 --> Agent agent_referee_20300@localhost:20300 successfully identified.
agent_colector_20000@localhost:20000] 14/08/2021 19:21:52.957 --> Identification process done.
agent_classifier_RF@localhost:20100] 14/08/2021 19:21:52.958 --> Identification process done.
agent_classifier_SVM@localhost:19900] 14/08/2021 19:21:52.958 --> Identification process done.
agent_classifier_NB@localhost:20200] 14/08/2021 19:21:52.959 --> Identification process done.
agent_colector_20000@localhost:20000] 14/08/2021 19:22:00.943 --> Subscription request from agent_classifier_NB@localhost:20200
agent_colector_20000@localhost:20000] 14/08/2021 19:22:00.945 --> Subscription request from agent_classifier_SVM@localhost:19900
agent_colector_20000@localhost:20000] 14/08/2021 19:22:00.946 --> Subscription request from agent_classifier_RF@localhost:20100
agent_classifier_NB@localhost:20200] 14/08/2021 19:22:00.957 --> Subscribe message accepted
agent_classifier_SVM@localhost:19900] 14/08/2021 19:22:00.958 --> Subscribe message accepted
agent_classifier_RF@localhost:20100] 14/08/2021 19:22:00.959 --> Subscribe message accepted
```

Figure (4.8) Sample of ABS execution logs, containing identification and subscription



```
[agent_classifier_NB@localhost:20200] 14/08/2021 19:22:00.980 --> class of 160643903 is [0, 1.0]
[agent_classifier_SVM@localhost:19900] 14/08/2021 19:22:00.988 --> class of 160643903 is [0, 0.9960457186192573]
[agent_classifier_RF@localhost:20100] 14/08/2021 19:22:01.018 --> class of 160643903 is [0, 1.0]
[agent_classifier_NB@localhost:20200] 14/08/2021 19:22:01.045 --> class of 1169957642866974721 is [0, 1.0]
[agent_classifier_SVM@localhost:19900] 14/08/2021 19:22:01.053 --> class of 1169957642866974721 is [0, 0.9999998999999902]
[agent_classifier_RF@localhost:20100] 14/08/2021 19:22:01.086 --> class of 1169957642866974721 is [0, 1.0]
[agent_referee_20300@localhost:20300] 14/08/2021 19:22:01.087 --> agent_classifier_SVM@localhost:19900160643903¢0¢0.9960457186192573¢Sat Aug 14 19:21:48 +0000 2021



checking
[agent_referee_20300@localhost:20300] 14/08/2021 19:22:01.087 --> agent_classifier_RF@localhost:20100160643903¢0¢1.0¢Sat Aug 14 19:21:48 +0000 2021



checking
[agent_referee_20300@localhost:20300] 14/08/2021 19:22:01.087 --> agent_classifier_NB@localhost:20200160643903¢0¢1.0¢Sat Aug 14 19:21:48 +0000 2021



checking
[agent_referee_20300@localhost:20300] 14/08/2021 19:22:01.087 --> 160643903 is bot? False probability 0.9986819062064191


save
```

Figure (4.9) Sample of ABS execution logs, containing the classification process focusing on the user 160643903

This chapter presented a detailed description of the development phases executed on the conception of the ABS. The discussion presented supports the implementation decisions. Theoretically, the chosen architecture is adequate for the bot detection task. Therefore, the next chapter contains an empirical validation of the proposed solution.

Chapter 5

Empirical Validation

In order to validate the ABS proposal, three experiments were carried out, with the classifiers isolated and acting together on the MAS. The research methodology defined in this work is exploratory, experimental, and quantitative on Twitter bot detection, since the intention was to explore new features aiming to improve bot detection while developing a solution for performing this task autonomously. The experimental character is latent throughout the proposal during the various experiments presented.

Each experiment was designed to improve the tool built. However, the complete method can be divided into two stages: Definition and validation of the feature set adopted (i) and development and validation of the ABS (ii). The first stage contains two experiments, where the feature set used is defined and validated. In the feature set definition experiment (Section 5.1.1) occurs the development of the feature set called reduced feature set. As presented in the work Fonseca Abreu et al. (2020), this feature set was discovered using only previous experience with Twitter bots. Thus, the feature set validation experiment (Section 5.1.2) was necessary to validate the reduced feature set more extensively. The chosen approach was the validation through different AutoML feature selection mechanisms.

The second stage consists of the remaining experiment to develop and refine the MAS solution (ABS). In Experiment 3 (Section 5.2) the construction of the ABS takes place, including all agents involved in it, and tests. On these tests occurs the validation of the ABS with offline data and an online stream of tweets. Next, we present the experiments related to this research. In addition, the evaluation of results, along with related criteria will also be explained.

5.1 Feature Selection

In this section the feature set definition and the feature set validation are presented in Sections 5.1.1 and 5.1.2, respectively.

5.1.1 Feature set definition

An initial experiment was performed, where three versions of social bots were built and executed on Twitter (Abreu et al., 2019, 2020). The bots sent links to 1,287 profiles on the platform for 38 days. This experiment demonstrated that it was possible to use bots for malicious activities on OSNs. It was observed that the detection of bots from Twitter did not involve linguistic analysis but observed non-human behavior patterns in parameters such as frequency and continuity of tweets, duration of the activity period, and the like. It is a simple approach that aims not to affect user experience or to consume considerable computational resources. This approach does not conflict with the interests of the apparent desire to keep bots active (according to Chu et al. (2012)). However, despite this, these behavioral features can identify bots.

The decision was to use the most common ML algorithms to compare the results with the existing algorithms in the literature for the tests (see Chapter 3). In addition, aiming to apply one reduced feature set in the thesis proposed by Rodríguez-Ruiz et al. (2020), the one-class algorithm ocSVM was also tested. After an in-depth study of related works, we defined the experimental method as exploratory and quantitative. The intention is to explore new features aiming to improve the detection of bots in Twitter.

This experiment was based on the work of (Rodríguez-Ruiz et al., 2020). The dataset provided by Cresci et al. (2017) is used, including several files in the CSV format containing information related to user profiles and tweets. Each file contains annotations with the type of profile produced in the present data. The types of profiles used by Rodríguez-Ruiz et al. (2020) are: Genuine accounts, Social spambots#1, Social spambots#2, Social spambots#3 and Traditional spambots#1. Nevertheless, this experiment only employed the datasets that Rodríguez-Ruiz et al. (2020) achieved their best results. The details of the method are described in the sequence.

- **Experiment objective**: test the implementation of a bot classifier using a dataset with real-world data extracted from Twitter as the dataset used.
- **Dataset used**: the following CSV files from the dataset provided by Cresci et al. (2017) were chosen: Social spambots #1 - includes accounts that retweeted one candidate for mayor of Rome in the 2014 Italian elections, where these candidates hired a marketing company that used 991 bots to carry out the campaign. Genuine

accounts with 3474 accounts were classified as legitimate using the following method: the research group asked a simple question in natural language for randomly chosen profiles in the OSNs. After this step, humans analyzed the responses to define which accounts are legitimate. Traditional spambots #1 is the same dataset provided by Yang et al. (2013), which contains information on 1000 bots that made posts containing malicious links.

- **Dataset partitioning**: 90% of the data are used for training the classifier. The remaining 10% are used for testing the classifier (according to Rodríguez-Ruiz et al. (2020)).
- **Validation strategy**: We adopt the 10-fold cross-validation where the accuracy is measured ten times interleaving training and test sets (presented by Rodríguez-Ruiz et al. (2020)).
- **Feature selection**: A selection of five features (hereinafter denominated by reduced feature set) was made empirically based on prior experience with creating bots for Twitter Abreu et al. (2019, 2020), including the number of tweets (*statuses_count*), number of followers ($followers$*_count*), number of friends ($friends$*_count*), number of likes given by the account ($favorites$*_count*), and number of lists that the account is included (*listed_count*).
- **ML algorithms tested**: RF, SVM, NB and ocSVM implemented by (Pedregosa et al., 2011).
- **Results evaluation metric**: The entire database was measured through accuracy, recall, AUC, and F1-score. Average and standard deviation of both algorithms and datasets.

Figure 5.1 presents the experimental execution illustration which is divided into seven steps:

1. **Clean** useless columns;
2. **Divide** the data, with only the feature set $f_1, f_2, ..., f_5$, into training and testing datasets;
3. **Combine** the Training/Testing datasets;
4. **Train** the ML classifier with the training dataset (90% of genuine + 90% of bots);
5. Create the classification **model**;
6. **Test** the ML classifier with the testing dataset (10% of genuine + 10% of bots);
7. **Calculate** the metrics: Accuracy, AUC, Recall and F1-score.

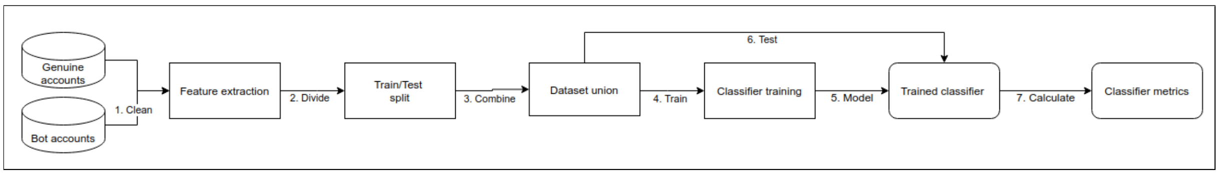


Figure (5.1) Experimental method illustration.

Result analysis

Experiment 1 was carried out with the reduced set of features. The results were quite similar compared to the results achieved with a more extensive set of features proposed by (Rodríguez-Ruiz et al., 2020). In addition, the reduced feature set uses simple features since they are common profile attributes, and the Twitter API returns them in one request. The feature set used in Rodríguez-Ruiz et al. (2020) is more complex regarding the need for additional requests to capture tweets. Examples of this type of feature are as follows: (i) $intertime$ - the average of seconds between posts made by the profile; (ii) $uniqueHashtags$ - the ratio between the number of unique hashtags used and the total number of tweets; and (iii) $uniqueURL$ - the ratio between the number of unique URLs used and the number of tweets; among other features. In addition, we used features that are simple counters related to profile activity.

