UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Content Polluters Detection on Twitter: A Machine Learning Approach

Ignacio Puche Lara

Tutor: Rafael Llobet Azpitarte

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2019-20

Valencia, 5 de julio de 2020

Resumen

Uno de los principales usos diarios de internet son las redes sociales. En estas plataformas son comunes las cuentas automatizadas no legítimas o también llamadas: "Contaminadores de Contenido". La existencia de dichos contaminadores resulta ser un problema tanto para los administradores de la plataforma (por conseguir ciertos beneficios a costa incumplir las políticas del servicio) como para los usuarios. Es por ello, que en este proyecto se propone el uso de modelos de Aprendizaje Automático para su detección. Durante el desarrollo del mismo, se analiza la situación actual de Twitter respecto a estos contaminadores. Seguidamente, se evaluan las técnicas actuales acerca de esta rama de la Inteligencia Artificial así como el estado del arte en este ámbito. Además, se realizan diversos experimentos haciendo uso de estas técnicas, entrenando modelos de Aprendizaje Automático con distintos conjuntos de datos de uso público sobre Contaminadores de Contenido. Concretamente se realizan tres aproximaciónes: detección a nivel de usuario, a nivel de tweet y por último, una combinación de los mismas. Finalmente, se concluye que estas técnicas son fructíferas respecto a la detección de estos usuarios no legítimos, destacando el rendimiento de modelos como los Bosques Aleatorios o las Redes Neuronales.

Resum

Un dels principals usos diaris d'Internet són les xarxes socials. En aquestes plataformes són comuns les comptes automatitzades il·legítimes o també anomenades «Contaminadors de Contingut». La existència de aquests contaminators resulta ser un problema tant per als administradors de la plataforma (per tal d'obtindre beneficis a costa d' incomplir les polítiques del servei) com per als usuaris. És per això, que en aquest projecte es proposa l'ús de models d'aprenentatge automàtic per a la seua detecció. Durant el desenvolupament del mateix, s'analitza la situació actual de Twitter respecte a aquests contaminadors. Seguidament, s'avaluen les tècniques actuals sobre aquesta rama de la intel·ligència artificial així com l'estat de l'art en aquest àmbit. A més, es realitzen diversos models d'aprenentatge automàtic amb diferents conjunts de dades d'ús públic sobre Contaminadors de Contingut. Concretament es realitzen tres aproximacions: detecció a nivell d'usuari, a nivell de tweet i, per últim una combinació d'aquestes. Finalment, es conclou que aquestes tècniques són fructíferes respecte a la detecció d'aquests usuaris il·legítims, destacant el rendiment de models com els «boscos aleatoris» o les «xarxes neuronals».

Abstract

One of the main daily uses of the internet are social networks. On these platforms, there are commonly automated non-legitimate accounts, the so-called "Content Polluters". The existence of these polluters is a problem for both platform administrators (since they acquire benefits by violating the terms of use) and users. Hence, this project proposes the use of Machine Learning models in order to detect and identify them. During its development, the current situation of Twitter regarding these polluters is analyzed. Next, both the currently employed techniques of this branch of Artificial Intelligence and the state of the art in this ambit are examined. Moreover, some experiments are done with the use of these techniques, training Machine Learning models with different datasets of public use about Content Polluters. Concretely, three approaches take

place: user level detection, tweet level detection and finally, a combination of both. In the end, the conclusion is that these methods are appropriate with respect to the detection of this non-legitimate users, emphasizing the good performance of models such as Random Forests or Neural Networks.

Contents

List of Figures

List of Tables

Part I

Project Report

Chapter 1

Introduction

1.1 Motivation

1.1.1 The General Problem

Nowadays, we can find ourselves in a world where digital technologies play a fundamental role in our daily lives. This fact can be seen as a really big difference in comparison to a couple of decades ago. By that time, the internet was a brand new invention and individuals without any relation to the IT field could find it difficult to understand its use and functionalities. From that point of view, there is no doubt about how impressive the evolution of both the telecommunications and computer sciences fields has been.

Technologies have improved so massively that experts have already defined the concept of Digital Revolution [1]. Within the wide range of effects caused by this new Revolution, we could state that one of the most notable is the fast flow of information, which is reinforced by the activity of social media. The popularity of these sites has entailed the appearance of several companies which provide these services, usually with different format and/or clients such as Facebook, Twitter, Instagram, Whatsapp, Youtube etc.

Social media allows people to share any kind of data (from plain text or links to multimedia files such as images or videos) in a matter of seconds. Moreover, this is even easier when these online services are accessed by smartphones, which give users the ability to use them anywhere at anytime. Although people usually connect to this sites for pure entertainment, e.g. to have a simple conversation with friends about the plans for the next weekend, there is a high percentage of users which share their opinions publicly, often supported by links to external articles, videos or similar.

Furthermore, these applications make it possible to communicate with new people without any previous contact in the physical world. Therefore, these published opinions can reach other users who may accept that opinion or start a discussion. In other words, world citizens with access to the internet, can use social media to see the opinions of other users, express their own, and discuss about any subject.

It is reasonable to think that the number of advantages of this flexibility in communications is high, since it encourages freedom of expression. In addition, it allows the observation of the immediate reaction of the users about a global event by everybody e.g. a natural catastrophe or a political

debate of a country during elections.

However, not all the points concerning this are positive. One of the dangers of this liberty may result in global misinformation. Social media can be seen as a medium which facilitates the spread of fake news, alternative facts and hoaxes to manipulate the rest of the users. Besides, if we consider that these publications can be done automatically (by the so-called ”bots”), then, this non-real information can be repeated by thousands of accounts reaching large amounts of people in reduced time.

The points commented in the previous paragraph are not suppositions, considering that similar things have already occurred: During the elections of the U.S. in 2016, social bots were discovered to generate big amounts of content to influence in the U.S. inhabitants opinions and condition the results [2].

In addition, due to the global pandemic caused by COVID-19 in 2020 the circulation of fake information has augmented, taking advantage of the fear and confusion caused by the disease in the world population. Some of the main observed cases of these fake news are: misinformation around vaccines, appealing to the anti-vaccines movement [3], and movements whose intention is to make people believe that 5G mobile networks are responsible for transmitting the virus using pseudo-scientific arguments [4]. To add a final example, we could emphasise the recent set of fake accounts that has massively interacted with the Spain’s Ministry of Health in relation to some publications concerning COVID-19 on Facebook [5].

All this set of fake accounts with different malicious aims can be described as the concept of **”Content Polluters”**. On the other hand, the rest of profiles with good intentions are considered **”Legitimate Users”** [6]. This distinction is necessary because there are bots in these networks whose objective is to provide entertainment or other purposes accepted by the website policies. These last mentioned benign accounts usually declare that they are bots in their description so the rest of consumers can be conscious of it.

1.1.2 Twitter as the Target of the Study

In spite of the fact that the problem may seem to affect a variety of different social media, Twitter is one of the most vulnerable because of its architecture and design. On this site, the users are able to share publications of other members by the mechanism of ”Retweeting”. Furthermore, when a particular subject is popular in the comments of the users, then it is shown in the Trending Topics list, popularising it even more. Hence, if a large amount of coordinated Content Polluters write about a particular concept, then their influence may increase in a spectacular way, becoming more dangerous.

In other words, the chances that a user observes a publication without previously searching for it are really big. This is different in other similar services e.g. Instagram. Instagram’s mechanism only shows to each user the content that they follow, which increases the difficulty for content polluters to reach people.

Although some people may use Twitter to have fun and watch the last viral video, a notable number of users connect to Twitter in order to watch the news or carry out social activism [7].

Twitter has been chosen in this project because of its API, which allows to collect data for research purposes. In addition, the availability of several public datasets related to content polluters, which

makes wider the range of possibilities to develop this work.

1.1.3 The Solution

It is clearly obvious that some methods are necessary to fight against these automated fake accounts, which violate the Terms of Service stated by social media companies. From their appearance, several methods have been used in order to restrain the bots, e.g. searching for spam keywords or similar. However, the advance in Machine Learning techniques in the recent years due to the improvement in hardware, suggests that it may be a good solution to this problem.

An example of the successes of Machine Learning in terms of classification tasks was the ability to predict if a person suffered depression regarding their Instagram accounts [8]. By analyzing, different features of the users, such as colours in each image, number of faces per photography, number of comments etc., the results were quite impressive. Machine Learning algorithms are able to find hidden patterns between large amounts of data, which can be difficult to be found by humans. Thus, as it was declared before, they seem to be an appropriate selection for this problem.

1.1.4 Personal Level Interest

In terms of the personal benefits obtained because of producing this study we can emphasise the acquired knowledge related to Machine Learning. This goal is really important because, as it was stated in the above paragraphs, it is a technology that has had a big impact in the recent years and promises big results and advances. This is reinforced by the fact that companies in general have large amounts of data to deal with. It is fundamental then, to have Data Science notions and this project can be seen as a chance to get them.

Moreover, the consciousness about the damages that content polluters can cause to society is another positive point. Once this project is finished, the conclusions obtained may help me in the future to distinguish between real or fake/spam posts anywhere in the internet. In addition, these detailed notions can be used to help non-IT experts to deal with this kind of ”online junk”.

At a more technical and practical level, it is expected to improve my programming skills while handling specific tools designed for Machine Learning tasks e.g. concrete libraries for building and evaluating classification models.

1.2 Objectives

1.2.1 Main Objective

The target of this project consists in evaluating if Machine Learning models are able to distinguish between Content Polluters and Legitimate Users on Twitter, given some profile input data and determine which techniques are more efficient and effective in this kind of problem.

1.2.2 Individual Objectives

In order to achieve the main objective of the project, the organization of the corresponding work has been divided into several subtasks. Each subtask has its own individual objective:

- Doing some research related to Twitter as a social media platform. This involves getting to know its main features, concepts and possibilities in order to gain a basic overview of what methodologies can be applied.
- Carrying out one first theoretical study related to the Machine Learning field and its models. Understanding how they work will allow us to decide which specific techniques are more adequate for our problem. Of course, choosing the models according to their theoretical background will help to confirm that the future results are valid, and not produced by a source of randomness.
- After the different Machine Learning models and Twitter itself have been analysed, then appropriate Twitter Datasets must be obtained. Therefore, this includes some investigation of the datasets publicly available or consider if it is necessary to obtain our own dataset.
- Executing some experiments and analysis to the collected datasets using the selected Machine Learning models. Different parameters and configurations will be tried in order to explore the particular possibilities of each model.
- Organizing all the results generated by the experiments. The results must be examined so as to extract relevant information. In other words, this subtask can be seen as post processing the outcome of our final models.
- Doing a general evaluation and derive concrete and accurate conclusions from the results. Then, it will be possible to decide if these set of techniques is effective enough or if some improvements are required.

1.3 Organization of This Report

Once the problem and its motivation have been introduced, in the second chapter the online social network Twitter is analysed. Moreover, the Content Polluter concept is explained in detail. Next, in the third chapter there is a theoretical description of Machine Learning basics such as the difference between supervised and unsupervised learning, the dissimilarity between classification and regression, and metrics for evaluation. Subsequently, in the fourth chapter, all the relevant Machine Learning models for this study are described as a theoretical introduction for the project.

The following part of this report includes the chapters which communicate the concrete practical activities carried out in the project. In the fifth chapter, the overall methodology is described, presenting the datasets used and the chosen tools. Continuing into the sixth chapter, all the experiments with their respective results are deeply examined. Finally, the seventh chapter contains the resulting conclusions of this study, and several future projects are proposed.

Chapter 2

Twitter Overview and Content Polluters.

2.1 Twitter Overview

2.1.1 General Description

Twitter is nowadays one of the most popular platforms in terms of social media. Its system for sharing information is based on the publications of the so-called ”**tweets**” by its users. These are simple string characters with a short length which may include:

- **Multimedia content** which can be a video, an image, a gif, etc.
- **Links** to external URL sites or to other tweets.
- **Mentions** of other users names. These are links to other accounts, usually related to the content of the tweet. They are preceded by an ”@” symbol.
- **Hashtags**. Specific keywords that make reference to an idea that summarizes or has any connection to the content of the tweet. Twitter has a mechanism that analyses the most common hashtags in the last published tweets all over the world in order to find the **”Trending Topics”** and popularise the related tweets. In this way, the users can easily see these Trending Topics in the site and what the people is posting about them. These keywords start by a ”#” character.

Customers in this social media platform can interact with each other through an architecture based on ”following”. Users are able to follow other accounts. When one profile sends one tweet, then all of its followers will immediately see it. Besides, these followers have the possibility of **retweeting** a tweet, spreading its content through its network. In addition, there is a section called **”Explore”** where all the tweets related to the Trending Topics of the moment can be found. It is not difficult to appreciate that these features make it easier to spread information as it was stated in Chapter 1.

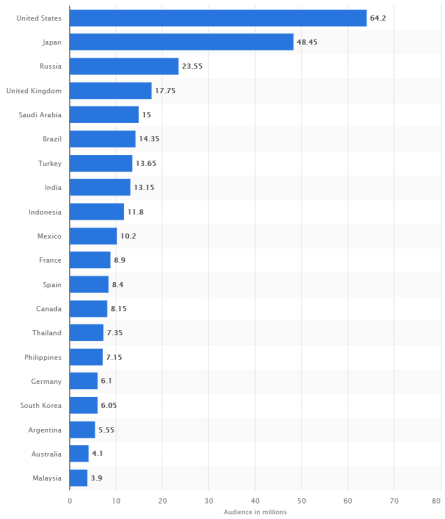

Figure 2.1: Geographical statistics in Twitter. Source: [9]

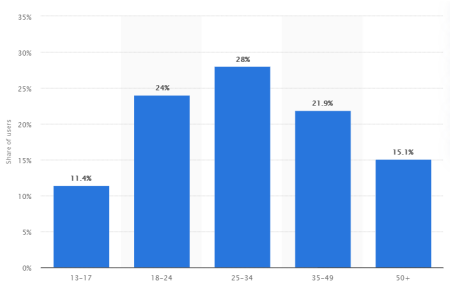

Figure 2.2: Ages of Twitter users gathered in groups. Source: [11]

2.1.2 Social Impact and Statistics

15 years approximately after its beginning in 2006, Twitter has become widely used around the world. In terms of geographical statistics, the largest number of members are from the United States with 64.2 millions, followed by Japan, Russia, United Kingdom and Saudi Arabia with 48.45, 23.55, 17.75 and 15 millions respectively as it is shown in Figure 2.1.

Analyzing the age of the users, the group within the range 25-34 years is dominant according to Figure 2.2. However, the percentage of users of the contiguous groups do not really differ significantly. These data reveals that young users are not predominant set of consumers such as in other social networks.

This variability of the ages of users may be the reason why the number of the topics present in Twitter's content is large. Some of these fields are politics, science, business, journalism, relationships and celebrities [10]. This suggests that this network is not used entirely for entertainment. Moreover, there is a clear diversity regarding Twitter community in terms of interests and intentions.

2.1.3 Twitter API

2.1.3.1 Description

One of the advantages of Twitter is its flexible API, which is freely accessible for developers. It specifically allows to collect data and metadata from users by their ID number or user name, to retrieve tweets etc. This is a fundamental tool in this work to complete Twitter datasets with appropriate data.

The API also includes the possibility to make publications in an automated manner and deal with all the ”write” functions that a Twitter account can carry out. However, these functions are not going to be used in this study, as it is only needed to collect information by using ”read” functions.

In spite of the fact that the standard API contains download rate limits and 15 minutes windows for each request, this is enough since the amount of data to be collected is not extremely big (more details in chapter 5).

2.1.3.2 Ethics and Responsibility

Current Twitter’s Developer Agreement and Policy [12] emphasises that it is forbidden to publicly share datasets with complete features. Concretely, if a researcher wants to publish one dataset, he or she can only post a list containing user and/or tweet IDs so that the researcher must complete the dataset him/herself. However, after this project has concluded, none dataset will be published. The datasets to be used will be lists of user and tweet IDs complemented using the API and complete public datasets which where legally published before Twitter added this restriction in the recent years.

In addition, it is prohibited to use the API in order to monitor and spy users doing several requests repeatedly about the same profile or similar. Nevertheless, in this activity, only the data concerning the users listed in our used datasets will be gathered.

Moreover, when the API key has been requested to Twitter Developers section, these points explained here have been explicitly detailed to ensure the correctness of the API use.

2.2 Content Polluters

2.2.1 Content Polluter: Definition

As it was briefly introduced in chapter 1, Twitter profiles can be separated in two groups: **Content Polluters** and **Legitimate Users** (following the interpretation provided in [6]). Content Polluters are accounts whose intentions are to fill the social network with fake information or spam, falsely inflate the number of followers of a concrete user, among others.

Furthermore, it is assumed that Content Polluters actions are automated. In other words, through this report, Content Polluter can be defined as malicious bots. Since their actions violate Twitter’s Terms of Use, there is a need of properly detecting and eliminating them.

The rest of profiles, including bots which do not contribute to negatively alter the reality in the platform will be considered as Legitimate Users.

2.2.2 State of the Art

In the recent years, there have been previous researches which try to detect Content Polluters in terms of academic literature. Besides, there is a wide variety of approaches since Content Polluters are constantly evolving so as not to be caught. Thus, academia researchers must persist in finding ways to defend social networks from Content Polluters.

Examining some of the most notable papers, we firstly find the work done by K. Lee et. al. in [6], which collected complete and accurate datasets differentiating between Content Polluters and Legitimate Users. These datasets were obtained using a honeypot strategy: 60 accounts were created and they only interacted to each other without interfering with Legitimate Users. Posting tweets gathered from Twitter's public timeline during seven months, these honeypots tempted 36,043 accounts, which were later analysed to conclude that they were Content Polluters. Given that the sequence of published tweets by each account had no sense, humans were not supposed to be tempted. The results obtained in regard to classification by using Machine Learning techniques, supervised learning in particular (more information in Chapter 3) were outstanding.

Basically, the Machine Learning application with supervised learning has been the most common solution. Some examples such as [13] by C. Yang et. al., [14] by Subrahmanian et. al. etc. In addition, Indiana University Bloomington developed an online system called **BotOrNot?** whose aim is to scan a Twitter account introduced by the user and predict the label bot or human [15] using several datasets previously used in these supervised learning approaches.

In 2012, a crowdsourcing system (method which uses contributors, which are external to the project, to carry out a task) was conducted by Wang et. al. in [16] to verify if humans are able to distinguish between real accounts and bots. The results were really favourable even though it is not a scalable solution, since bots can rapidly replicate with no cost while humans cannot accelerate the classification of a list of accounts.

Continuing with more crowdsourcing techniques, Cresci et. al. in [17] (2017) used a similar system to encourage the collaborators to differentiate between spambots and genuine accounts. The difference with the previous paper is that Content Polluters from a new spambot wave called "social-spambots" were included and mixed with traditional spambots. While the participants in the crowdsourcing could label traditional spambots correctly, social-spambots were potentially misclassified. The conclusion reveals that humans are good enough at detecting traditional spambots. Yet, the experience in discriminating between humans and this new wave of social-spambots is disastrous. This is another evidence which supports that Content Polluters are constantly evolving.

Lately, in 2018, Kudugunta et. al. in [18] made use of Deep Learning based on a Long Short-Term Memory (LSTM). This is a special kind of Artificial Neural Network which allows to assess a sequence of words (tweets) and extract information from its order. In other words, the context of a text can be approximately obtained.

A quick view over the most modern practices, Cresci et. al. research in [19] suggest unsupervised learning methods to defeat coordinated bots which post large amounts of human-generated content, complicating their detection. Their work consists in encoding the sequence of actions of a large dataset of accounts as character strings, defining it as the "Digital-DNA" of a Twitter profile. These character strings are compared to each other in order to find similarities. Basically, when a long substring matches in different account sequences, then it indicates that these accounts are acting in

a coordinated manner, and hence, are not human.

Chapter 3

Machine Learning Fundamentals.

3.1 Definition

Machine Learning can be defined as the branch of Artificial Intelligence (AI) dedicated to build intelligent models whose ability is to automatically learn how to do a specific task. The main difference with a normal computer program consists in the fact that when solving a specific task, a simple program executes an algorithm with some rules designed to solve this specific problem. However, when a Machine Learning solution is applied, the algorithm must obtain this rules after making some observations. In other words, the algorithm is **learning**.

3.2 Types of Learning

Depending on the kind of problem to be solved and how the observations are made by the models we can identify two main different ways of learning: supervised and unsupervised learning [1].

3.2.1 Supervised Learning

In this paradigm of learning, there is a **training set**, formed by n input-output pairs, usually called **samples**:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \tag{3.1}$$

where each y_i was generated by an unknown function $y = f(x)$ [20]. The objective is to use models in order to obtain another function which approximates y so that we will be able to **predict** unknown values for certain input x_j. In other words, supervised Machine Learning models have the ability to find hidden patterns between the inputs and the outputs by evaluating a large set of examples. This process of approximating y is commonly known as "Training the model".

Concretely, the output values y_i are called **target values** since they are the **labels** which are needed to be predicted by the model. On the other hand, the input values, which can be more than one per

[1] There are also other learning methods such as reinforcement learning or statistical learning. Nevertheless, these are not considered because of their lack of relevance in this study.

output value, are denominated as **features**.

It is important to emphasize the reliability of the dataset quality. For acquiring acceptable predictions, a really large number of samples is required. Basically, the more samples are available to the model, the better result will be reached, which makes sense given that when a dataset is bigger, more information is contained in it.

3.2.2 Unsupervised Learning

In this other methodology of learning, another training dataset is used in the process of learning. However, the difference with supervised learning is the absence of output values in the training dataset. The most common technique employed here is **clustering**. The task of a clustering model is to group the inputs according to common patterns shared by certain subsets of input. In other words, the model learns to distinguish between subclasses in the training dataset without receiving any feedback during the process. This technique is useful when the target label in a dataset is unknown. In terms of how this models work, we can state that the similarity between the samples in each cluster is usually measured by using metrics such as Euclidean or probabilistic distance.

As supervised learning is the paradigm used in this study, from this point all the descriptions will be focused on its context. In the case that anything is referred to unsupervised learning, it would be explicitly written.

3.3 Classification vs Regression

With regard to the type of target values, two different tasks can be identified. If the values are **discrete**, that implies that the separability of the samples is well-defined in **classes**, so this problem consists in **classification**. Therefore, if the target value is a qualitative feature, similarly to our problem (Content Polluter or Legitimate User), then the different possible labels can be numbered so that a classification Machine Learning model can be used.

Otherwise, if the target numbers are continuous and real values, then the problem is denominated as **regression**. One example of this kind of problem could be ”linear regression” whose aim is to approach a linear function to a set of samples in order to later use this function to predict new values.

Some models can be used for both tasks with small modifications. Moreover, classification models may generate real values rather than discrete ones. In this case, output values are usually the probability of a sample belonging to a specific class.

Given that Content Polluters and Legitimate Users are clearly separated labels, this is a classification problem. Therefore, all the following explanations will be in terms of classification models.

3.4 Evaluation Techniques

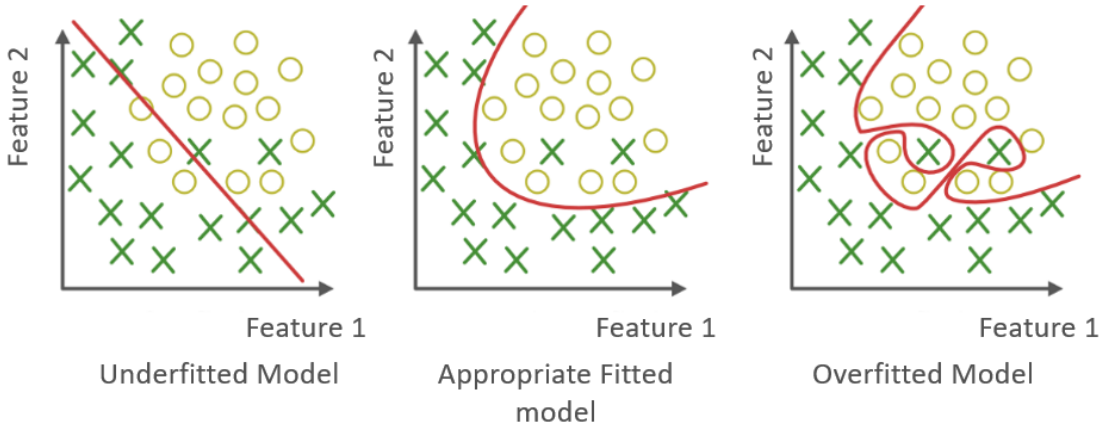

Figure 3.1: Examples of underfitting, appropriate fitting and overfitting. Altered Figure from: [21]

3.4.1 Overfitting

Overfitting takes place when some training process has generated a Machine Learning model which is not able to properly generalize; and therefore, cannot make accurate predictions with previously unseen examples. In other words, the model is **biased** and has basically memorized the training examples or learnt from the noises and inaccuracies in the samples of the dataset. This may be due to a reduced number of training examples provided to the model. On the other hand, and **underfitted model** is the opposite case, when the model excessively generalizes, so it will have a poor performance even on the training data.

To make it more clear, an example is shown in Figure 3.1. In this Figure, we can see three different approaches for one problem which tries to classify between two different classes: the ”x” points and the ”o” points. We can see in the first graphic an underfitted model as the linear model cannot properly classify an appropriate number of the data samples. In the third graphic, even though the model fits all the data by using a more complex function, it is overfitted since this curve is too complex and it has fitted the training data in a forced way. Thus, it will likely be overfitted.

Despite it can be remarkably arduous in some cases, the wisest option is to choose a model which is at some point between underfitting and overfitting. In our example, the graphic in the middle in Figure 3.1, which uses a quadratic function seems to be the best option. Between the strategies to reduce overfitting we can emphasize increasing the number of samples in the training process, reducing the complexity of the model and removing useless input features.

3.4.2 Test and Training Splitting Techniques

Once the model is trained, its performance can be evaluated by predicting output values of samples whose outputs are already known and comparing those predicted values with the actual values. However, due to the problem of overfitting, it is never correct to do this assessment with examples already used in training, so it is necessary to use a new dataset. This new dataset is commonly called as **test set**. Normally, only one dataset is accessible, so the training set and test set must be obtained from splitting it. To ensure a proper analysis, there are different strategies to do this

separation.

The most basic method is known as **hold-out**. It consists in taking one percentage of the samples (usually 80%) for the training set, and using the rest (20%) as the test set. Although this separation may seem really simple, its usage is truly frequent in large datasets, where training is a long process and computationally expensive. In addition, it is a good practice to randomize the order of the samples in the original dataset prior to the separation.[2]

In the cases where it is feasible and efficient to do several train processes for a model, another widely used technique is **K-fold cross-validation** (or simply cross-validation). Here, the original dataset is splitted in K subsets of equal length and the model is trained K times, using one subset as test set and the remaining $K-1$ as training sets. Hence, this allows to check if there are significant variations between the results acquired from the different splits, which is a good way of ensuring the robustness of the dataset and the absence of overfitting.