Another essential point to mention is that even using a reduced set of features, it is possible to detect bots with a certain level of sophistication. It is possible to state this because the dataset social spambots #1 has several automated accounts similar to Genuine accounts in many aspects. All profiles are filled with detailed (false) personal information, such as photos, short bio, and location. Additionally, some of them could easily pass for reliable sources of information since they had thousands of followers and friends, the majority of which were legitimate users (Cresci et al., 2017).

Table 5.1 presents the AUC comparison between the RF, SVM, and NB algorithms used in this work with those used by (Rodríguez-Ruiz et al., 2020). The values refer to the training of the algorithms with 90% of the dataset Genuine accounts combined with 90% of Social spambots #1 Xor Traditional spambots #1. The classifiers were tested with the remaining 10% of the set used for training.

Comparing the reduced set of features using the RF, SVM, and NB algorithms in contrast to the work of Rodríguez-Ruiz et al. (2020) (that uses 14 features) is noticeable in Table 5.1: the results reach better AUC values for the Traditional dataset than for the Social1 dataset. On average, the results are close to the standard deviation smaller than 0.066. The low standard deviation values suggest that the results between the two datasets tested are similar even with a reduced feature set. Note in Figure 5.2 the results presented in Table 5.1, where for each classifier, the two first bars represent the AUC values achieved using the two datasets (i.e., Social1 and Traditional) and the third bar the mean value. The standard deviation is also presented in the first two bars. Visually the results are very similar using a reduced feature set that positively influences the computational complexity.

Table (5.1) The AUC comparison between this work and Rodríguez-Ruiz et al. (2020).

ML Technique	**Social1**	**Traditional**	**Mean**	**Standard deviation**
RF	0.9758	**0.9999**	0.9878	0.0170
SVM	0.9382	**0.9978**	0.9680	0.0421
NB	0.9011	**0.9937**	0.9474	0.0654
RFRodríguez-Ruiz et al. (2020)	**0.9800**	0.9990	0.9895	0.0134
SVMRodríguez-Ruiz et al. (2020)	**0.9600**	0.9220	0.9410	0.0269
NBRodríguez-Ruiz et al. (2020)	**0.9580**	0.9850	0.9715	0.0191

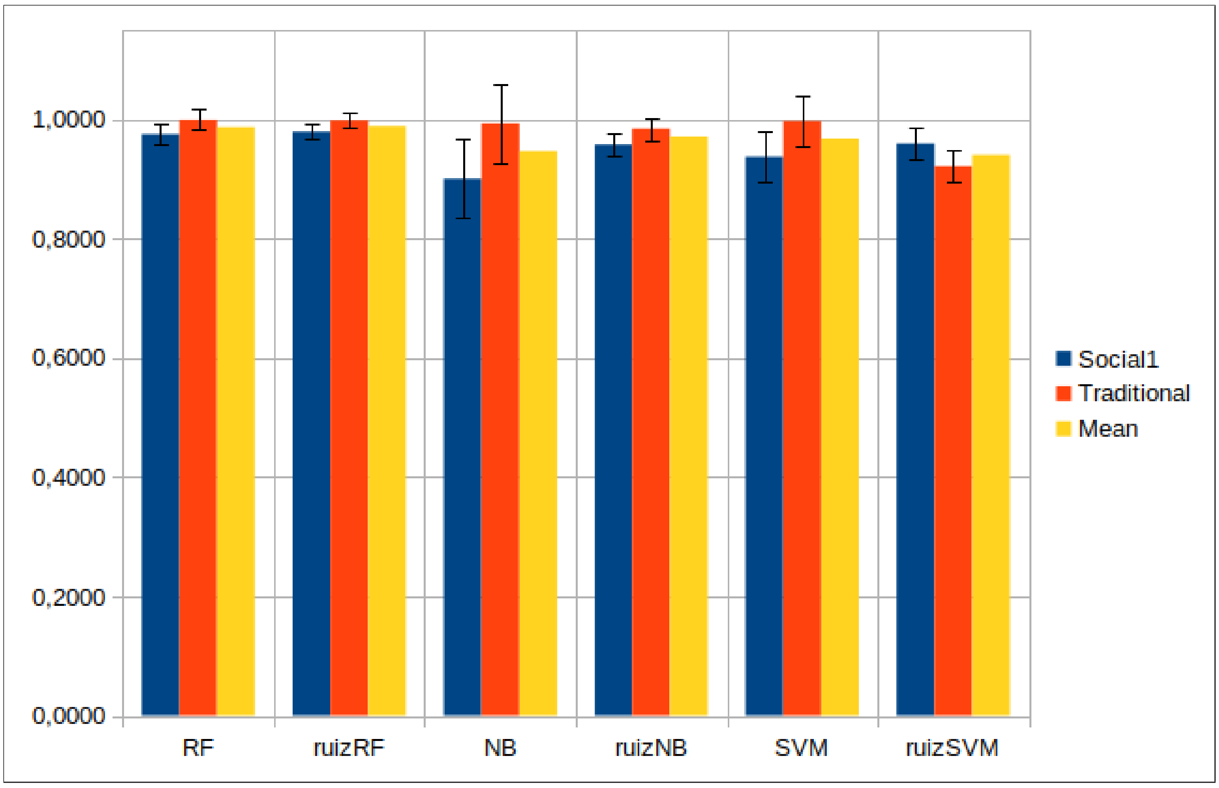


Figure (5.2) Parallel of the AUCs achieved by this work and Rodríguez-Ruiz et al. (2020).

Table 5.2 presents a collection of metrics for the classifiers considering the dataset variation to view their behavior. The metrics include commonly used metrics, such as accuracy, AUC, recall, and F1-score. The algorithms performed well, with a mean score of 0.8549. The standard deviation of 0.1889 suggests that the accuracy of the algorithms is considered homogeneous. It is possible to highlight the RF algorithm in the Traditional dataset with an accuracy of 0.9933 (with recall equal to 0.9850), indicating good accuracy and sensitivity. We believe the notable results are derived from the nature of the dataset that contains less sophisticated bots.

Table (5.2) Metrics for proposed classifiers - training/test dataset.

ML Technique/Dataset	**Accuracy**	**AUC**	**Recall**	**F1-score**
RF/Social1	0.9821	0.9758	0.9636	0.9737
RF/Traditional	**0.9933**	0.9999	**0.9850**	0.9902
SVM/Social1	0.8281	0.9382	0.6186	0.6421
SVM/Traditional	0.9821	0.9978	0.9778	0.9744
NB/Social1	0.5000	0.9011	0.6710	0.4980
NB/Traditional	0.8438	0.9937	0.8994	0,8145

As expected, the probabilistic algorithm NB obtained less striking results. However, some techniques can improve the algorithm's overall performance, although such techniques were not applied because they are outside the scope of the work. The main objective of this work is to compare the results with Rodríguez-Ruiz et al. (2020), considered a work in the state–of–the–art. It is also worth mentioning that authors in Rodríguez-Ruiz et al. (2020) did not make clear the execution parameters used in the algorithms tested. However, adopting basic implementations of the algorithms, we achieved similar results.

The central argument of Rodríguez-Ruiz et al. (2020) is that despite being more accurate, the multiclass classifiers are not good at classifying bots different than the types they were used in their training. Thus, a different approach is needed for the classification of new bots (with greater robustness), treating bots as profiles that exhibit anomalous behavior. For that, a one-class classifier is used. The Rodríguez-Ruiz et al. (2020) usage of one-class classifiers motivated the need to test the reduced feature set with this kind of classifier. Was adopted the ocSVM for three reasons: (i) it is a one-class version of one of the most used algorithms for bot classification; (ii) it has been tested by Rodríguez-Ruiz et al. (2020) allowing the performance comparison; and (iii) there is a version of the classifier in the same library adopted on the experiments (Pedregosa et al., 2011).

A comparison using the AUC for ocSVM from Rodríguez-Ruiz et al. (2020) and this proposal is presented in Table 5.3. The one-class classifier has lower performance when compared to multiclass classifiers trained using a specific type of bot. However, they maintain the performance for new types of bots (Rodríguez-Ruiz et al., 2020). Thus, the ocSVM classifier obtained a lower AUC than the multiclass binary version. This behavior was expected since the classifier achieved an AUC average of 0.7361, indicating that it is applicable for bot classification.

The reduced set of features proposed in this work achieved a lower AUC in contrast to (Rodríguez-Ruiz et al., 2020). However, the low standard deviation of 0.0177 shows that the AUC varies little for the distinct datasets. According to a work in the state– of– the– art, this slight variation is one of the main reasons for using a one-class classifier.

Table (5.3) AUC comparison of ocSVM classifiers.

ML Technique	**Social1**	**Traditional**	**Mean**	**Standard deviation**
ocSVM	0.7236	0.7486	0.7361	0.0177
ocSVM (Rodríguez-Ruiz et al., 2020)	0.8710	0.8980	0.8845	0.0191

Through the use of a few but expressive features, a classifier was obtained with efficiency comparable to the state–of–the–art work (Rodríguez-Ruiz et al., 2020). It is possible to mention the good results achieved by the multiclass classifiers during the experiments. The algorithms performed well with a mean score of 0.8549 and a standard deviation of 0.1889, suggesting that the accuracy of the algorithms can be considered homogeneous. Besides, all multiclass classifiers obtained AUCs greater than 0.9. The scores achieved in the algorithm evaluation metrics suggest that the reduced feature set is sufficient for classifying bots on Twitter with multiclass algorithms.

In addition to the good results with multiclass algorithms, the one-class approach suggests a priori that the reduced feature set can detect bots using this strategy. In comparison to the one-class classifiers used by Rodríguez-Ruiz et al. (2020), we used only one. In the experiments, we did not apply optimization techniques. Thus, the results achieved with the one-class classifier could be even better.

5.1.2 Feature set validation

After the previous experiment, the work Yang et al. (2020) was found, providing another evidence that the reduced feature set proposed is a good option for efficient and effortless Twitter bot detection. Since that, the same features present on the reduced feature set were included (and achieved the higher importance's scores) on the feature analysis presented on the cited work.

Nevertheless, an additional experiment was executed to validate the feature set established in the first experiment. The chosen approach was to do this task through an AutoML tool. This kind of tool can automatize the ML pipeline, including the feature selection task. Thus, it allows the comparison between the reduced feature set (Experiment 1 and Fonseca Abreu et al. (2020)) and the AutoML generated feature set. This comparison can be used to validate the proposed feature set. Since the feature selection provided by some AutoML tools can deliver more substantial evidence to establish better the reduced feature set used.