A particular case of K-fold cross validation is **leave one out**. In this technique $K = N$, where N is the number of samples of the dataset. This means that the whole dataset except one sample is used to train the model, making the training set the biggest possible reducing overfitting and using all this information to make only one prediction per each training. As the reader may have already noticed, this is the most expensive one in terms of computation.

3.4.3 Metrics for Classification Models

Once the model has been trained and tested, it is necessary to use some metrics extracted from the results in order to make it easier to infer conclusions. The testing part provides as outcome the probability of each sample belonging to one class. Then, an appropriate threshold is chosen to make the difference between the different classes. In the case of two classes, which is the most significant for this study, a threshold of 0.5 is commonly used.

3.4.3.1 Confusion Matrix and Metrics.

When the classification of each test sample is available, then the number of correct and incorrect predictions can be counted. Considering each individual sample and two possible classes, there are four possible results:

1. **True Positives (TP)**: These are the cases properly classified as positive. In our case, these are the Content Polluters accounts accurately identified.
2. **True Negatives (TN)**: These are the cases correctly classified as negative. In other words, Legitimate Users identified without any error.
3. **False Positives (FP)**: If a sample is a False positive, it means that it has been erroneously classified as positive, whereas it was an actual negative. In our study, this is a Legitimate User misclassified as a Content Polluter.
4. **False Negatives (FN)**: In this case, a sample has been classified as negative, whereas it was an actual positive. In our study this is a Content Polluter misclassified as a Legitimate User.

[2] This helps to avoid conditioning the model, since in the original dataset there may be some spurious correlations between the samples next to each other.

The number of each of this four cases above are usually placed in the so-called **Confusion Matrix**, which is defined as:

$$\begin{pmatrix} TrueNegatives & FalsePositives \\ FalseNegatives & TruePositives \end{pmatrix} \tag{3.2}$$

Once the predictions are done, some metrics can be calculated from the values in the Confusion Matrix in order to evaluate the proportions of each case. The most important are:

1. **Accuracy**: It measures the number of cases correctly classified. Thus, it can be considered one of the most intuitive metrics.

$$Accuracy = \frac{CorrectCases}{NumberOfPredictions} = \frac{TP + TN}{TP + TF + FP + FN} \tag{3.3}$$

2. **Precision**: Ratio of Positives correctly predicted to the total number of positive predictions.

$$Precision = \frac{CorrectPositives}{NumberOfPositivePredictions} = \frac{TP}{TP + FP} \tag{3.4}$$

3. **True Positive Rate (TPR)**: Number of positives predicted in the correct manner divided by the total number of actual positives. This metric is also known as **Sensitivity** and **Recall**.

$$TruePositiveRate = \frac{CorrectPositives}{NumberOfActualPositives} = \frac{TP}{TP + FN} \tag{3.5}$$

4. **False Positive Rate(FPR)**: Number of false positives divided by the number of actual false samples.

$$FalsePositiveRate = \frac{FalsePositives}{NumberOfActualNegatives} = \frac{FP}{TN + FP} \tag{3.6}$$

5. **Specificity**: It is the ratio of the samples correctly classified as negative to the number of actual negatives. It is also called **True Negative Rate**. Moreover, it is the opposite (complement) of False Positive Rate.

$$Specificity = \frac{TN}{TN + FP} = 1 - FalsePositiveRate \tag{3.7}$$

3.4.3.2 ROC Curve and AUC

In the introduction of this section it was stated that, given the probability values of each sample belonging to one class or other, a threshold of 0.5 is commonly used. However, a remarkably effective way of performance measuring of a classifier is to vary this threshold and studying the produced effects. Specifically, this analysis is executed through the use of the Receiver Operating Characteristics (ROC) Curve.

The ROC Curve is a function ordinarily plotted with FPR and TPR in the axes. If the threshold is changed, then a new Confusion Matrix is generated since the values of TP, TN, FP and FN vary.

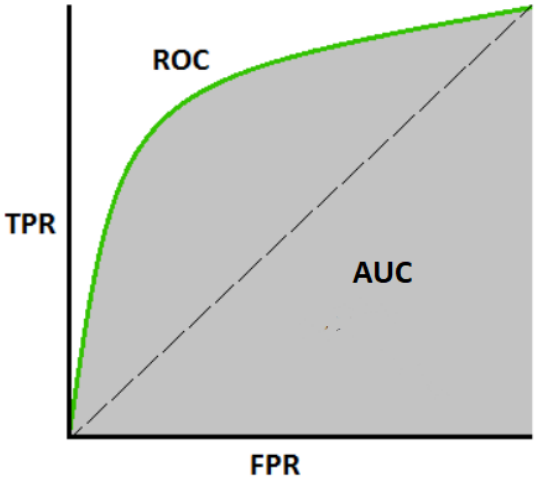

Figure 3.2: Example of a ROC Curve and AUC. Source: [22]

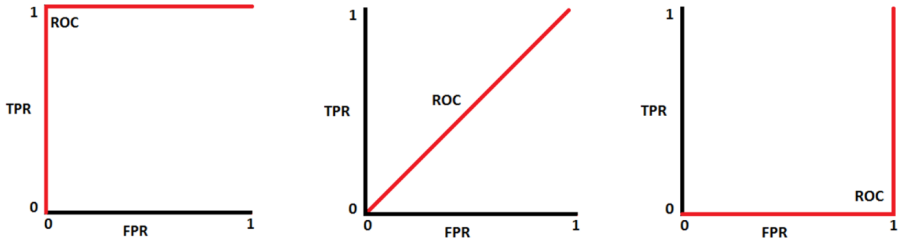

Figure 3.3: Ideal cases for the ROC Curve. From left to right, AUC = 1, AUC = 0.5 and AUC = 0. Source: [22]

Each point in the ROC Curve is related to one Confusion Matrix, which has been generated by a threshold. Thus, in order to get the ROC Curve, it is fundamental to do a scanning through different values of thresholds and record every pair of FPR-TPR values. An example of a ROC Curve can be appreciated in Figure 3.2

At first sight, the ROC Curve itself does not provide any accurate information regarding the performance of the classifier. What gives performance measurement in terms of separability is the Area Under the ROC Curve (AUC), which is highlighted in Figure 3.2. If we suppose an ideal case where the classifier is able to adequately classify all the test samples, then, the ROC Curve will look similar to the graph on the left side in Figure 3.3 and its AUC will be 1. This is quite reasonable, since it implies that the TPR reaches the maximum number while the FPR is at its minimum value.

On the other hand, if a classifier produces an AUC of 0.5, that entails that the model is classifying the samples randomly. On the contrary, if the AUC is 0, then, the model is confusing the two classes, classifying the positives as negatives and the negatives as positives, since the FPR gets to be maximum while the TPR is minimum. Both ideal graphs are shown in Figure 3.3.

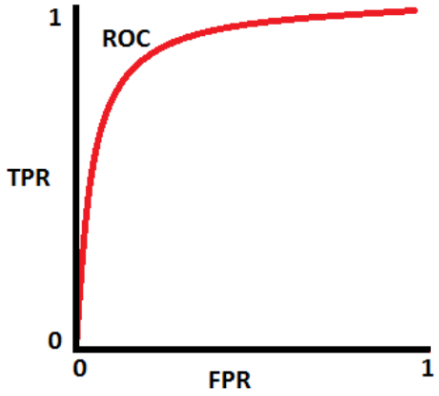

Figure 3.4: Real ROC case of a proper classifier. AUC close to 1. Source: [22]

In terms of a typical real ROC curve of a good classifier, its AUC will be close to 1, but still strictly minor than 1, due to noises and inaccuracies in the datasets. A curve of this form is represented in Figure 3.4.

Chapter 4

Machine Learning Classifiers

4.1 Introduction

Throughout Machine Learning history, a lot of different proposals have been made to face classification problems. Within this wide variety of models, it is possible to find mechanisms which are based on extremely different strategies, some of them more intuitive than others, to differentiate if one input belongs to one class or to another.

4.2 Two Types of Classifiers

Before coming into the details of the most relevant models, it is important to explain that Machine Learning classifiers can be differentiated into two types, according to their use of the training data.

1. **Lazy Learners**: These learners basically only store the training data without carrying out any training process or similar [23]. By the time a test sample needs to be predicted, these models check their stored training data to compute an output. Hence, one ”natural” property of these classifiers is the large amount of time needed for each prediction, since all the training dataset (or a significant part) must be evaluated to achieve that output. In contrast, the training process cost in time is nonexistent or really reduced. Some important examples are K-Nearest Neighbours or Case-based Reasoning.

2. **Eager Learners**: Unlike Lazy Learners, these kind of models always execute some process with the training data prior to making predictions. In other words, they are able to extract some general abstract information from the training data, which is stored as ”the state” of the model. This information will be subsequently applied to testing new data without using the training data one more time. The main advantage is that this abstracted information allows to predict in a much more efficient way compared to Lazy Learners. Oppositely, the drawback is that these models imply learning processes which may result expensive in terms of time and/or resources. Some of the most representative examples of this group are Decision Trees, Random Forests and Artificial Neural Networks.

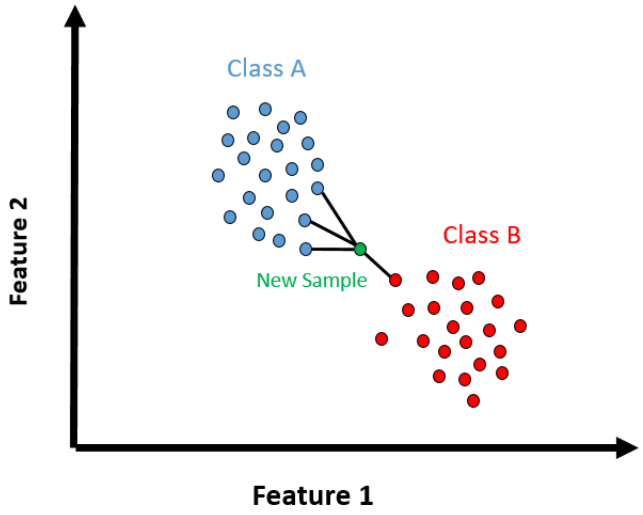

Figure 4.1: K-Nearest Neighbours example with 2 features.

4.3 Lazy Learners

4.3.1 K-Nearest Neighbours

K-Nearest Neighbours (also known as KNN, in its abbreviated form) is considered one of the simplest classifiers in the Machine Learning Field. As we already know, it is a Lazy Learner, therefore it stores the training data and uses it for new predictions. Particularly, this classifier interprets each instance of the training dataset as points placed in a n-dimensional space, considering each feature as one of these dimensions. Consequently, the features for this classifier must be numeric[1].

In terms of predicting the class of a new sample, K-Nearest Neighbours classifier, as its name indicates, searches in the n-dimensional space for the K points whose distance to the new input is minimum compared to the rest of points. After this, the algorithm checks the classes of those K neighbours and returns the probability of each class C as

$$P(C) = \frac{n_c}{K} \tag{4.1}$$

where n_c is the number of neighbours belonging to class C. In Figure 4.1 an example of KNN classification can be appreciated. In this example, we can find the training data formed by two classes represented in a two-dimensional space (since these dataset only has two features). One sample whose class is unknown is emphasized in green colour. This Figure shows the distances of the 4 nearest neighbours of this sample. Therefore, according to the explanation above, $P(A)$ will be $\frac{3}{4}$ and $P(B)$ will be $\frac{1}{4}$ suggesting that this sample is contained in class A.

One relevant point in terms of KNN that should be added to this description is the fact that different definitions of distances between two points are used. Naming the n features values for point a as $f_{a1}, f_{a2}, f_{a3}, \ldots, f_{a(n-1)}, f_{an}$ and $f_{b1}, f_{b2}, f_{b3}, \ldots, f_{b(n-1)}, f_{bn}$ for point b the definitions of some

[1] However, there are methods to convert categorical features into numeric ones, so they can be adapted to be used in a KNN classifier.

of them are:

1. **Euclidean Distance**: This is the most intuitive idea of distance. Basically, it consists in the length of the segment which joins the two points.

$$D = \sqrt{(f_{b1} - f_{a1})^2 + (f_{b2} - f_{a2})^2 + \cdots + (f_{b(n-1)} - f_{a(n-1)})^2 + (f_{bn} - f_{an})^2} \quad (4.2)$$

2. **Manhattan Distance**: It can be defined as the distance between two points along straight-lines which are parallel to the axes in Cartesian coordinates.

$$D = |f_{b1} - f_{a1}| + |f_{b2} - f_{a2}| + \cdots + |f_{b(n-1)} - f_{a(n-1)}| + |f_{bn} - f_{an}| \quad (4.3)$$

3. **Minkowski Distance**: It is another metric of distance which can be considered as a generalization of the two previous distances described above. If the parameter $p = 1$, then we find Manhattan distance. Otherwise, if $p = 2$, then we obtain Euclidean distance.

$$D = \left(|f_{b1} - f_{a1}|^p + |f_{b2} - f_{a2}|^p + \cdots + |f_{b(n-1)} - f_{a(n-1)}|^p + |f_{bn} - f_{an}|^p\right)^{\frac{1}{p}} \quad (4.4)$$

As the reader may have already noticed, all the features are treated equally for computing distances. Therefore, if some features are values whose scale is different respect to the rest, that will cause a problem when making predictions. The best solution for this issue is to normalize or standardize all the features [2]. Normalization consists in subtracting the minimum value all the elements of a feature and dividing them by the difference of the maximum and minimum value. In this way, all the instances are within the $[0, 1]$ interval. On the other hand, standardization is based on changing the scale so that the final result is similar to a Gaussian distribution (mean 0 and standard deviation 1). This technique is preferred, since Normalization may cause **outliers**, which are features which present significant deviation from the rest of the features.