Thus, a series of tests involving the AutoML TPOT were executed, including tests involving the classifier for each combination of datasets. First, the classification was executed using the TPOT and all non-textual features available on the datasets. The resultant feature selection was computed the feature importance of each feature chosen by the TPOT and the performance measures of the classifier using them. After this, the TPOT was executed: without the feature selection; with the reduced feature set presented in authors (2020); and with all non-textual features. These executions allow the comparison of the performances of the classifiers with different feature sets. The feature set validation experiment description includes the following details.

- **Experiment objective**: explore the AutoML tools to execute feature selection on the Twitter bot detection problem.
- **Dataset used**: the following CSV files from the dataset provided by Cresci et al. (2017) were chosen: Social spambots #1 - includes accounts that retweeted one candidate for mayor of Rome in the 2014 Italian elections, where these candidates hired a marketing company that used 991 bots to carry out the campaign. Genuine accounts with 3474 accounts were classified as legitimate using the following method: the research group asked a simple question in natural language for randomly chosen profiles in the OSNs. After this step, humans analyzed the responses to define which accounts are legitimate. Traditional spambots #1 is the same dataset provided by Yang et al. (2013), which contains information on 1000 bots that made posts containing malicious links.
- **Dataset partitioning**: 90% of the data are used for training the classifier, and the remaining 10% are used for testing the classifier (as used in authors (2020)).
- **Validation strategy**: cross-validation used by the chosen technique (TPOT).
- **Feature selection**: the feature selection from TPOT between all non-textual features available on the datasets.
- **ML algorithms tested**: the algorithm selection was performed by the AutoML tools. The TPOT selects its ML algorithm from Scikit-learn Pedregosa et al. (2011) finding the best algorithm and the best hyper-parameters.
- **Results evaluation metric**: the feature importance rank provided by the TPOT.

Result analysis

The feature set validation experiment was implemented using TPOT to perform a feature selection on Social spambots #1 and Traditional spambots #1 datasets. In both cases, from the 18 initial features, the AutoML selected nine features and provided the importance of each feature after the classification. Table 5.4 shows the selected features and their importance in the bot classification for the Social spambots #1 dataset. Note that the three most important features are favourites_count, utc_offset, and friends_count.

Table (5.4) Feature importance for the TPOT on the Social spambots #1 dataset.

Selected feature	**Feature importance**
favourites_count	0,7645
utc_offset	0,0525
friends_count	0,0409
geo_enabled	0,0372
default_profile	0,0277
followers_count	0,0265
statuses_count	0,0252
listed_count	0,0149
profile_use_background_image	0,0107

Table 5.5 presents the selected features with Traditional spambots #1 dataset. The most important features were favourites_count, friends_count, and utc_offset. At the Social1 spambots #1 dataset, the importance of utc_offset is higher than in Traditional spambots #1. We believe that it occurs because of the context that the dataset is inserted. Social spambots #1, is composed of bots that were used to retweeted a candidate for mayor of Rome in the 2014 Italian elections. Therefore, it is reasonable to assume that they share the same time zone. Note that TPOT tool selected the same features for both datasets (Tables 5.4 and 5.5). Moreover, the five features used in authors (2020) were included in the list ({favorites, friends, followers, statuses, listed}_count).

Table (5.5) Feature importance for the TPOT on the Traditional spambots #1 dataset.

Selected feature	**Feature importance**
favourites_count	0,9511
friends_count	0,0258
utc_offset	0,0176
statuses_count	0,0029
geo_enabled	0,0012
followers_count	0,0006
listed_count	0,0003
profile_use_background_image	0,0001
default_profile	0,0001

The results achieved with the TPOT feature selection indicate that the reduced feature set developed previously was adequate for the bot detection task. It is interesting to note how close results obtained by feature selection based on previous knowledge and experience are with those obtained by AutoML. In a certain way, both approaches tend to validate each other mutually.

5.2 AgentBotSpotter experiments

Two different tests were carried to validate the proposal. The first one intended to measure the performance of the MAS developed, using the same methodology as Fonseca Abreu et al. (2020) and comparing the two works. At this first test, the ABS performance was measured using real-world offline data from the datasets provided by (Cresci et al., 2017). The second experiment is a proof of concept involving using the MAS to detect Twitter bots on an online stream of tweets. Both tests are better discussed in the following subsections. Moreover, they can be summarized as follows.

- **Experiment objective:** test the implementation of the ABS with real-world data extracted from Twitter.
- **Dataset used:** the following CSV files from the dataset provided by Cresci et al. (2017) were chosen: Social spambots #1 - includes accounts that retweeted one candidate for mayor of Rome in the 2014 Italian elections, where these candidates hired a marketing company that used 991 bots to carry out the campaign. Genuine accounts with 3474 accounts were classified as legitimate using the following method: the research group asked a simple question in natural language for randomly chosen profiles in the OSNs. After this step, humans analyzed the responses to define which accounts are legitimate. Traditional spambots #1 is the same dataset provided by Yang et al. (2013), which contains information on 1000 bots that made posts containing malicious links.
- **Dataset partitioning:** 90% of the data are used for training the classifier. The remaining 10% are used for testing the classifier (according to Rodríguez-Ruiz et al. (2020)) on the offline tests.
- **Feature selection:** A selection of five features made empirically by prior experience with creating bots for Twitter Abreu et al. (2019, 2020), including the number of tweets ($statuses\_count$), number of followers ($followers\_count$), number of friends ($friends\_count$), number of likes given by the account ($favorites\_count$), and number of lists that the account is included ($listed\_count$).
- **Validation strategy:** The holdout method was adopted here since each classifier passed through cross-validation at the feature set definition experiment.

- **ML algorithm tested:** ensemble of RF, SVM,and NB implemented by (Pedregosa et al., 2011).
- **Results evaluation metric:** The entire database was measured through accuracy, recall, AUC, and F1-score. Average and standard deviation of both algorithms and datasets.

5.2.1 Offline assessment

To validate the proposed solution, the ABS performance was compared with the three algorithms executed separately as presented in Fonseca Abreu et al. (2020). In Table 5.6 we present the comparison of the scores for the three algorithms - RF, SVM, and NB - using the Social1 and Traditional datasets. As in Fonseca Abreu et al. (2020) the implementation of the ML algorithms are provided by (Pedregosa et al., 2011). ABS presented the second-best performance among other algorithms with an accuracy of 0.9795 in dataset Social1 and 0.9840 in the Traditional dataset. It was only behind the RF results with 0.9821 and 0.9933 with Social1 and Traditional datasets.

Table (5.6) Accuracy, AUC, recall and e F1-score for classifiers (alone and combined in ABS), according to train/test dataset.

ML algorithm/ Dataset	**Accuracy**	**AUC**	**Recall**	**F1-score**
RF/Social1	**0,9821**	0,9758	0,9636	0,9737
RF/Traditional	**0,9933**	0,9999	0,9850	0,9902
SVM/Social1	0,8281	0,9382	0,6186	0,6421
SVM/Traditional	0,9821	0,9978	0,9778	0,9744
NB/Social1	0,5000	0,9011	0,6710	0,4980
NB/Traditional	0,8438	0,9937	0,8994	0,8145
ABS/Social1	**0,9795**	0,9716	0,9587	0,9682
ABS/Traditional	**0,9840**	0,9996	0,9776	0,9756

According to Powers (2011); Sokolova et al. (2006), the recall (also known as sensitivity) is the ratio between the correctly classified examples (true-positive - TP) and the incorrectly classified examples (false-negative - FN). In other words, it is the proportion of true-positive cases that are correctly predicted. As we can see, the recall score from the ABS is considerably high (it is only lower than the RF) in both datasets. This measure indicates that the ABS maintains the high sensitivity achieved by the best ML classifier.

According to Tharwat (2018), F1-score represents the harmonic average between precision and recall. The value of the F1-score is a number between zero and one, where values closer to one indicate high classification performance. Over again, the MAS keeps the good score achieved by the experiment performed at Fonseca Abreu et al. (2020), which is a good indicator that the solution proposed applies to the task of classifying Twitter bots. The results related to the AUC score are better explained below.

In Table 5.7 there is a comparison between the AUC of the classifiers executed separately and combined in the ABS. For the AUC calculus, the ABS' probability was measured considering the average of the probabilities of the algorithms. This procedure is the same used in Pedregosa et al. (2011) for the calculus of the probability for the Adaboost ensemble method. Since we have fewer examples for each type of bot than examples of legitimate accounts in the dataset, choosing AUC as a performance measure is appropriate (Provost, F and Fawcett, T, 1997; Rodríguez-Ruiz et al., 2020). The ABS achieved the second-best performance among the other algorithms with an average AUC of 0.9856 (standard deviation of 0.0199) only behind the RF results with 0.9878 (standard deviation of 0.0170). Figure 5.3 contains the results presented in Table 5.7, where for each classifier, the first two bars represent the AUC values obtained using the two datasets (i.e., Social1 and Traditional), and the third bar contains the average value. The standard deviation is also shown in the first two bars.

Table (5.7) Comparison between AUC obtained for ABS and the classifiers alone as in (Fonseca Abreu et al., 2020)

Técnica de ML	**Social1**	**Traditional**	**Average**	**Standard Deviation**
RF	0,9758	0,9999	**0,9878**	**0,0170**
SVM	0,9382	0,9978	0,9680	0,0421
NB	0,9011	0,9937	0,9474	0,0654
ABS	0,9716	0,9996	**0,9856**	**0,0199**

According to the presented experiments, the ABS results are close to the best algorithm (RF), meaning the approach can be used for classification with satisfactory performance. We might consider that the method by which the vote is taken is not complementary. Also, the RF algorithm presents better results considering the datasets used, but we cannot guarantee it will perform better with any dataset. The ensemble methods in ML produce optimal predictive models, as pointed out in the literature and the MAS approach, which can provide an ideal framework for implementing such methods. In this work, we provided the ABS proof of concept with three different ML algorithms and a simple decision process based on voting. However, other solutions can enhance the results of ensemble methods especially considering online tweet bot detection.