4.4 Eager Learners

4.4.1 Decision Trees

Decision Trees operate in an intuitive manner, at least compared to other classifiers. They are based on a tree architecture which is created during the training process. Each node in the tree corresponds to one ”question” with a discrete number of possible answers. Each answer is a branch that leads to other nodes. When a leave node is reached, then the output is provided. An example of a Decision Tree is shown in the Figure 4.2. In this example, the decision to be made is to do some leisure activity or to stay at home. As we can see, leave nodes are possible options for the decision and the intermediate nodes are the appropriate questions such as ”Work to do?”, or ”Weather outlook?”.

In terms of the training process, a Decision Tree is built by automatically analyzing the training dataset. Normally, algorithms which make their best to find the most relevant features in the dataset

[2] In spite of the fact that this is not necessary for all the classification models, it is a recommended practice to do this rescaling process in all the datasets.

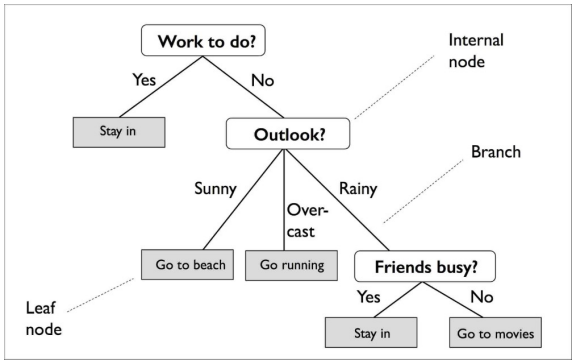

Figure 4.2: An example of a Decision Tree. Source: [24]

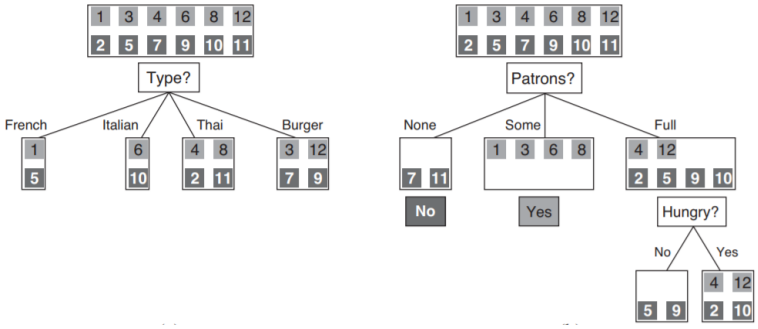

Figure 4.3: An example of the training process of a Decision Tree. The training dataset is based on restaurants. Source: [20]

are used. The adjective ”relevant” in this paragraph refers to the separability that can be inferred by examining this attribute [20]. In other words, the most relevant feature is the one that facilitates the most to classify a sample. Once the most important attribute has been found, the question corresponding to the root node is generated in relation to this feature. Afterwards, the algorithm splits the dataset according to the samples that satisfy each answer.

In this way, the proportions of samples belonging to each class are recorded and related to each branch (answer) of the root node. The next step consists in evaluating the next attribute in terms of relevance in each branch, to repeat the entire process and create a new query node on each branch. The learning algorithm finishes either when all the leave nodes only contain samples of one class or the parameter of maximum depth is reached.

An example of this process is reflected in Figure 4.3. This illustration uses a specific dataset which contains several cases to decide if it is worth for a customer to wait in a restaurant for a table or not. The boxes with the numbers display the positive and negative samples in the training dataset, with light and dark colours respectively. Moreover, the Figure proposes two possible Decision Trees to solve this problem. Nonetheless, the one placed on the left side is completely useless since there are the same samples of the two classes on each leaf. The output would always be 0.5. In contrast, the Decision Tree on the left side does a good job in terms of splitting. This demonstrates that the feature ”Patrons” is more relevant than ”Type”, which does not give any information.

In regard to the testing part, the tree is traversed by applying the different question nodes to the input sample. If a leave node which only contains training samples of one class is reached, the

output of the model is clear. Otherwise, if it stores samples of multiple classes, then the outcome is usually the probabilities of belonging to each class, obtained as the fractions of samples of each group.

This classifier can deal with both categorical and numerical features. In the case of real numeric values, the discrete answers for a question will be obtained as a result of splitting this feature in continuous intervals. Furthermore, this tree architecture allows to compute predictions in a fully efficient manner.

4.4.2 Random Forests

Random Forests are an extension of Decision Tree classifiers. As its name suggests, it is composed of several individual Decision Trees working in a coordinated way. Hence, this methodology provides all the benefits from teamwork philosophy, applied to Decision Trees. Although this may seem to be a simple improvement, not every possible set of trees provides outstanding results. The key to get a proper predictive Random Forest model relies on the fact that trees are uncorrelated [25]. If the trees have this property, then individual errors in a tree do not affect the rest of the forest in a prediction.

In order to properly generate uncorrelated trees, the main technique available is known as **Bagging**. This technique is based on the principle of making a different training dataset called **Bootstrapped Dataset**, which is used to train the Random Forest. To create this alternative dataset, a subset of samples is taken from the original dataset. However, the aim is to ensure that the bootstrapped dataset shares the same length with the original dataset. Therefore, the resulting dataset contains some repeated samples from the original dataset.

In addition, during the training process, the method of **random sampling with replacement** is also employed. It consists in extracting random features from the dataset, with replacement (therefore, it is possible to choose a feature previously selected). Concretely, for the creation of each different individual Decision Tree, only a few features of the Bootstrapped Dataset are used. This is illustrated in Figure 4.4. By using this method, different trees and uncorrelated trees will result since the searches for the most relevant feature will provide different outcomes for each tree.

With regard to the testing part, the output of the model for an input is the proportion of individual predictions for each class provided by all the individual Decision Trees. In a practical sense, this model has a wider range of possibilities, since different approaches can be tried varying the maximum depth parameter and the number of estimators.

One of the Random Forest advantages is the stability of the model. In individual Decision Trees, a few new examples in the training dataset may alter the different splits in the tree, generating a different model. However, Random Forests are more strong against changes in the dataset, due to the variability in the trees.

4.4.3 Artificial Neural Networks

Artificial Neural Networks are nowadays one of the most widely used Machine Learning models. In spite of the fact that this concept was created by the end of the 1950s with the birth of the **Perceptron**, their powerful capacity has not been exploited until the recent years due to hardware limitations. There are currently hundreds of different researches in this field, concretely known as

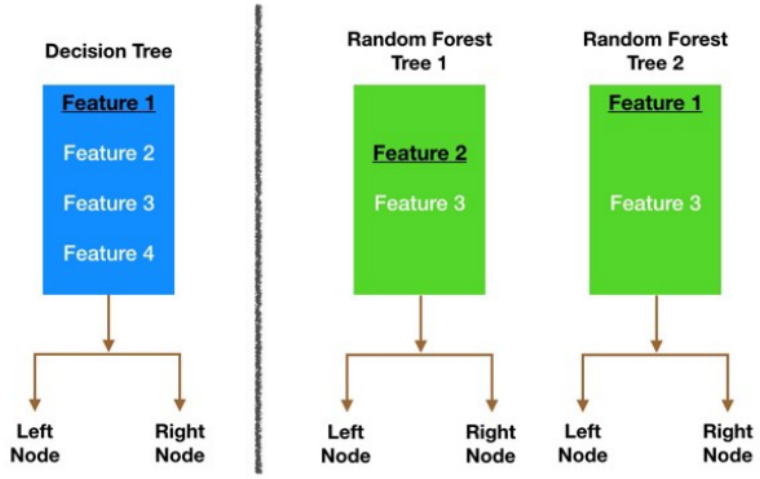

Figure 4.4: A comparison between individual Decision Tree and Random Forest with Feature Randomness. Source: [25]

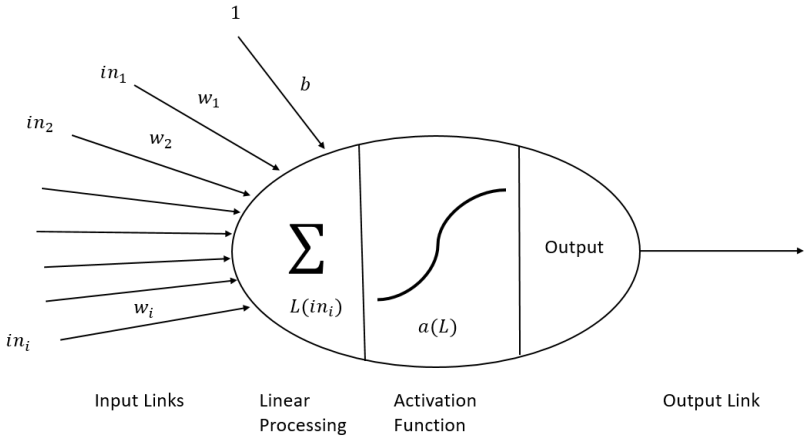

Figure 4.5: Complete Mathematical model of a Simple Neuron. Altered Figure from: [20]

Deep Learning.

4.4.3.1 Individual Neuron (Perceptron)

The mathematical model of an Artificial Neuron is quite simple: denoting the inputs as in_i for i input links, the Perceptron abstracts information from the training dataset by processing them linearly, as shown in equation 4.5.

$$L(in_i) = \sum w_i \cdot in_i = w_0 \cdot in_0 + w_1 \cdot in_1 + \cdots + w_i \cdot in_i + b \tag{4.5}$$

where w_i denotes the **weights** for each input link of the model. Moreover, there is one more added term b, usually called **bias**. The weights and the bias are the characteristic parameters of each neuron.

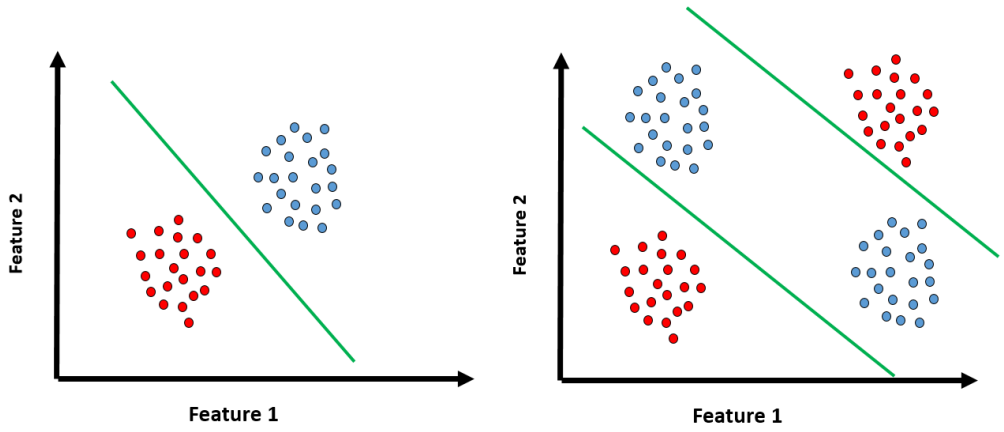

Figure 4.6: Two linearly separable problems. The example on the left-side can be solved by using a single neuron, whereas the other one needs two neurons.

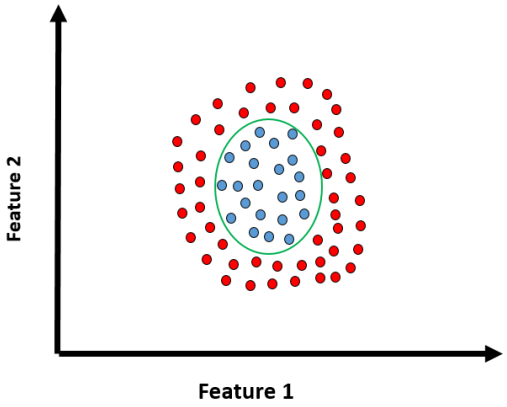

Figure 4.7: Not linearly separable problem. Activation Function is required.

During the learning process of a neuron, this will vary its weights and bias until it is able to distinguish between the different classes of the training dataset. As it is noticeable, this method operates likewise a Linear Regression model. Thus, this mathematical model works effectively in classification models where the classes are linearly separable. One example of linearly separable problem can be appreciated in the left graphic in Figure 4.6.

The matter gets a bit more complicated when it is not possible to do this separation by using a straight line. One example is the second graph in Figure 4.6, where there is no possible single straight line able to split the two groups of samples. The solution proposed in the Figure is to use two lines (and therefore, two neurons in parallel) instead.

For those cases where the classes cannot be trivially split by using several lines, such as the distribution in Figure 4.7, it is necessary to look for a different approach. Concatenating different neurons will not provide a better solution, given that applying several linear transformations sequentially is equivalent to do a single linear function. Hence, one new element is required in the neuron: the **Activation Function**.

An Activation Function $a(L)$, is a non-linear function whose aim is to allow the neuron to solve non-linear problems. Intuitively, the output of the Linear Processing $L(in_i)$ is the input of the Activation Function. The complete model of a simple neuron is shown in Figure 4.5. Some of the most common Activation Functions are:

1. **Threshold Function**: It is also called **Step Function**. This is the simplest case of Activation Function, due to the fact that it returns one value if the input is greater than a threshold (usually 0) or a different value instead. Equation 4.6 and Figure 4.8 represent the case when the threshold is 0.

$$T(z) = \begin{cases} 0 \quad if \quad z \leq 0 \\ \\ 1 \quad if \quad z \geq 0 \end{cases} \tag{4.6}$$

2. **Relu Function**: This function does not alter the input only if it is positive. Otherwise, it is truncated to zero. It can be observed in Equation 4.7 and in the Figure 4.8.

$$R(z) = max(0, z) \tag{4.7}$$

3. **Sigmoid Function**: It is also known as **Logistic Function**. As appreciated in Figure 4.9 , it is S-shaped and within the $[0, 1]$ interval. When the input contains large values, the output gets closer to 1, whereas if the input is reduced, the output tends to 0. This is ideal to obtain probability values as the output of a neuron.

$$S(z) = \frac{1}{1 + e^{-z}} \tag{4.8}$$

4. **Hyperbolic Tangent**: Similar to Sigmoid Function but its values are within the range $[-1, 1]$. Its representation is in Figure 4.9.

$$Tanh(z) = \frac{sinh(z)}{cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{4.9}$$

During the training process, the Neuron will be initialised with random parameters (weights and bias) and will start making predictions and comparing them to the known target values in the training dataset. Then, its weights and bias will be gradually fitted until an optimal model is reached. In order to properly achieve that goal, the performance of the model must be somehow measured. Concretely, with the use of an error function. Some of these **Loss Functions** are:

1. **Mean Absolute Error(MAE)**: It is the absolute value of the difference between the predicted target value y_i^p and the actual value y_i.

$$MAE = \frac{\sum |y_i - y_i^p|}{n} \tag{4.10}$$

2. **Mean Squared Error Loss(MSE)**: It is the mean of the squared difference between the predicted target value and the actual value. It is usually used for regression problems. The

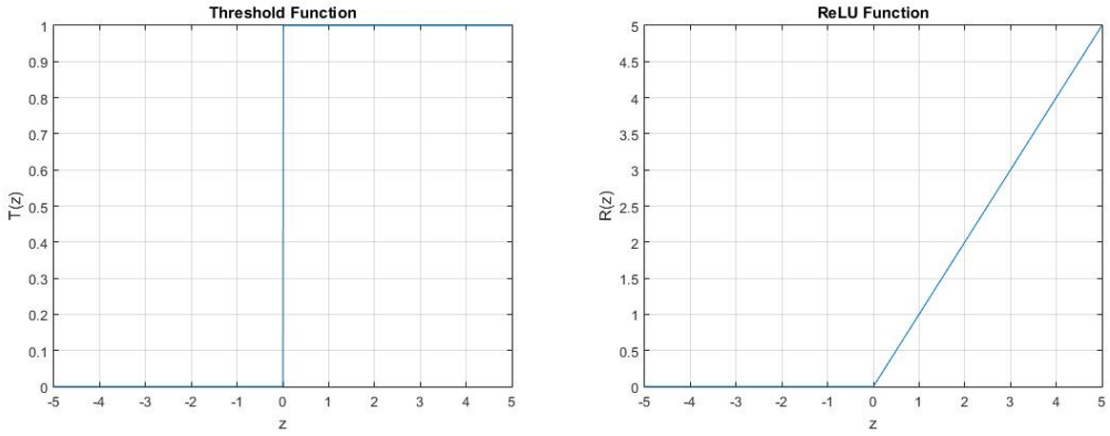

Figure 4.8: Plot of Threshold and Relu Functions.

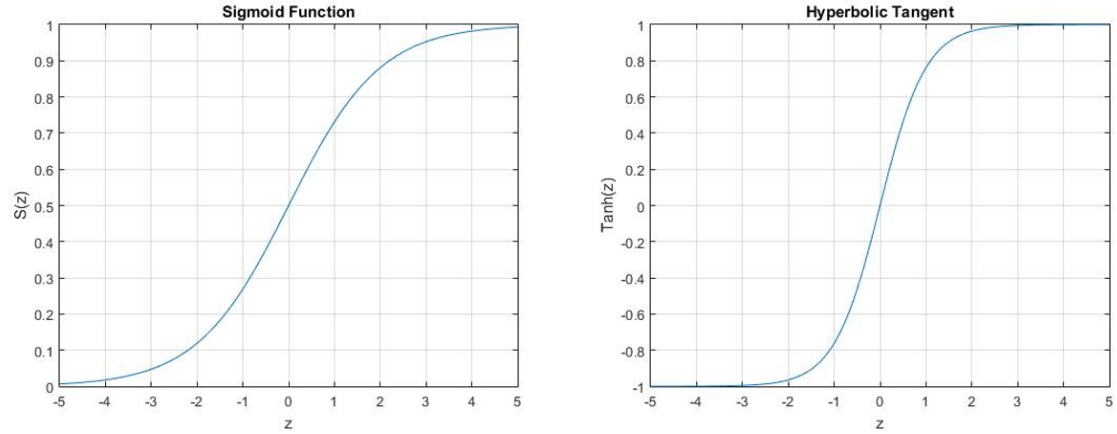

Figure 4.9: Plot of Sigmoid and Hyperbolic Functions.

advantage regarding MAE is that large errors result in greater losses compared to MAE due to the quadratic effect.

$$MSE = \frac{\sum (y_i - y_i^p)^2}{n} \tag{4.11}$$

3. **Log-Cosh Loss**: This loss function computes the logarithm of the hyperbolic cosine of the prediction error. This function can be approximated as $log(cosh(x)) \approx \frac{x^2}{2}$ for small x and $log(cosh(x)) \approx |x| - log(2)$ for big x. This implies that this function works mostly like MSE, without being so strongly affected by high error predictions [26].

$$L = \sum log(cosh(y_i^p - y_i)) \tag{4.12}$$

The objective is to minimize the Loss function used in each case. To do this minimisation, the **Gradient Descent Algorithm** is employed. This mechanism computes the partial derivatives of the Loss Function with respect to the parameters of the neuron in order to obtain the gradient. The gradient represents the ”direction” of maximum positive variation in an n-dimensional space. Hence, once the gradient has been obtained, it is possible to know how the neuron parameters must be changed to get closer to the minimum of the Loss Function.

4.4.3.2 Multi Layer Perceptron (Dense Neural Network)

After analyzing the concept of an individual Artificial Neuron, we can now discuss some of the configurations of Neural Networks which combine the power of multiple single neurons[3]. Artificial Neural Networks are formed by several layers serially connected. Moreover, each layer can be different configurations of neurons. There are 3 types of layers:

1. **Input Layers**: The set of neurons that receive the input data during both training and testing processes.
2. **Hidden Layers**: Intermediate layers which are invisible for the user of the network.
3. **Output Layers**: This layer will provide the output of the model.

The simplest Neural Network architecture is publicly known as **Dense Neural Network** because it is composed by **Dense Layers**. This is a formal way of saying that its layers contain several Neurons working in parallel, whose outputs are fully connected to the inputs of the adjacent layer. Furthermore, the output layer is a single neuron. This architecture is illustrated in Figure 4.10.

This architecture could be trained similarly to one individual neuron by the use of the Gradient Descent Algorithm. However, calculating the gradient in its pure form is quite exhaustive in this type of Neural Network, given that the variation of one weight of the first layers affects all the neurons in the subsequent layers. The solution for this is the **Back-propagation Algorithm**.

The intuition of how the Back-propagation Algorithm proceeds is to evaluate the Neural Network backwards when an error comes out. This means to first assess the last layer and analyze the

[3]Only those networks which are relevant for this study will be explained in detail. Nevertheless, there are other widely-used architectures in other contexts such as Convolutional Networks.

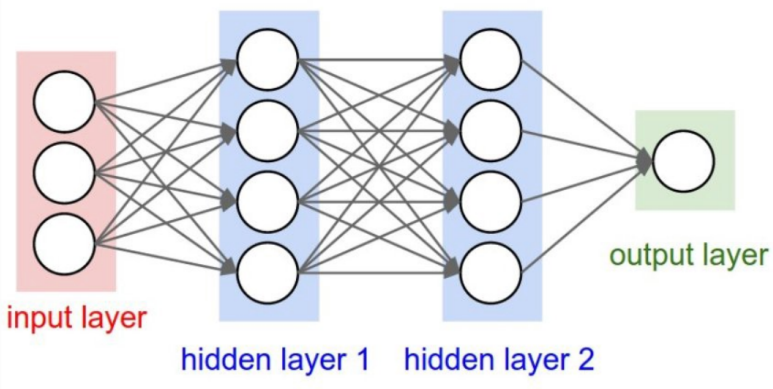

Figure 4.10: Dense Neural Network (Multi-Layer Perceptron). Source: [27]

responsibility of each parameter of the layer in the final result to deduce which parameters to alter. If the result has not properly improved, the error is "back-propagated" to the previous hidden layer and the process is repeated. To satisfactorily do these analyses, several recursive derivatives are computed[4].

4.4.3.3 Recurrent Neural Networks, LSTM and Embedding Layer

In the field of **Natural Language Processing (NLP)** there have been several advances in the recent years. It is necessary to have them into account in this study since analyzing datasets which contain tweets involve the employment of some text processing technique.

Recurrent Neural Networks (RNN) are a special architecture for Neural Networks whose aim is to deal with problems that involve sequential inputs, such as language, where the order of the words contains high significance. These networks process one input element of a sequence at a time, storing in their hidden units a **State Vector** which has some abstract information about the history of the previous elements of the network [28]. That is the main difference with respect to regular Dense Neural Networks, which can process a similar input sequence without learning any pattern from the neighbours of each element.

In other words, that State Vector stores the context learnt from previous inputs causing that two outputs may be different for the same input if the context in the hidden units differs in the two cases. Hence, as proposed in [28], these models are suitable for predicting the next character in a word or the next word in a sentence.

The general design of this class of Neural Networks can be observed on the left-side in Figure 4.11. In this Figure, the inputs are denoted as x and the outputs as o. The node s makes reference to the set of hidden neurons that form the internal State Vector, which receives information from previous states, pointed out in the Figure as the small black box next to the s node. Moreover the matrices U, V and W represent the weights of the neurons.

Once explained the elements in the left side of the Figure, we can focus on the unfolding part. In

[4]This is only a really brief and intuitive way of how the Back-propagation Algorithm operates. However, it is much more complex in terms of precise mathematical details, which are not explained here because it is not the intention of the study. If the reader is interested in acquiring a deeper knowledge, the complete theoretical approach can be checked in [20].

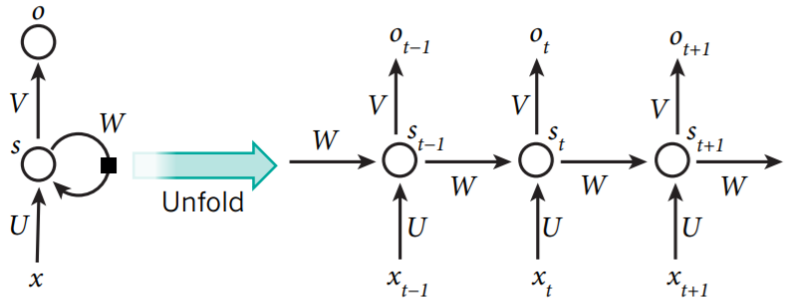

Figure 4.11: Basic design of a RNN and the unfolding in time of its process of predicting. Source: [28]

this particular illustration, the black box constitutes a delay of one time step; so, as it is observed, the previous state s_{t-1} influences the current state s_t, implying that somehow the output o_t depends on the previous inputs (not only on x_{t-1}). Furthermore, in terms of learning process, the Back-propagation Algorithm can be applied to the unfolding scheme, training the network similarly to a Dense Neural Network.

In practice, storing the abstract context for so long is not possible with this architecture. Thus, one new component should be introduced: **Long Short-Term Memory (LSTM) network**. This type of network is quite similar to basic RNN, but which makes use of special hidden units prepared to store the extracted dependencies for a long time [28]. Moreover, these special hidden units contain sub-neural-networks which control which knowledge must be cleared from the memory and which should stay. More technical details about it can be found in [29].

The inputs of the LSTM Neural Networks are usually previously processed by an **Embedding Layer**. This layer represents its input tokens (which can be words, characters or sub-words) as dense vectors. These vectors are projections of the words in a continuous space. This characteristic provides advantages such as the possibility of representing two words whose meanings are similar but not equal as vectors whose distance is small, and otherwise, as vectors with a greater distance instead. These representations as vectors are learned during the training process.

4.4.4 Naive Bayes

Unlike the rest of the models already explained in this chapter, this classifier is based on **Probability Theory**. Denoting the output as Y and the vector of K inputs (features) as $X = (X_1, X_2, \dots X_K)$ this model considers X and Y as random variables with values x and y respectively. In terms of probability equations, the target is detailed in Equation 4.13. In other words, the objective is to find a value y that maximizes the probability of y being the actual value of the output, given that all the input features.

$$Prediction = argmax_y P(Y = y | X = (x_1, x_2, \dots x_K)) \tag{4.13}$$

In addition, this model utilizes the **Bayes Theorem** (Equation 4.14) in order to solve the target probability equation. This is due to the facility to calculate the probability with inverse causality $P(X|Y)$ instead of directly computing the target $P(Y|X)$. Later, $P(Y|X)$ can be obtained by

using the Bayes Theorem.