5.2.2 Online assessment

An essential attribute for a bot classifier is execution online, thus becoming a detector. In Abreu et al. (2019, 2020) there is a demonstration that the detection of bots from Twitter allows very simple bots to continue working on the platform. Therefore, it is necessary to have tools that help discover bots as close as possible to real-time. A very slow tool can detect bots when they have already been excluded or have already caused damage. Thus, harmful bots can be discovered earlier by minimizing the damage caused by them. During the literature review, it was noticeable that most articles do not detect bots online. In this way, a proof of concept (PoC) of the execution of the ABS was also carried out using online data collected from the Twitter API. The main objective of this PoC was to assess the ABS approach feasibility for online detection.

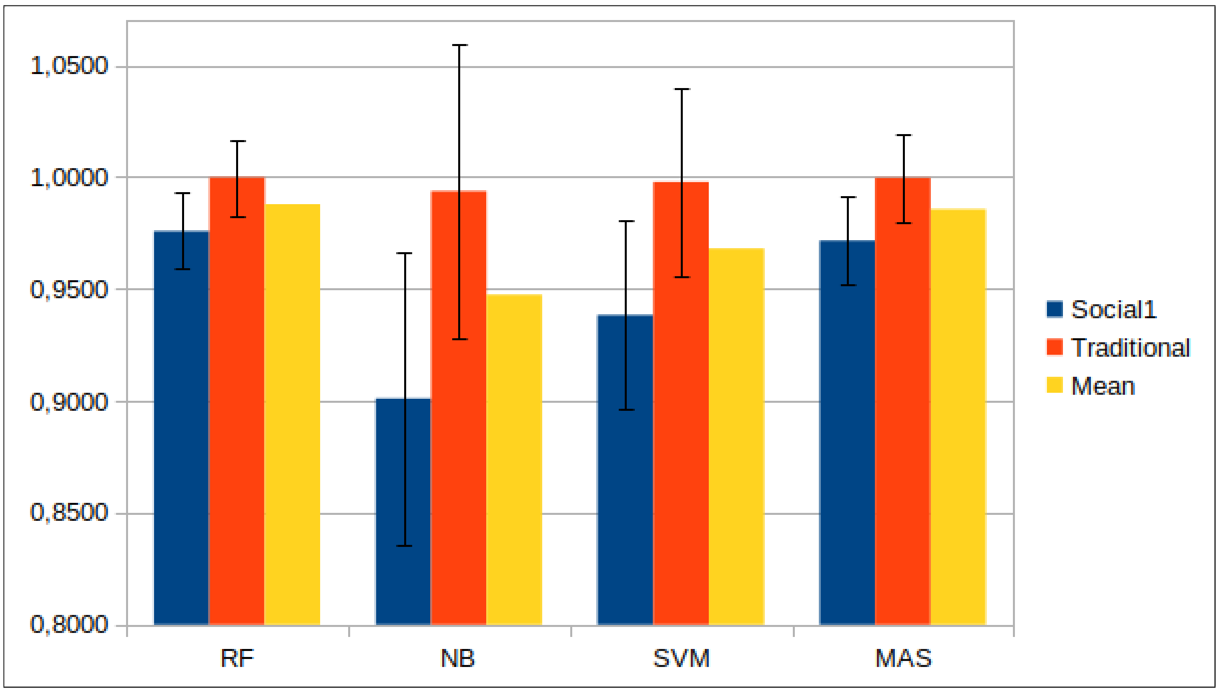


Figure (5.3) Parallel of the AUCs obtained for ABS and Fonseca Abreu et al. (2020).

Online PoC

The methodology was simple and used the ABS set up with the three ML classifiers (RF, SVM, and NB) trained with the dataset Social1 combined with the dataset genuine accounts (both provided by Cresci et al. (2017)), the same used in part of the first test. This dataset choice was made intending to discover the most sophisticated bots, as argued in (Cresci et al., 2017; Fonseca Abreu et al., 2020). The stream of tweets was retrieved looking for the keyword "*vacina*" which is the Portuguese word that means vaccine. This keyword was chosen because the test was carried out during the COVID19 pandemic, and this theme was a hot topic in Brazil. It is notorious that it is more likely to capture bots in a larger sample of accounts, and the keyword is beneficial to allow for it. The test was executed using one notebook with the following specs: $7,8$ GiB of RAM, CPU Intel®

Core^{TM} i7-3610QM (clock 3.3 Ghz, 4 cores and 8 threads), GPU NVIDIA® GeForce® GT 630M with 2GB DDR3 VRAM, Solid State Drive 250 GB, Operational System Xubuntu 20.04.2 LTS (64 bits).

The execution took place on February $12^{th}, 2021$, for approximately six hours (from 11:56 am to 6:00 pm). About 44,156 tweets were captured, corresponding to 31,271 unique users, which were classified (representing a throughput of approximately 123 tweets per minute, roughly two per second). Among these, 183 tweets came from profiles classified as bots by ABS. 144 different Twitter accounts published these 183 suspicious tweets.

In March $25^{th}, 2021$, a request for data about these 144 profiles classified as bots was carried out. The Twitter API returned data about only 17 profiles, which means that 127 suspicious profiles were suspended by Twitter or excluded by their owners. It suggests that $88,19\%$ of the users classified as bots by the MAS method were correctly labeled, using the same methodology as Lee et al. (2011) to validate their results. This preliminary test, although quite simple, showed that the proposed methodology was on the right track and applies to one data.

Extended online experiment

From results obtained from the PoC, an extended online experiment was carried out. This experiment also applied the same validation methodology as the PoC (and Lee et al. (2011)) to validate their results. Moreover, it applied the ABS set up with the three ML classifiers (RF, SVM, and NB) trained with the dataset Traditional combined with the dataset genuine accounts (both provided by Cresci et al. (2017)). This combination was also tested on the offline assessment. The dataset choice was based on two key ideas:

- provide evidence of the robustness of the method when the training dataset varies;and
- avoid any bias inserted by the adoption of a single combination of datasets during the online experiments.

Thus, it would be feasible to demonstrate any possible influence of the dataset on the online validation adopted and demonstrate some independence of choice of the dataset on the proposed solution. The chosen keyword, "*vacina*", remained the same.

The experiment was carried out from August $18^{th}, 2021$ to August $23^{th}, 2021$, during the COVID19 pandemic, which justifies the maintenance of the keyword. The test was executed using the same hardware setup as the one used in the PoC. During the approximately 114 hours of experimentation, were analyzed $797,241$ tweets from $300,469$ different users. 1597 were published from the 678 probable bots. Figure 5.4 is possible to visualize the difference between the experiment metrics through a graphic. Because of the discrepancy between the numbers, the logarithmic scale was employed.

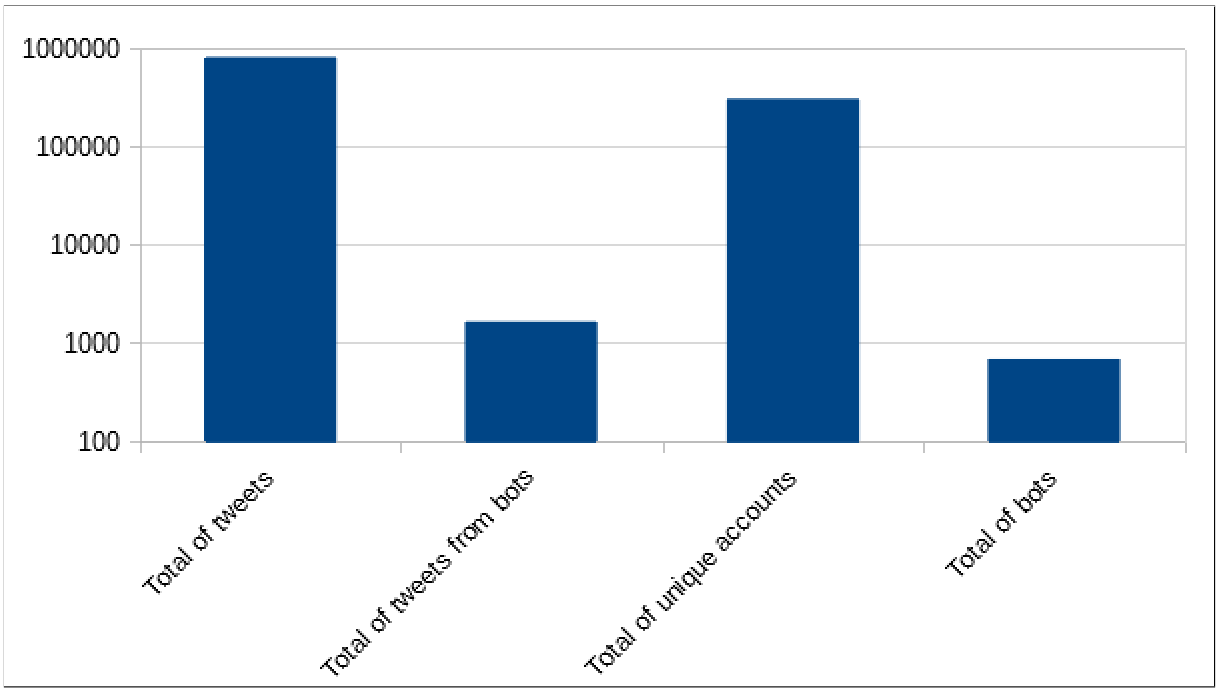


Figure (5.4) Visualization of the classification results from the extended experiment.

This experiment used the same hardware setup as the PoC and included some improvements in the code instrumentation compared to the PoC, and some additional measuring was done on resource consumption and time of detection. The ABS performed well with a throughput of 116.6 tweets per second. It was slightly worse than the PoC, but at an extended experiment, it can be explainable by the presence of periods with lower user activity on the analysis.

The average time of detection was 14.428 seconds, and 99.02% of the detections occurred in less than 30 seconds. It interesting to highlight that, as presented in a previous work Abreu et al. (2019, 2020) the default bot detection response from the Twitter platform took from hours to days. Thus, a tool that performs the detection in seconds is a considerable improvement. In terms of resource usage, the peak of RAM consumption was observed near 780 Mb, and the highest CPU usage observed was 84%. Given the hardware adopted, the consumption of computational resources can be rated as low, despite the limited hardware setup used.

Figure 5.5 presents the behavioral pattern of humans and bots distributed by the weekday, where the blue line refers to the human profiles, and the red line represents the bots. The horizontal axle is referent to the weekdays (zero represents Monday, one represents Tuesday, and so on), and the vertical axle shows the number of profiles observed. Both patterns are considerably similar.