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)} \tag{4.14}$$

Therefore, the prior probabilities $P(X)$ and $P(Y)$ together with the conditional probability $P(X|Y)$ must be directly computed from the training dataset. The necessary details for doing this computation are reflected in Equations: 4.15 and 4.16. The notation used in those equations is: n_x and n_y for the number of samples in the dataset that correspond to a fixed value of X and Y respectively; m refers to the total number of samples in the dataset.

$$P(X = x) = \frac{n_x}{m} \qquad P(Y = y) = \frac{n_y}{m} \tag{4.15}$$

$$P(X|Y) = \frac{P(X \wedge Y)}{P(Y)} \tag{4.16}$$

Nevertheless, there is a practical issue, if the quantity of samples in the dataset is numerous, then it may be infeasible to do these computations. However, this obstacle can be overcome by accepting a usually-called **naive assumption**[30] . If we suppose that all the features $(X_1, X_2, \ldots X_K)$ are conditionally independent with each other, given Y = y, we obtain the result in Equation 4.17 [5]. It is said to be ”naive” in the bibliography because this premise is not true in most of the cases, since independence is a rare characteristic between the features in a dataset.

$$P(X_1, X_2, \ldots, X_K|Y) = P(X_1|Y) \cdot P(X_2|Y) \ldots P(X_K|Y) \tag{4.17}$$

Despite its contemptuous name, this assumption significantly reduces the computational cost, transforming it from exponential to linear in terms of the size of the dataset. An advantage caused by that is the efficiency of this model when facing high dimensional datasets. Moreover, this model has empirically demonstrated that it can work competently.

[5]The term of conditionally independence is different from the concept of pure independence. Two variables are conditionally independent if, given some value for a third variable, the probability of one of the variables is not affected by the other, even though this is not true without the given information by the third variable.

Chapter 5

Methodology

Once the context of the problem caused by Content Polluters has been introduced, and the theoretical background in terms of Machine Learning has been properly analyzed, it is time to get started with the concrete methodology used in the project.

5.1 Utilized Datasets

One of the crucial parts of the study is to choose which datasets will take part. It is strictly necessary to use proper datasets in order to train the models in an adequate manner and therefore, obtain trustworthy results. Obviously, if a dataset, which does not reflect the reality, is employed, then the Machine Learning models will not be able to classify decently. Given that collecting data may be complicated and may take enormous amounts of time, it is convenient to do some research about different public datasets.

5.1.1 Caverlee

One of the most important researches in the state of the art analysis in Chapter 2 was the deep research carried out by **Caverlee et. al.** in [6]. This team was able to obtain a really complete dataset with an abundant number of samples with both Content Polluters and Legitimate Users.

The methodology used was based on tempting the Content Polluters by the use of honeypot accounts. The researchers created 60 social automated Twitter accounts whose activity consisted on posting random tweets previously acquired from the Twitter public timeline. As they were tweets with no relation between each other, theoretically no Legitimate User would follow those honeypots.

Moreover, those Content Polluter hunters were carefully designed to avoid interfering with Legitimate Users as it is detailed in [6]. Concretely, these profiles only followed each other and only interacted with each other (posting tweets with mentions to other honeypots). This choice would reduce the chance of getting a Legitimate User as a follower. To reinforce the confidence in the hypothesis of collecting Content Polluters, a clustering study was applied to the collected users. The clusters obtained were subsequently analyzed by the researcher, concluding that the clustering model had identified several type of spammers and malicious promoters.

After 7 months, the honeypots had gained 36,043 Content Polluters as followers. After removing the fake users who followed more than one honeypot account and those who were rapidly detected and removed by Twitter, the final dataset resulted in 22,223 Content Polluters. This last action was done because the objective in the study was to create Machine Learning models which were able to improve Twitter techniques for detecting and eradicating this type of bots.

On the other hand, to find Legitimate Users, Caverlee et. al. randomly harvested 19,297 accounts from Twitter. To ensure that those accounts were more likely to be legitimate rather than malicious, they were monitored during three months. Once this period ended, the 19,276 accounts which were still active (and not eliminated by Twitter) were labeled as Legitimate Users. In addition, the final reason given to support this hypothesis was the fact that a big error in Legitimate Users sampling would be reflected in the metrics used in the study, and they resulted to be outstanding.

The main advantages of this dataset is the great number of samples that includes and its completeness in terms of the features. Given that this project was done before Twitter limited the publication of complete datasets, this one contains the entire collected data. This is the main reason why this dataset is the basis of our project and will set the features that we use for our experiments.

With regard to the original dataset features that this dataset incorporate, they are shown in Figure 5.1. Their descriptions are:

1. **UserID**: This is not strictly a feature since its only purpose is to identify the users in the dataset.
2. **CreatedAt**: Date and time when the account in question was created.
3. **CollectedAt**: Date and time when the creators of this dataset founded the user.
4. **NumberOfFollowers**: This is the quantity of other profiles which follow the account in the sample.
5. **NumberOfFollowings**: It is the number of accounts that the user voluntarily follows.
6. **NumberOfTweets**: Total number of tweets which the user has published, without taking into account those previously removed by the user.
7. **LengthOfScreenName**: This is the length of the users' name. This name is unique for each user in Twitter and it is chosen by the owner of the account.
8. **LengthOfDescription**: This is the size of the description of the header of the profile. This is also written by the holder of the profile.
9. **ContentPolluter**: This is the target value of our project. If the user is a Content Polluter, this value will be "1" and otherwise it will be "0".

Fortunately, this dataset not only contains features related to users but also to specific tweets. Specifically, all the past tweets from each account were gathered. Furthermore, during the seven months of research, their system checked for new tweets. Similarly, the tweets had their own features, represented in the Figure 5.2. Their definitions are:

1. **UserID**: Identifier of the user who created the tweet. It is the same parameter as in the users dataset.

	UserID	CreatedAt	CollectedAt	NumberOfFollowings	NumberOfFollowers	NumberOfTweets	LengthOfScreenName	LengthOfDescription	ContentPolluter
0	6301	2006-09-18 01:07:50	2010-01-17 20:38:25	3269	3071	861	8	132	1
1	10836	2006-10-27 14:38:04	2010-06-18 03:35:34	1949	793	226	9	134	1
2	10997	2006-10-29 09:50:38	2010-04-24 01:12:40	1119	9644	38674	12	158	1
3	633293	2007-01-14 12:40:10	2010-01-24 11:59:38	2174	6029	12718	11	121	1
4	717883	2007-01-27 22:14:18	2010-02-06 06:25:58	7731	7029	873	6	70	1

Figure 5.1: Header of the Caverlee dataset. The features that this Raw dataset contains are illustrated.

	UserID	TweetID	Tweet	CreatedAt	ContentPolluter
0	6301	5599519501	MELBOURNE ENQUIRY: Seeking a variety of acts f...	2009-11-10 15:14:31\r\n	1
1	6301	5600313663	THE BURLESQUE BOOTCAMP SYDNEY - Open Date tick...	2009-11-10 15:46:05\r\n	1
2	6301	5600328557	THE BURLESQUE BOOTCAMP SYDNEY - Open Date tick...	2009-11-10 15:46:40\r\n	1
3	6301	5600338093	THE BURLESQUE BOOTCAMP SYDNEY - Open Date tick...	2009-11-10 15:47:03\r\n	1
4	6301	5600564863	Come to "The Burlesque Bootcamp - Sydney" Satu...	2009-11-10 15:56:03\r\n	1

Figure 5.2: Header of the tweets in the Caverlee dataset. Their features are shown.

2. **TweetID**: Unique identification of the tweet. It is therefore not used as an input feature to our models.
3. **tweet**: Text content of the tweet in question. Thus, it is the most relevant feature.
4. **CreatedAt**: Date and time when the tweet was published.
5. **ContentPolluter**: Target value of the tweet. Moreover, it uses the same values as the case of the users dataset.

For the tweets analysis, it has been supposed for our whole project that all the tweets posted by a Content Polluter are characteristic of a malicious account. This may not be entirely true, since a spammer could alternate between legal and illegal tweets to hide in a better way from Twitter spam hunters. However, this choice is quite reasonable due to our lack of deeper knowledge. Moreover, in the case of a spammer as Content Polluter, an important fraction of its tweets must be polluting in order to accomplish its objective.

The concrete statistics of this dataset, accompanied by the data of the other datasets are represented in the Table 5.1. As it can be observed in that table, this dataset is practically balanced regarding the number of samples in each class. In other words, the percentages of each classes are nearly 50%. This is an important consideration to ensure that our models are not misclassifying one class more than the other.

5.1.2 Gilani

Another public dataset acquired is the one generated by **Gilani et. al.** in [31] and it is in the Bot Repository of Indiana University [32]. This Bot Repository contains all the datasets used to train

an online tool, created by this university, whose purpose is to examine a Twitter account introduced by the client and return the probability of being a Content Polluter. The advantage is that all the datasets included in this repository are completely public.

This dataset has been created by manually labeling accounts collected from the streaming Twitter API as bot or human. This concrete streaming Twitter API allows the developer to contemplate tweets that are being published in real time. Therefore, all the samples in this dataset belong to accounts which were constantly active in 2017 which is a recent year as opposed to **Caverlee** dataset.

One of the advantages of this dataset is its approximately balanced number of samples for each class as shown in Table 5.1.

5.1.3 Vendor-Verified Mixed

This dataset is formed by two public independent datasets. Both of them are part of the Bot Repository of Indiana University [32] and logically mentioned in [2]. In this paper, Yang et. al. properly analyse the online tool created by this university and mention all the datasets used in it.

The first dataset is called **Vendor Purchased 2019**. This dataset consists of several bot accounts with no other intention than simulate that legitimately follow some other account. In other words, they are fake followers. Specifically, these bot accounts have been created by several companies with the objective to sell them to entities which want to inflate their number of followers. In contrast to **Caverlee**, this dataset was created approximately at the beginning of 2019, whereas **Caverlee** was published in 2011. It is important to consider current datasets given that bots are constantly in evolution.

The main issue of this dataset is that it only contains examples of Content Polluter profiles. Thus, it is imperative to mix it with one dataset of Legitimate Users. Our choice has been the dataset **Verified 2019**, which only holds Legitimate Users which are accounts whose identity has been verified by Twitter. Therefore, we can be sure that these profiles belong to humans and are Legitimate according to our definition.

Considering that the attribute ”verified” is not a feature in our dataset, our models will not be explicitly affected by the fact that these accounts are verified. Although there may be a bit of bias with respect to the classification in this dataset, it is still a good option. There are a wide range of different profiles which are verified, since the only condition is that the intention of the account in question must be of public interest [33]. Moreover, this dataset has been chosen because of its year of publication, so that the two datasets of this mixture make reference to the same epoch.

Finally, it is necessary to take into account the reduced number of public datasets of Legitimate Users obtained without being manually labeled by humans. Hence, by using this dataset we are trying a different approach and we will be able to contrast the results with **Gilani** dataset.

As it can be seen in Table 5.1, this dataset is quite unbalanced in terms of number of Content Polluters versus number of Legitimate Users. To avoid that our dataset introduces bias in the model, this dataset has been balanced so that exactly the same number of samples of each class takes place.

5.1.4 Split between Train and Test

As we already know, Machine Learning models need to be trained before being able to make predictions and we must never use the same samples in the training part and testing part. Thus, we need to split all the dataset into train datasets and test datasets.

In this project, each dataset has been split in the same way. First, to ensure the variability in both datasets, the samples of the original dataset has been shuffled. After this, it has been separated using the first 80% of the samples for training and the remaining 20% for testing, which is known as **Hold-out** technique. With these proportions, the models will learn properly and there will be enough samples for testing.

In addition to this, for the experiments carried out at the **user level**, other evaluations have been done by the use of **cross validation** with 5 groups. In other words, the dataset have been split in 5 groups and 5 tests have been completed by using 4 groups for training and the remaining one for testing.

5.1.5 Discarded Datasets

There are also other datasets publicly available which have not been considered for this project due to different reasons. Namely, the datasets: **Botwiki**, **cresci rtbust 2019**, **political bots 2019**, **botometer feedback 2019** are other datasets from the Bot Repository [32] which have been discarded due to their reduced number of samples (some of them within the order of a few tens, and the rest of a few hundreds). If these datasets had been used, there would not have been enough information to train the models, likely causing underfitting.

Besides, there was an alternative public dataset with only Legitimate Users to be mixed with **Vendor Purchased 2019**: **Celebrity 2019**, which was formed by almost 6,000 accounts belonging to celebrities. In spite of the fact that this dataset may seem to be better due to the number of samples, it has not been chosen because the aspect that all of these accounts are celebrities introduces more bias in the model than verified accounts in **Verified 2019**. This is because the feature ”verified” implies a wider range of different accounts than the case ”verified and celebrity”. Moreover, it would have been mixed with **Vendor Purchased 2019**, resulting in a really unbalanced dataset.

Another dataset was found from Kaggle, the popular website which lead competitions and share datasets related to Machine Learning and Data Science [34]. Concretely, the dataset name is **Popular Twitter bots Data** [35]. Nevertheless, this dataset contains mostly creative non-spammy bots, and according to our definition of Content Polluter in Chapter 2, this dataset is not suitable for this study. Furthermore, the number of samples is quite reduced.

5.2 Libraries and Tools

5.2.1 Python

The core tool of the project has been the Python programming language due to its broad possibilities in terms of Machine Learning tools. Moreover, one of its advantages is its legibility and ease for creating code. This makes it an efficient solution for this kind of projects.

	Content Polluter	Legitimate Users	Total Samples
Caverlee 2011 (Train)	53.549%	46.45%	33,199
Caverlee 2011 (Test)	53.56%	46.44%	8,299
Vendor Purchased	100%	0%	1,087
Verified 2019	0%	100%	2,000
Vendor Verified Mixed (Train)	50%	50%	1,180
Vendor Verified Mixed (Test)	50%	50%	294
Gilani 2017 (Train)	42.06%	57.94%	1,990
Gilani 2017 (Test)	42.34%	57.66%	496

Table 5.1: Statistics of the different user datasets.

	Content Polluter	Legitimate Users	Total Samples
Caverlee 2011 (Train)	42.07%	57.93%	4,493,538
Caverlee 2011 (Test)	41.35%	58.65%	1,119,621
Vendor Purchased	100%	0%	18,515
Verified 2019	0%	100%	24,059
Vendor Verified Mixed (Train)	46.47%	53.53%	21,819
Vendor Verified Mixed (Test)	46.34%	53.65%	5,442
Gilani 2017 (Train)	43.38%	56.62%	34,048
Gilani 2017 (Test)	43.94%	56.06%	8,526

Table 5.2: Statistics of the tweet datasets.

5.2.2 Tweepy

The datasets previously described: **Vendor-Verified mixed** and **Gilani 2017**, due to the new Twitter restrictions, only include user IDs and the labels for each ID as bot or human. Hence, those datasets must be completed by using Twitter API in order to obtain the adequate features for each sample. Specifically, an open source version of Twitter API for Python called **Tweepy** was used. This set of code functions allows to interact with Twitter data in a more simple and efficient way than using the usual API.

The features downloaded for each sample are the same contained in **Caverlee** dataset. In this way, we will be able to compare between the different experiments with the different datasets, observing if using the same features for each dataset provide the same results. In regard to the tweets, from each user in those datasets, 20 tweets were collected. The final number of tweets collected from each user is reflected in Table 5.2.

The free version of Twitter API has been used for this project. Therefore, this limits the possibilities regarding the datasets which did not previously contain the features of each users, especially for the tweet datasets. However, these datasets are still useful to get a first approach of the performance of the classifiers on them, obtaining more results than only using **Caverlee** dataset which already included the features.

5.2.3 Pandas

This is a Python library which allows the developer to perform several operations with datasets such as, splitting them by columns or rows, shuffling their samples, filter samples which fit some condition, among others, in an efficient way. It is also really popular in terms of Machine Learning tools. Its documentation can be found in [36].

5.2.4 Scikit-Learn

Scikit-Learn is a quite complete library for Python aimed to help developers to carry out a proper feature extraction from the datasets, implement Machine Learning models (both of regression and classification, and of course, unsupervised learning), and evaluate metrics during testing, without thinking about its implementation. Moreover, it contains example datasets to help beginners learn faster. The whole Scikit-learn project is detailed by its creators in [37] and its documentation can be found in [38].

This library works at the model level. This means that you can create a model, vary its main parameters, train it, evaluate it and make predictions. This simplicity is the main positive point of using this library for models such as Decision Trees, Random Forests etc. Nonetheless, one limitation is that this ease of creating models without implementing them reduces the possibilities for some models, for instance, Artificial Neural Networks. It is possible to implement Neural Networks with this library but it is not the best option if it is required to try non-usual Deep Learning architectures.

5.2.5 Keras

Keras is another Machine Learning Python library dedicated to the Deep Learning field, in other words, Artificial Neural Networks. By using this Library, the user must implement these networks specifying their parameters at the layer level, rather than at the model level like with Scikit-Learn. This makes a bit more complex the implementations of the models but allows the developer much more flexibility. One concrete example of this flexibility is the possibility to combine different types of layers in a network. The documentation of this Library can be revised in [39].

Specifically, Keras has been used in this project over the Tensorflow Backend with Graphic Process Unit (GPU) support. This means that, during the training process, the GPU has been used instead of the Central Process Unit (CPU). This is a common practice, since the exhaustive number of linear operations that must be done to train a Neural Network can be carried out more efficiently if the GPU is employed.

5.2.6 Matplotlib

Once the models have been trained and tested, it is usual to plot the results in graphs in order to facilitate the analysis and understanding of them. The chosen option to do it, has been the Python library Matplotlib. With this library, the developer is able to generate all kind of personalised graphs of data contained in arrays. Its documentation can be found in [40] and the project details are in [41].

5.3 Experiments

Once we have detailed the dataset research and the tools to be used, it is possible to briefly explain what the experiments are based on. First of all, an approach at the user-level classification has been done, using different Machine Learning models. Subsequently, a different method has been applied to the tweet-level classification, by using Natural Language Processing (NLP) techniques. Finally, a combination of both has been implemented in order to improve the tweet-level classification by joining the information provided by the user features and the tweet features.

On each experiment several metrics have been evaluated such as accuracy or AUC (described in Chapter 3). Moreover, the hold-out separation and cross-validation have been applied in the experiments. The complete details about this can be found in Chapter 6.

Chapter 6

Experiments and Results

6.1 Introduction

In this project, several and variate experiments have been carried out. First of all, an approach at the **user level** has been done. In other words, the models have been trained and tested with the user datasets, i. e. using only user-level features, without taking into account the tweet datasets. This is due to the fact that for each user, there are several tweets, so they cannot be analyzed by the same model.

Next, a different method has been applied at the **tweet level**. For this, all the tweets belonging to a Content Polluter have been assumed to be ”polluting” and similarly with the Legitimate Users tweets. With these models, we are able to predict if a specific tweet has been written by a Content Polluter or a Legitimate User.

For these two first strategies, metrics such as accuracy and Receiver Operating Characteristic/ Area Under the Curve (ROC/AUC) have been used. With these values, we will be able to compare the results of each model and conclude if the selected techniques are adequate or not.

Finally, these two approaches have been mixed in order to form a single classifier, in order to improve the tweet level classifier with the user features. Basically, with this final experiment, the optimal classifier will be build.

Concretely, the process explained above is structured in three Studies. The Study 1 corresponds to the user-level approach, based on the use of user-level features. Subsequently, the Study 2 comprises the tweet-level strategy and finally, the Study 3 contains the combination of both methods.

6.2 Study 1: User-Level Features

On this first Study, different classifiers from the library Scikit-Learn have been programmed to obtain different results from the different datasets. However, before training the models, it is necessary to adapt some of the features and create new ones which may help the classifier. In other words, we have to execute what it is commonly known in the Machine Learning field as **feature extraction**.

	NumberOfFollowings	NumberOfFollowers	NumberOfTweets	LengthOfScreenName	LengthOfDescription	DailyTweetsNumber	DailyFollowingsNumber	DailyrFollowersNumber
0	-0.319816	-0.073695	-0.278185	-0.012146	-1.255662	-0.323902	-0.348322	-0.082461
1	-0.319769	-0.053999	1.741038	1.493962	0.154964	2.868848	-0.348252	-0.034616
2	6.123119	-0.039600	-0.025375	1.117435	-0.207769	-0.124296	4.467730	-0.041197
3	-0.311512	-0.069906	-0.275667	-0.765200	0.759518	-0.316274	-0.325192	-0.065113
4	1.167062	-0.065071	0.524342	1.117435	1.122250	1.028653	2.023032	-0.060104

Figure 6.1: New header of the Caverlee dataset with features extracted.

6.2.1 Feature Extraction

The features which are present in Caverlee dataset were shown in Figure 5.1, and those are also the ones obtained from Twitter API for the other datasets. Therefore, all the new features obtained must be from these basic ones.

The first step consisted in converting the features CreatedAt and CollectedAt from date and time format to a number of seconds following the Coordinated Universal Time (UTC) format. In this way, it is possible to handle these features, for instance, to do some operations.

Once those two features were numerical values representing certain numbers of seconds, then it was possible to compute new features. These, together with other computed features, the features that form the final dataset are:

1. **DailyTweetsNumber**: This value represents on average the number of tweets per day published by the user in question.

$$DailyTweetsNumber = \frac{NumberOfTweets}{\frac{CollectedAt - CreatedAt}{24 \cdot 3600}} \tag{6.1}$$

2. **DailyNameFollowers**: This feature expresses the number of followers acquired on average per day by the user.

$$DailyNumberFollowers = \frac{NumberOfFollowers}{\frac{CollectedAt - CreatedAt}{24 \cdot 3600}} \tag{6.2}$$

3. **DailyNameFollowings**:

$$DailyNumberFollowings = \frac{NumberOfFollowings}{\frac{CollectedAt - CreatedAt}{24 \cdot 3600}} \tag{6.3}$$

4. **LengthOfScreenName**: This is the length of the screen name.

5. **LengthOfDescription**: This is the length of the description of the profile written by the user.

Once the new features have been created, all the columns have been standardized so that each one is a Gaussian distribution. In other words, these distributions have mean 0 and standard deviation 1. This is an important step for some models as it was stated in Chapter 4. From this point, the datasets are ready to train the different classifiers used in this Study.

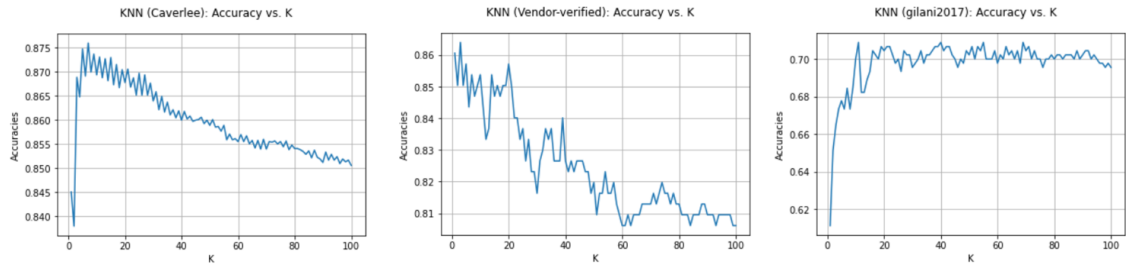

Figure 6.2: Accuracies vs K for the tests of the three datasets. Manhattan distance has been used.

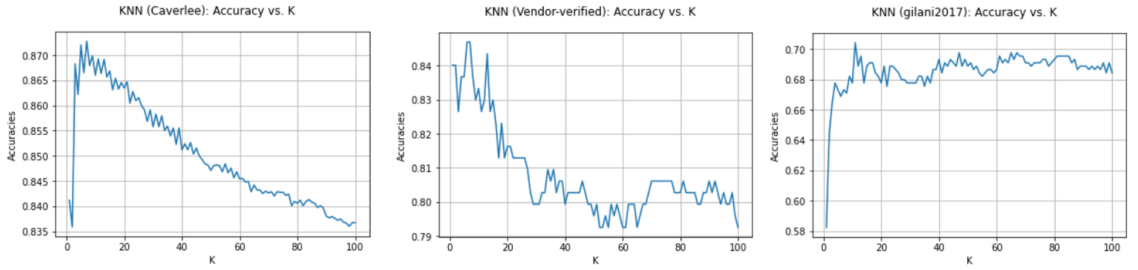

Figure 6.3: Accuracies vs K for the tests of the three datasets. Euclidean distance has been used.

6.2.2 K-Nearest Neighbours

The first model to be used is K-Nearest Neighbours since it is one of the simplest available within the Machine Learning field. In order to find the parameters that make this classifier optimal, before doing a deeper test with several metrics, several attempts have been carried out varying the parameter K and computing the accuracy obtained in each test dataset with a threshold of 0.5. This threshold is used to convert the output probabilities of being Content Polluter, the so-called **a posteriori probability** into pure predictions (Boolean values). Furthermore, this has been repeated for both Manhattan and Euclidean distances. The results for each case are shown in Figures 6.2 and 6.3.

The metric accuracy has been chosen to do this first parameter analysis due to the results of the survey carried out in [2]. This survey was done to the users of the bot detector created by Indiana University and one of its conclusions is that its users (people and/or entities which need to identify Content Polluters) were approximately equally worried about Content Polluters misclassified as Legitimate Users and vice versa (False Negatives and False Positives).

The accuracy metric measures the accounts erroneously classified without taking into account if it is a False Positive or a False Negative, so it is the perfect metric. Moreover, it is the most intuitive. This will give as an idea of which is the optimal model before carrying out the ROC - AUC study on it.

In both Figures 6.2 and 6.3 the reader can observe different behaviours in each dataset when the parameter K (number of neighbours) is varied. From this analysis, the K which produces the maximum accuracy has been registered in Table 6.1. Next, those values have been used to generate optimal models. The ROC curve for each case has been obtained and represented in Figures 6.4 and

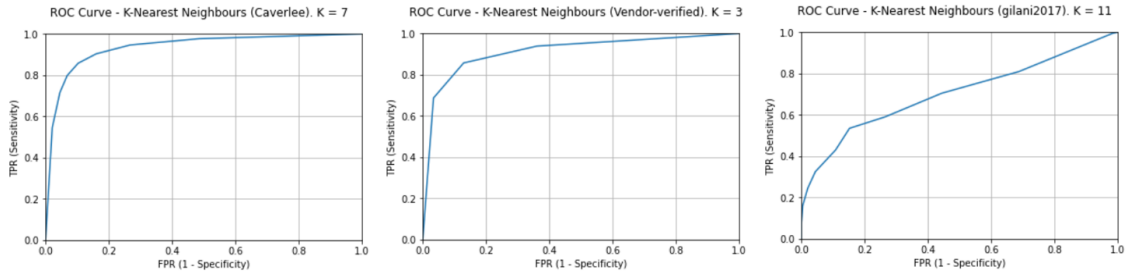

Figure 6.4: ROC Curves for the tests of KNN in Study 1. Manhattan distance has been used.