Figure 5.6 presents the average daily behavioral pattern of humans and bots distributed by the time of the day, where the blue line refers to the human profiles, and the red line represents the bots. The horizontal axle corresponds to the time of the day (zero is midnight, one represents 1 am, and so on), and the vertical axle shows the number of profiles observed. A horizontal dashed line represents the mean of tweets and two vertical dashed lines that mark the period when the PoC was executed. Again, both patterns are considerably similar. However, at this comparative, the bots curve is slightly unstable, presenting more complex variations.

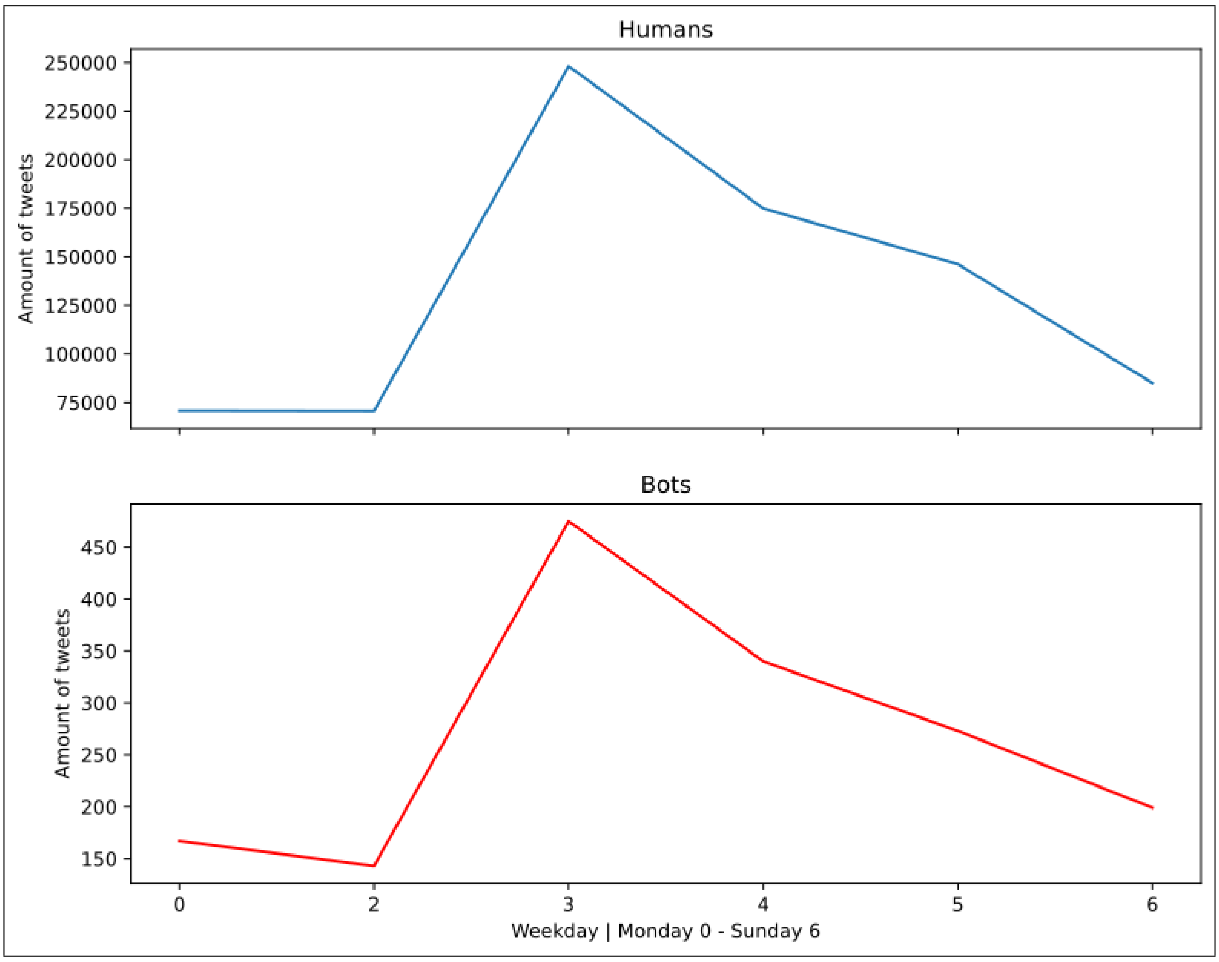


Figure (5.5) Analysis of the behavioral pattern of humans and bots distributed by the weekday.

When compared to the PoC, the extended experiment presented a significant drop in the detection rate. The number of bots detected on the PoC was near 0.46% over the total of tweets analyzed, while on the extended experiment this rate was 0.23%. To explain this, Figures 5.5 and 5.6 are instrumental. Three hypotheses might try to explain the drop. The first one is that simply the number of bots using this keyword diminished. An analysis of the full historical records with the keyword "*vacina*" is required to confirm this. The second one is that the dataset used influenced the detection rate, but the offline assesment does not corroborates this, since there is not a significant difference in the performance when comparing the two datasets . An additional extended experiment should be executed with the Social1 dataset to confirm this. These hypotheses were not tested because of a lack of resources.

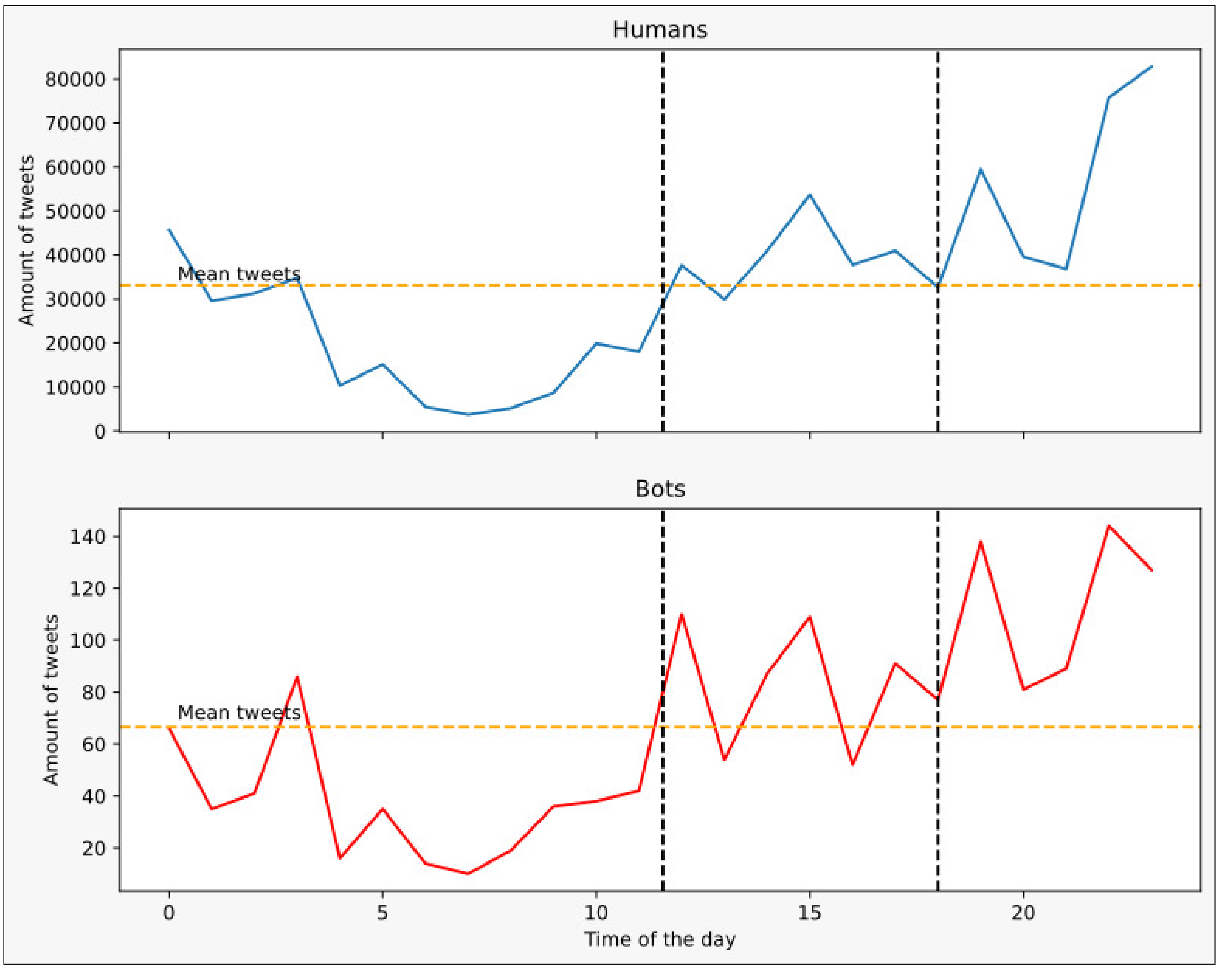


Figure (5.6) Analysis of the behavioral pattern of humans and bots distributed by time of the day.

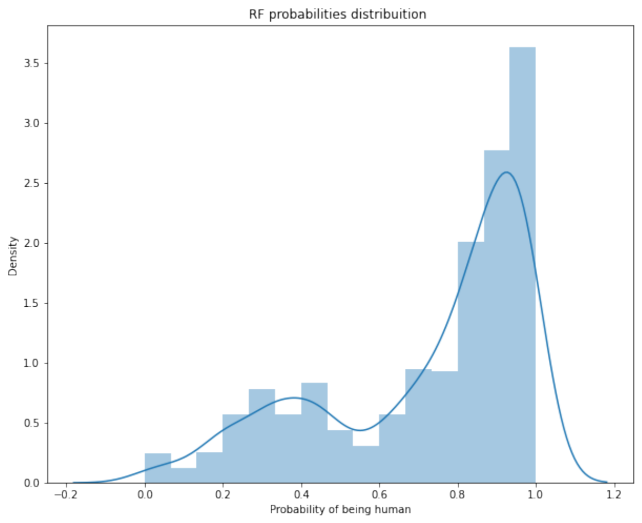
The third hypothesis, which is believed to be more likely, is that the extended experiment diluted the detection rate because it covers more activity periods. This hypothesis can be stated because, according to Figure 5.5, the PoC occurred in one weekday with activity records among the highest: the Friday. Moreover, Figure 5.6 reveals that the time interval of the PoC coincidently contains considerably high records of tweets. None of those attributes of the PoC were intentional, and the period of the PoC execution was chosen randomly. Nevertheless, any stronger statement can not be made because it requires a deeper analysis of the Twitter records.

Moreover, Figure 5.7 shows the histograms of the probabilistic distribution of votes from the classifiers. As said before, the ABS' probability was measured considering the average of the probabilities of the algorithms. Figures 5.7a, 5.7b and 5.7c demonstrate that each classifier is specialized in a different pattern of profiles. Figure 5.7d shows that the combination of the classifiers allows the ABS to cover a wider spectrum of possible bot profiles.

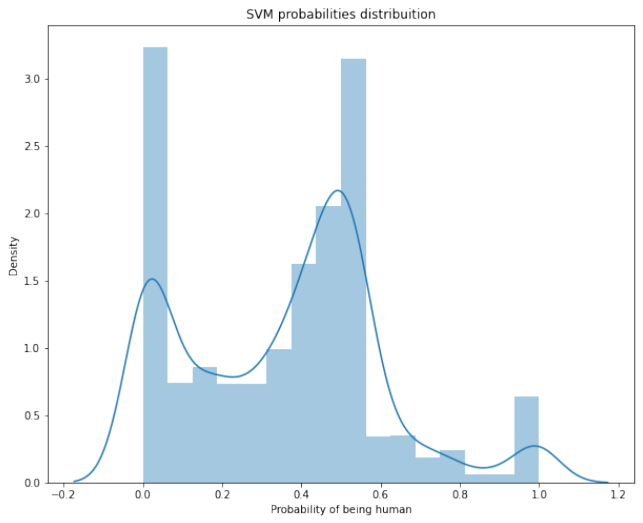
5.3 Summary of the empirical validation

This chapter reported the empirical validation for the ABS. The experiments described are divided into two different areas: feature selection and MAS experiments. The feature selection experiments successfully defined a reduced feature set on an excellent twitter bot detection. On the other side, the MAS experiments showed that the ABS could detect Twitter bots online and offline. The next chapter contains the conclusions gathered from the experiments.

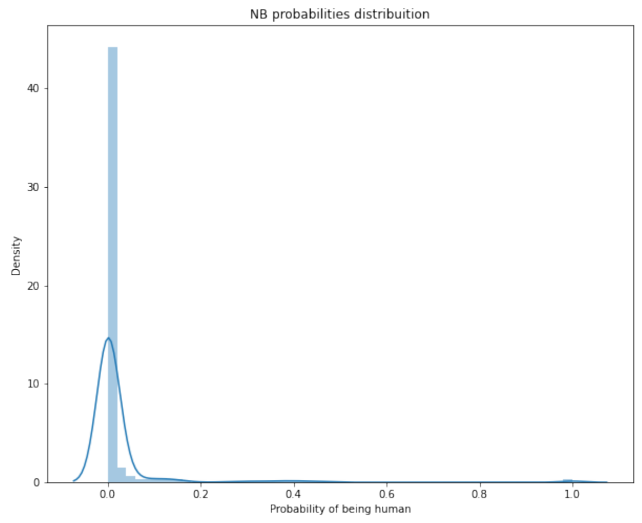
Figure (5.7) Distribution of the probabilistic votes of the classifiers



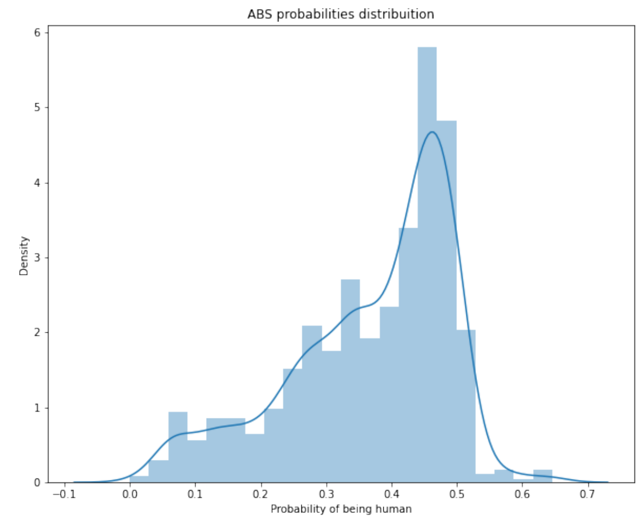
(a) Distribution of the probabilistic votes of RF.



(b) Distribution of the probabilistic votes of SVM.



(c) Distribution of the probabilistic votes of NB.



(d) Distribution of the probabilistic votes of ABS.

Chapter 6

Conclusion

This work's main goal is reached through the development of an online MAS to identify bots on Twitter. The secondary objectives were reached too. The literature review was able to identify significant works with AI techniques, but none employed the MAS approach. Among the ML works, the principal techniques were identified and adopted for the developed ABS.

In addition to the theoretical background, details related to the solution design, implementation, and experiments were presented. Among the initial experiments (Section 5.1), it is vital to emphasize the reduced but expressive feature set, which achieves performance comparable to a state–of–the–art work, i.e., average AUC of 0.9878 against 0.9895 obtained by Rodríguez-Ruiz et al. (2020). Furthermore, the reduced feature set was validated through an AutoML approach and other works found in the literature. According to Buczak and Guven (2016), the reduced feature set influences the computational complexity of the RF and NB algorithms.

Considering the conducted experiments, the good results achieved by the multiclass classifiers are noticeable. The algorithms performed well with a mean score of 0.8549 and a standard deviation of 0.1889, suggesting that the accuracy of the algorithms is considered homogeneous. Besides, all multiclass classifiers obtained AUCs greater than 0.9. The scores achieved in the algorithm evaluation metrics suggest that the reduced feature set is sufficient for the Twitter bot classification with multiclass algorithms.

Nevertheless, this work has boundaries like the sole use of supervised ML algorithms, the exclusive adoption of the reduced feature set, the chosen MAS development framework, and Twitter's API response time. Despite this, the initial experiments established ML algorithms and a reduced set of features proving the suitability for the Twitter bot classification. Moreover, at the MAS experiment, the agents mentioned in Section (1.3) and Chapter (4) were built and integrated on the ABS. Furthermore, this work was able to answer the questions established in Chapter 1 as follows.

- RQ1: Is it feasible to build a MAS to detect Twitter bots using an offline dataset to train the ML algorithms?
  - Yes, the ABS proved his ability to detect Twitter bots offline. The ABS obtained an average AUC of 0.9856 and a standard deviation equal to 0.0199. These metrics show that MAS is not only feasible but practical and useful for labeling bots on Twitter.
- RQ2: Can the MAS detect bots over an online flow of tweets in a period that the information is still useful to react against bots?
  - Yes, the developed ABS was tested over an online setup with a flow of tweets. On a preliminary PoC the ABS delivered an accuracy of detection of 88.19% and an indication of good throughput. Furthermore, the comprehensive online experiment preserves the throughput characteristic achieving an average detection time of 14.428 seconds. Thus, it is fair to say that the ABS detection provides information about bot activity in a timely, precise, and relevant manner.

The ABS architecture naturally accommodated online detection as it uses a direct pipeline for processing and classifying tweets captured by agents. Additionally, the approach applied in the ABS development allows exploring ML classifiers by replacing, removing, or adding algorithms. Despite promising results, the investigation needs to be deepened through future works. The histograms of the probabilistic distribution of votes from the classifiers (Figures 5.7a, 5.7b, 5.7c, and 5.7d) demonstrate that each classifier is specialized in a different pattern of profiles. Thus, the ensemble of the classifiers allows the ABS to cover a wider spectrum of possible bot profiles.

As future work, there is the possibility to provide the ABS online for Twitter users. The way to achieve this is by creating and publishing a REST API to access the application. Another way is the creation of a website similar to the works of Varol et al. (2017) and Sayyadiharikandeh et al. (2020), which might improve the ABS usability. The website can contain a complete dashboard with statistics about the detections performed.

References

Abreu, J. V. F., Fernandes, J. H. C., and Gondim, J. J. C. (2019). Desenvolvimento interativo de um robô para execução de ataques de engenharia social no twitter. In *2019 IADIS Conferencia Ibero Americana WWW/Internet 2019 (CIAWI)*, pages 243–247. IADIS. 2, 3, 49, 50, 57, 59, 62

Abreu, J. V. F., Fernandes, J. H. C., Gondim, J. J. C., and Ralha, C. G. (2020). Bot development for social engineering attacks on twitter. *CoRR*, abs/2007.11778. 2, 3, 49, 50, 57, 59, 62

Ahmed, F. and Abulaish, M. (2013). A generic statistical approach for spam detection in online social networks. *Computer Communications*, 36(10):1120 – 1129. 32, 33, 79

Al-Qurishi, M., Alrubaian, M., Rahman, S. M. M., Alamri, A., and Hassan, M. M. (2018). A prediction system of sybil attack in social network using deep-regression model. *Future Generation Computer Systems*, 87:743 – 753. 1, 30, 34, 78

Al-Qurishi, M., Hossain, M. S., Alrubaian, M., Rahman, S. M. M., and Alamri, A. (2018). Leveraging analysis of user behavior to identify malicious activities in large-scale social networks. *IEEE Transactions on Industrial Informatics*, 14(2):799–813. 32, 78

Alharthi, R., Alhothali, A., and Moria, K. (2019). Detecting and characterizing arab spammers campaigns in twitter. *Procedia Computer Science*, 163:248 – 256. 16th Learning and Technology Conference 2019Artificial Intelligence and Machine Learning: Embedding the Intelligence. 32, 36, 78

Alothali, E., Zaki, N., Mohamed, E. A., and Alashwal, H. (2018). Detecting social bots on twitter: a literature review. In *International Conference on Innovations in Information Technology (IIT)*, pages 175–180. IEEE. 1, 5, 28, 29, 34

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press. 15, 16, 19

authors, A. (2020). Anonymized title. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages x–y. 55, 56

Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing multi-agent systems with JADE*, volume 7. John Wiley & Sons. 14

Beğenilmiş, E. and Uskudarli, S. (2018). Organized behavior classification of tweet sets using supervised learning methods. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, WIMS '18, New York, NY, USA. Association for Computing Machinery. 2, 33, 36, 78

Bilalli, B., Abelló, A., Aluja-Banet, T., and Wrembel, R. (2018). Intelligent assistance for data pre-processing. *Computer Standards & Interfaces*, 57:101–109. 24

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg. 19, 21

Borges, R. (2020). No twitter, bolsonaro volta a defender hidroxicloroquina e rebate críticas – money times. `https://www.moneytimes.com.br/no-twitter-bolsonaro-volta-a-defender-hidroxicloroquina-e-rebate-criticas/`. Accessed 09/09/2020. 1

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32. 18

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236. 12, 38

Buczak, A. L. and Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176. 67

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16 – 28. 40th-year commemorative issue. 22

Chauhan, K., Jani, S., Thakkar, D., Dave, R., Bhatia, J., Tanwar, S., and Obaidat, M. S. (2020). Automated machine learning: The new wave of machine learning. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 205–212. IEEE. 23

Chen, Z. and Subramanian, D. (2018). An unsupervised approach to detect spam campaigns that use botnets on twitter. *CoRR*, abs/1804.05232. 1, 30, 78

Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. (2012). Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing*, 9(6):811–824. 2, 8, 31, 49, 79

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. (2015). Fame for sale: Efficient detection of fake twitter followers. *Decision Support Systems*, 80:56–71. 30, 33

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. (2016). Dna-inspired online behavioral modeling and its application to spambot detection. *IEEE Intelligent Systems*, 31(5):58–64. 1, 33

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. (2017). 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*, WWW '17 Companion, page 963–972, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee. 32, 37, 45, 49, 51, 55, 57, 60, 61

Cruz, F. B. (2020). Adnet: Secom e mário frias partem para cima de adnet após imitação | veja. `https://veja.abril.com.br/cultura/secom-e-mario-frias-partem-para-cima-de-adnet-apos-imitacao/`. Accessed 09/09/2020). 1

da Mata, L. (2019). Plano de trabalho CPMI da Fake News. Congresso Nacional. Disponível em: `http://legis.senado.leg.br/sdleg-getter/documento/download/d78bab7d-515f-45df-b785-03e364a7e138` Acessado em: 19/04/2020. 2

Dalpiaz, F., Franch, X., and Horkoff, J. (2016). istar 2.0 language guide. 13, 38

Davis, C. A., Varol, O., Ferrara, E., Flammini, A., and Menczer, F. (2016). Botornot: A system to evaluate social bots. In *Proceedings of the 25th international conference companion on world wide web*, pages 273–274. 33

Dickerson, J. P., Kagan, V., and Subrahmanian, V. S. (2014). Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 620–627. 8, 31, 79

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *MULTIPLE CLASSIFIER SYSTEMS, LBCS-1857*, pages 1–15. Springer. 3, 20, 21

Echeverria, J., Besel, C., and Zhou, S. (2017). Discovery of the twitter bursty botnet. *Data Science for Cyber-Security*. 8

Fazil, M. and Abulaish, M. (2018). A hybrid approach for detecting automated spammers in twitter. *IEEE Transactions on Information Forensics and Security*, 13(11):2707–2719. 7, 8, 31, 37, 79

Fernquist, J., Kaati, L., and Schroeder, R. (2018). Political bots and the swedish general election. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 124–129. IEEE. 2, 8, 9, 33, 36, 79

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., and Hutter, F. (2020). Auto-sklearn 2.0: The next generation. *arXiv preprint arXiv:2007.04074*. 24

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970. 24

Fonseca Abreu, J. V., Ghedini Ralha, C., and Costa Gondim, J. J. (2020). Twitter bot detection with reduced feature set. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. xi, xii, 1, 4, 37, 42, 46, 48, 54, 57, 58, 59, 60

Foundation, P. S. (2021). pickle — python object serialization — python 3.9.2 documentation. `https://docs.python.org/3/library/pickle.html`. (Accessed on 03/18/2021). 46

Freitas, C., Benevenuto, F., Ghosh, S., and Veloso, A. (2015). Reverse engineering socialbot infiltration strategies in twitter. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 25–32. IEEE. 2

Gao, H., Chen, Y., Lee, K., Palsetia, D., and Choudhary, A. N. (2012). Towards online spam filtering in social networks. In *NDSS*, volume 12, pages 1–16. 32, 80

Gilani, Z., Farahbakhsh, R., Tyson, G., Wang, L., and Crowcroft, J. (2017). Of bots and humans (on twitter). In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 349–354. 33

Gorwa, R. and Guilbeault, D. (2020). Unpacking the social media bot: A typology to guide research and policy. *Policy & Internet*, 12(2):225–248. 8

Gui, T., Liu, P., Zhang, Q., Zhu, L., Peng, M., Zhou, Y., and Huang, X. (2019). Mention recommendation in twitter with cooperative multi-agent reinforcement learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 535–544, New York, NY, USA. Association for Computing Machinery. 7, 38

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182. 21

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1. 18, 19

Howard, P. N. and Kollanyi, B. (2016). Bots, #strongerin, and #brexit: Computational propaganda during the UK-EU referendum. *CoRR*, abs/1606.06356. 2, 9

Howden, N., Rönnquist, R., Hodgson, A., and Lucas, A. (2001). Intelligent agents - summary of an agent infrastructure. In *In 5th International conference on autonomous agents*. 14

Jin, H., Song, Q., and Hu, X. (2019). Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. 24, 25

Karzan, M. A. and Erdogan, N. (2013). Topic based agent migration scheme via publish/-subscribe paradigm. *International Journal of Information and Education Technology*, 3(3):290. 44

Latah, M. (2020). Detection of malicious social bots: A survey and a refined taxonomy. *Expert Systems with Applications*, 151:113383. x, 1, 5, 8, 28, 29, 34, 38

Lee, K., Eoff, B., and Caverlee, J. (2011). Seven months with the devils: A long-term study of content polluters on twitter. In *International Conference on Weblogs and Social Media (ICWSM)*. AAAI. 29, 31, 33, 34, 61, 80

Liu, S., Hooi, B., and Faloutsos, C. (2019). A contrast metric for fraud detection in rich graphs. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2235–2248. 30

Lutz, M. (2013). *Learning python: Powerful object-oriented programming.* " O'Reilly Media, Inc.". 46

Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., and Tesconi, M. (2019). RT-bust: Exploiting Temporal Patterns for Botnet Detection on Twitter. In *Proceedings of the 10th ACM Conference on Web Science*, WebSci '19, pages 183–192, Boston, Massachusetts, USA. Association for Computing Machinery. 2, 8, 30, 78

Melo, L. S., Sampaio, R. F., Leão, R. P. S., Barroso, G. C., and Bezerra, J. R. (2019). Python-based multi-agent platform for application on power grids. *International Transactions on Electrical Energy Systems*, 29(6):e12012. 15, 43

Menze, B., Kelm, B., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W., and Hamprecht, F. (2009). A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC bioinformatics*, 10:213. 22, 23

Miller, Z., Dickinson, B., Deitrick, W., Hu, W., and Wang, A. H. (2014). Twitter spammer detection using data stream clustering. *Information Sciences*, 260:64–73. 1, 30, 33, 34, 79

Minnich, A., Chavoshi, N., Koutra, D., and Mueen, A. (2017). Botwalk: Efficient adaptive exploration of twitter bot networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, page 467–474, New York, NY, USA. Association for Computing Machinery. 1, 30, 79

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill, New York. 17, 18

Mitnick, K. D. and Simon, W. L. (2003). A arte de enganar. *São Paulo. Person Education do Brasil Ltda.* 2

Mora Lizán, F. J. and Rizo-Maestre, C. (2017-05). Intelligent buildings: Foundation for intelligent physical agents. 42

Munson, M. A. (2012). A study on the importance of and time spent on different modeling steps. *ACM SIGKDD Explorations Newsletter*, 13(2):65–71. 24

Nembrini, S., König, I. R., and Wright, M. N. (2018). The revival of the Gini importance? *Bioinformatics*, 34(21):3711–3718. 22

O'Brien, P. D. and Nicol, R. C. (1998). Fipa—towards a standard for software agents. *BT Technology Journal*, 16(3):51–59. 43

Oentaryo, R. J., Murdopo, A., Prasetyo, P. K., and Lim, E.-P. (2016). On profiling bots in social media. In Spiro, E. and Ahn, Y.-Y., editors, *Social Informatics*, pages 92–109, Cham. Springer International Publishing. 32, 79

Olson, R. S. and Moore, J. H. (2019). *TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning*, pages 151–160. Springer International Publishing, Cham. 4, 24, 25

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. 24, 50, 53, 55, 58, 59

Pimentel, J. and Castro, J. (2018). pistar tool – a pluggable online tool for goal modeling. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 498–499. 13, 38

Poole, D., Mackworth, A., and Goebel, R. (1997). *Computational Intelligence: A Logical Approach*. Oxford University Press, Inc., USA. 9

Powers, D. M. (2011). Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63. 25, 58

Provost, F and Fawcett, T (1997). 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*, pages 43–48. 59

RedaçãoCorreio24horas (2020). No twitter, neymar parabeniza bayern e lamenta derrota na final - jornal correio | notícias e opiniões que a bahia quer saber. `https://www.correio24horas.com.br/noticia/nid/no-twitter-neymar-parabeniza-bayern-e-lamenta-derrota-na-final/`. Accessed 09/09/2020. 1

Rodríguez-Ruiz, J., Mata-Sánchez, J. I., Monroy, R., Loyola-González, O., and López-Cuevas, A. (2020). A one-class classification approach for bot detection on twitter. *Computers & Security*, 91:101715. x, xii, 1, 7, 16, 25, 32, 36, 49, 50, 51, 52, 53, 54, 57, 59, 67, 78

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition. x, 7, 9, 10, 11, 15, 16, 18, 19, 20, 21, 22

Russell, S. and Norvig, P. (2013). Inteligência artificial. *GEN LTC, 3ª Edição. São Paulo, Brazil*. x, 17

Ruz, G. A., Henríquez, P. A., and Mascareño, A. (2020). Sentiment analysis of twitter data during critical events through bayesian networks classifiers. *Future Generation Computer Systems*, 106:92–104. 7

Sayyadiharikandeh, M., Varol, O., Yang, K.-C., Flammini, A., and Menczer, F. (2020). Detection of novel social bots by ensembles of specialized classifiers. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 2725–2732, New York, NY, USA. Association for Computing Machinery. 1, 34, 36, 68, 78

Schneier, B. (2001). Secrets & lies: Digital security in a networked world. *International Hydrographic Review*, 2(1):103–104. 2

Shafahi, M., Kempers, L., and Afsarmanesh, H. (2016). Phishing through social bots on twitter. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 3703–3712. IEEE. 2

Sokolova, M., Japkowicz, N., and Szpakowicz, S. (2006). Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. In Sattar, A. and Kang, B.-h., editors, *AI 2006: Advances in Artificial Intelligence*, pages 1015–1021, Berlin, Heidelberg. Springer Berlin Heidelberg. 25, 26, 58

Stringhini, G., Kruegel, C., and Vigna, G. (2010). Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, page 1–9, New York, NY, USA. Association for Computing Machinery. 2, 8, 9, 31, 80

Subrahmanian, V. S., Azaria, A., Durst, S., Kagan, V., Galstyan, A., Lerman, K., Zhu, L., Ferrara, E., Flammini, A., and Menczer, F. (2016). The darpa twitter bot challenge. *Computer*, 49(6):38–46. 8

Tavares, G. and Faisal, A. (2013). Scaling-laws of human broadcast communication enable distinction between human, corporate and robot twitter users. *PloS one*, 8(7). 2, 9, 31, 79

Tharwat, A. (2018). Classification assessment methods. *Applied Computing and Informatics*. 25, 26, 58

Theodoridis, S. and Koutroumbas, K. (2008). *Pattern Recognition, Fourth Edition*. Academic Press, Inc., USA, 4th edition. 19

Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, C. B., and Farivar, R. (2019). Towards automated machine learning: Evaluation and comparison of automl approaches and tools. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1471–1479. IEEE. 23, 24

Tuggener, L., Amirian, M., Rombach, K., Lörwald, S., Varlet, A., Westermann, C., and Stadelmann, T. (2019). Automated machine learning in practice: state of the art and recent results. In *2019 6th Swiss Conference on Data Science (SDS)*, pages 31–36. IEEE. 24

Twitter Inc. (2020a). Api documentation|docs|twitter developer. `https://developer.twitter.com/en/docs/twitter-api`. Accessed 09/17/2020. 8

Twitter Inc. (2020b). Help center. `https://help.twitter.com/en`. Accessed 09/17/2020. 7, 8, 38

Varol, O., Ferrara, E., Davis, C. A., Menczer, F., and Flammini, A. (2017). Online Human-Bot Interactions: Detection, Estimation, and Characterization. In *11th International AAAI Conference on Web and Social Media (ICWSM)*. 1, 8, 30, 33, 34, 36, 68, 79

Wang, A. H. (2010). Detecting spam bots in online social networking sites: A machine learning approach. In Foresti, S. and Jajodia, S., editors, *Data and Applications Security and Privacy XXIV*, pages 335–342, Berlin, Heidelberg. Springer. 2, 30, 80

Weiss, G., editor (2013). *Multiagent systems.* MIT press, 2nd edition. 9, 11

Wooldridge, M. (2009). *An introduction to multiagent systems.* John Wiley & Sons Ltd, UK, 2nd edition. 2, 9, 11

Yang, C., Harkreader, R., and Gu, G. (2013). Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8(8):1280–1293. 30, 32, 33, 46, 50, 55, 57, 79

Yang, K.-C., Varol, O., Hui, P.-M., and Menczer, F. (2020). Scalable and generalizable social bot detection through data selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1096–1103. 54

Appendix A

Literature Review Table

This appendix contains an extended version of the table, which synthesizes the works read on the literature review.

Table (A.1) Full comparison of literature review works

Reference	Approach/ ML technique	Performance measures	Dataset/ Bot type	Benefits	Problems
Sayyadiharikandeh et al. (2020)	supervised / ensemble of RF	public/ any	applies an ensemble of specialized classifiers which according to the authors improves the detection, public dataset, detection framework available via website	uses over 1,200, only uses one ML algorithm on the ensemble	
Rodríguez-Ruiz et al. (2020)	supervised / BN, J48, RF, Adaboost, Bagging, KNN, LR, MLP, NB, SVM, BTPM, BRM, OCKRA, ocSVM	AUC	public/ any	13 features, technique is more effective against heterogeneous bot types, tested ten different algorithms	one-class classifiers are worse then the specific classifiers, was not tested online,
Alharthi et al. (2019)	semi-supervised/ Label propagation, Label spreading	AUC, F1-score, precision, accuracy, recall, specificity	not informed/ Spam	16 Features, evidences that botnet participants have different roles,	Does not inform if the dataset is public, was not tested online
Mazza et al. (2019)	unsupervised/ LSTM	precision, recall, accuracy, MCC, F1-score	public/ retweets	12 features, projected visualization methodology ratifies the classifier, public dataset, identified new botnets, good result evaluation compared to other works	focus on retweet bots, was not tested online
Al-Qurishi et al. (2018)	supervised/ IR, RF, J48, regression, SVM	accuracy, incorrect classified instances, Kappa Statistic, mean absolute error , true-positive rate , false-positive rate	not informed/ any	75 features, demonstrates technique to find keywords in a specific application domain, experiments with five ML algorithms	does not inform if dataset is public, was not tested online, Twitter and youtube analysis is confusing, uses features with high computational costs
Al-Qurishi et al. (2018)	unsupervised/ Deep learning model	accuracy and loss	not informed/ any	80 features, allows the improvement of the deep learning model by entering new data	does not inform if dataset is public, was not tested online
Beğenilmiş and Uskudarli (2018)	supervised/ RF, LR, SVM	accuracy, precision, recall, F1-score, AUC	public / any (coordinated action in elections)	23 features, experiments with three ML algorithms, public dataset, open source	was not tested online
Chen and Subramanian (2018)	unsupervised/- clustering	-	public/ spam (focus on URLs)	one feature[1], tested online, detection framework available via REST	focus on bots that post malicious URLs, use of few assessment methods

Table A.1: Continues on next page

[1]Despite not showing the features, the authors implies that only text is analyzed in clustering

Table A.1: Continued from previous page

Reference	**Approach/ ML technique**	**Performance measures**	**Dataset/ Bot type**	**Benefits**	**Problems**
Fazil and Abulaish (2018)	supervised/ RF, DT, BN	recall, false-negative rate, F1-score	not informed/ spam	8 features, boa avaliação de resultado em comparação com outro trabalho, features baseadas nos seguidores são difíceis de serem contornadas	does not inform if dataset is public, was not tested online
Fernquist et al. (2018)	supervised/ RF	accuracy, precision, recall, F1-score	public/ any	140 features, public dataset	Evaluated period of elections is not the most influential among voters, was not tested online
Minnich et al. (2017)	unsupervised/ Ensemble of unsupervised anomaly detection approaches	precision, detection rate	public/ any	130 features, open source, public dataset, comparison with other works, feature selection through adaptative approach	use of few assessment methods, high computational cost
Varol et al. (2017)	supervised/ RF	AUC, true-positive rate and false-negative rate	public/ any	public dataset, detection framework available via website	use of 1140 features
Oentaryo et al. (2016)	supervised/ NB, RF, SVM, LR	precision, recall, F1-score	not informed/ various	358 features, experiments with four ML algorithms, classification of good bots, multivalued classification (humans and bots: transmission, consumption and spam)	does not inform if dataset is public, worse than exclusive malicious bot classifiers, was not tested online
Dickerson et al. (2014)	supervised/ Gaussian Naïve Bayes, SVM, RF, extremely randomized trees, AdaBoost, gradient boosting	precision, recall, AUC	not informed/ any	31 features, suggests that sentiment analysis improves the classification, experiments with five ML algorithms, determines bot and human feeling patterns	does not inform if dataset is public, was not tested online, uses features that have a high collection cost
Miller et al. (2014)	unsupervised/ DenStream, StreamKM++	specificity, true-positive rate, accuracy, balanced accuracy, precision, F1-score, recall, AUC	not informed/ spam	107 features, demonstrates that using tweet characters as features can improve detection, experiments with three ML algorithms	Does not inform if the dataset is open, use of tweet characters as features as proposed does not apply to languages with different alphabets
Ahmed and Abulaish (2013)	supervised/ NB, Jrip, J48	true-positive rate and detection rate	not informed/ spam	14 features, Performs analysis of spam campaigns	does not inform if dataset is public, was not tested online
Tavares and Faisal (2013)	supervised/ NB	correctness rate	public/ spam	four temporal features, public dataset, open source, multivalued classification (bots, organizational profile and humans)	results evaluation only cites the correctness rate
Yang et al. (2013)	supervised/ NB, RF, DT, Decorate	true-positive rate, detection rate, F1-score	not informed/ spam	10 features, good result evaluation compared to other works, formalizes the robustness of the features, experiments with four ML algorithms	does not inform if dataset is public, was not tested online, the weights attributed to features are questionable
Chu et al. (2012)	supervised/ RF	true-positive rate	not informed/ spam	11 features, multivalued classification (bots, ciborgs and humans), uses entropy of tweets as a component of detection	does not inform if dataset is public, was not tested online, results evaluation only cites the true-positive rate

Table A.1: Continues on next page

Table A.1: Continued from previous page

Reference	**Approach/ ML technique**	**Performance measures**	**Dataset/ Bot type**	**Benefits**	**Problems**
Gao et al. (2012)	supervised/ SVM, DT	true-positive rate and false-positive rate	not informed/ spam	six features, conceives a solution that can (according to the authors) perform real-time detection without requiring frequent training	does not inform if dataset is public, was not tested online
Lee et al. (2011)	supervised/ RF and other 30 classifiers	accuracy, F1-score, AUC	public/ any	16 features, methodology for assembling bots dataset, public dataset, true-positive verification of results through the banishment of accounts , experiments with various ML algorithms (Weka)	only presents RF results
Stringhini et al. (2010)	supervised/ RF	true-positive rate and false-negative rate	not informed/ spam	six features, verification of results through the banishment of accounts, identified spam campaigns	does not inform if dataset is public, results assessment only cites true-positive and false-negative rates
Wang (2010)	supervised/ DT, NN, SVM, NB	precision, recall, F1-score	not informed/ spam	six features, experiments with four ML algorithms	does not inform if dataset is public, was not tested online