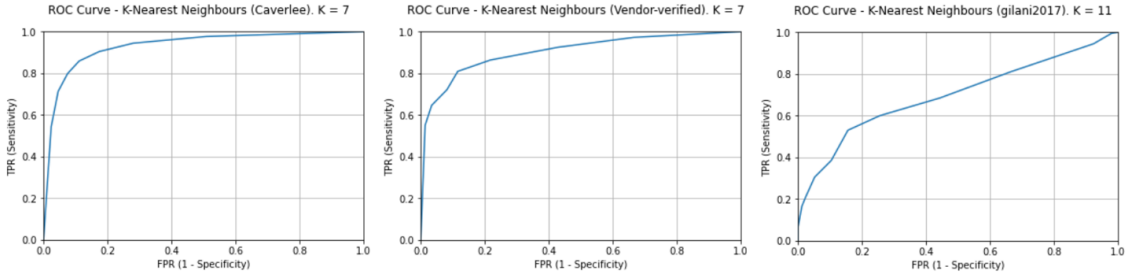

Figure 6.5: ROC Curves for the tests of KNN in Study 1. Euclidean distance has been used.

6.5 . This evaluation curve allows us to have a deeper and more robust idea about the performance of each model during the test process as it is not conditioned by the threshold selected. Logically, the AUC values have also been computed and are shown in Table 6.1.

As observed in the Figures 6.4 and 6.5, the ROC curves obtained for **Caverlee** and **Vendor Verified** datasets are quite similar to the expected aspect, stated in Figure 3.4. Moreover, both AUCs are around 0.90, which is outstanding. Given that AUC is not conditioned by the selected threshold, obtaining good values for the AUC ensures that the classifier is robust and that it is able to classify the 90% of the samples that it receives as input.

In terms of the results provided by **Gilani**, they are not as excellent as the rest. However, they are considerably acceptable as this classifier can classify the 70% of the samples. Furthermore, in all the cases, the accuracy values obtained are correlated with AUC values. In regard to the differences between the results obtained with the different distance functions utilized, there is no noticeable distinction. In most of the cases the best K parameter is the same.

	Manhattan Distance			Euclidean Distance		
	Best K	Accuracy	AUC	Best K	Accuracy	AUC
Caverlee 2011	7	0.876	0.935	7	0.873	0.931
Vendor Verified Mixed	3	0.864	0.913	7	0.847	0.903
Gilani 2017	11	0.709	0.703	11	0.704	0.697

Table 6.1: Values of optimal K and the metrics obtained with those models in Study 1.

	Test Accuracy					AUC				
Caverlee 2011	0.883	0.8835	0.886	0.883	0.873	0.8817	0.942	0.938	0.937	0.933
Vendor Verified Mixed	0.847	0.807	0.834	0.814	0.847	0.883	0.881	0.925	0.883	0.903
Gilani 2017	0.707	0.689	0.675	0.680	0.704	0.725	0.696	0.708	0.695	0.699

Table 6.2: Cross validation for K-Nearest-Neighbours. Euclidean distance has been used.

Datasets	Test Accuracy Average	AUC Average
Caverlee 2011	0.881	0.9364
Vendor Verified Mixed	0.8298	0.8950
Gilani 2017	0.691	0.7046

Table 6.3: Mean values of the results in the cross-validation test for KNN in Study 1.

After this first approach, to ensure the fidelity of these results, another experiment over the optimal model has been done by using **cross validation** splitting the dataset in 5 subsets. In this way, the proportions used for training and testing are similar to the **hold-out** experiment (80% for training and 20% for testing). For each test, the accuracies and AUC have been computed and are represented in Table 6.2. As we can observe, the results are similar for each test. Thus, these results are a sign of **stability** in the model, which suggests that the previous results were not caused by overfitting. The average values are illustrated in Table 6.3.

6.2.3 Decision Tree

Once finished with K-Nearest Neighbours, we can continue with the next model. In the case of Decision Tree, the most relevant parameter is the **maximum depth**. As explained in the theoretical part of this report, a change in this parameter may make the difference between underfitting and overfitting. Thus, it is worth to analyze the accuracy obtained with respect the variation of it, as it was done with parameter K in the KNN classifier. The results for the three datasets are shown in Figure 6.6.

In comparison to the graphs generated with the KNN model in Figures 6.2 and 6.3, with the Decision Tree model, the accuracies seem to variate within ranges of the same size as with KNN. However, in this case, the accuracy ends up stabilizing in a concrete value.

Identically to the previous experiment, the maximum depths that cause the maximum value of the accuracies has been used to generate Decision Tree models and evaluate their ROC curves. These curves are illustrated in Figure 6.7 and their respective AUC values, together with the accuracies obtained are in Table 6.4.

For **Caverlee** and **Vendor-verified**, the obtained ROC curves are even better than the ones obtained with KNN, getting AUC values around 0.95. However, in respect of **Gilani**, the results are quite

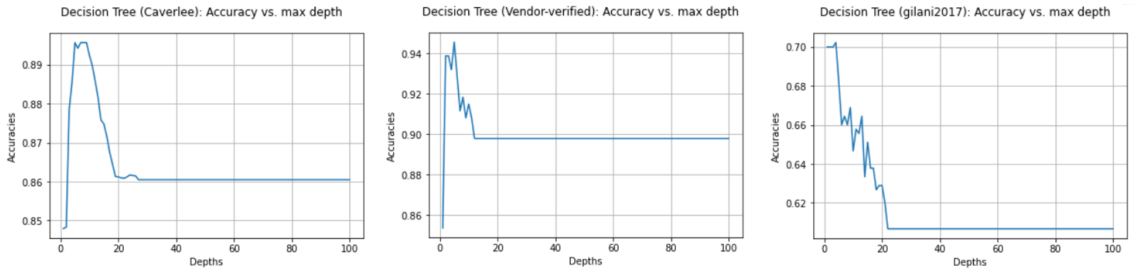

Figure 6.6: Accuracies vs maximum depth for the Decision Tree experiment.

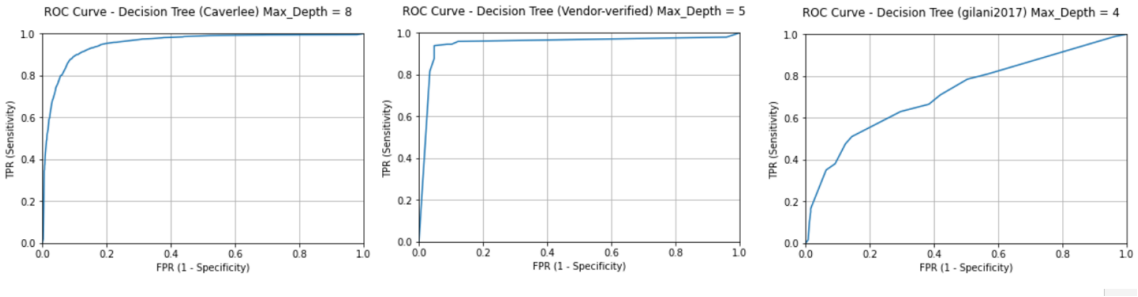

Figure 6.7: ROC Curves for the tests of Decision Tree in Study 1. The maximum depth used is indicated in each graph.

similar in terms of accuracy, causing a slightly greater AUC.

Similarly to the KNN experiment, the final action with this model has consisted of an evaluation based in **cross validation** with 5 folds. The results are shown in Table 6.5. As observed, the values obtained are approximately constant for each test, which demonstrates the stability of the model. In terms of the average values, these can be appreciated in Table 6.6.

6.2.4 Random Forest

As we already know, Random Forest consists of several uncorrelated Decision Trees working as one. Hence, the same analysis of the accuracy with respect the maximum depth allowed can be performed. Concretely, for this experiment, 100 estimators (trees) have been chosen, and the ”create bootstrap dataset” option enabled, technique which is explained in Chapter 4, section 4.4.2. The results for this are presented in Figure 6.8.

Repeating the same methodology once more, the ROC curves for each dataset have been obtained

	Best Maximum Depth	Accuracy	AUC
Caverlee 2011	8	0.896	0.952
Vendor Verified Mixed	5	0.946	0.948
Gilani 2017	4	0.702	0.721

Table 6.4: Results of the Decision Tree experiments in Study 1.

	Test Accuracy					AUC				
Caverlee 2011	0.896	0.913	0.916	0.914	0.896	0.958	0.962	0.961	0.956	0.952
Vendor Verified Mixed	0.946	0.959	0.936	0.939	0.946	0.943	0.966	0.941	0.972	0.948
Gilani 2017	0.757	0.728	0.754	0.732	0.702	0.771	0.744	0.743	0.739	0.721

Table 6.5: Cross validation for Decision Tree in Study 1.

Datasets	Test Accuracy Average	AUC Average
Caverlee 2011	0.9070	0.9578
Vendor Verified Mixed	0.9452	0.9540
Gilani 2017	0.7342	0.7436

Table 6.6: Mean values of the results in the cross-validation test for Decision Tree in Study 1.

by using the optimal maximum depth in each case. The results are collected in Figure 6.9 and Table 6.7.

So far, the best results have been proportioned by the Random Forest approach. We can emphasise the fact that for the **Vendor Verified** dataset, the AUC obtained is almost equal to 1. Moreover, in the case of **Gilani**, the AUC has also increased until a value close to 0.8, with a slightly greater accuracy than with the use of the other models already described. Concerning **Caverlee**, the outcome is almost identical to the Decision Tree modeling.

Ultimately, **cross validation** has been carried out similarly to the other experiments, obtaining the results in Table 6.9. By observing them, it can be stated that each test produced a similar outcome, which shows that the model is stable. The average values are shown in Table 6.12.

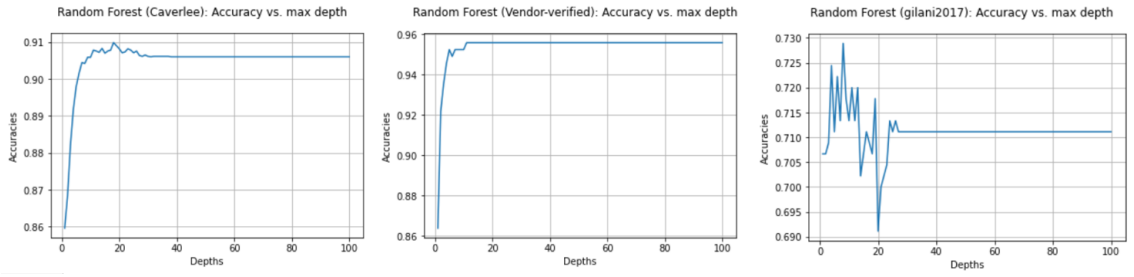

Figure 6.8: Accuracies vs maximum depth for the Random Forest experiment. The number of estimators employed is 100.

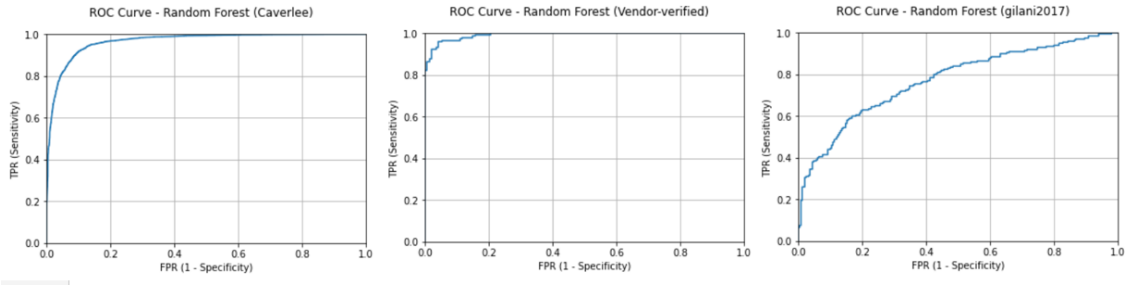

Figure 6.9: ROC Curves for the tests of Random Forest in Study 1. The maximum depth used is indicated in each graph.

	Best Maximum Depth	Accuracy	AUC
Caverlee 2011	18	0.91	0.966
Vendor Verified Mixed	11	0.956	0.992
Gilani 2017	8	0.729	0.773

Table 6.7: Results of the Random Forest experiments in Study 1.

Datasets	Test Accuracy Average	AUC Average
Caverlee 2011	0.9264	0.9744
Vendor Verified Mixed	0.9494	0.9854
Gilani 2017	0.7386	0.7820

Table 6.8: Mean values of the results in the cross-validation test for Random Forest in Study 1.

	Test Accuracy					AUC				
Caverlee 2011	0.925	0.928	0.928	0.926	0.925	0.976	0.973	0.975	0.974	0.974
Vendor Verified Mixed	0.9561	0.946	0.946	0.949	0.956	0.978	0.987	0.983	0.987	0.992
Gilani 2017	0.741	0.743	0.757	0.732	0.729	0.793	0.779	0.797	0.764	0.777

Table 6.9: Cross validation for Random Forest in Study 1.

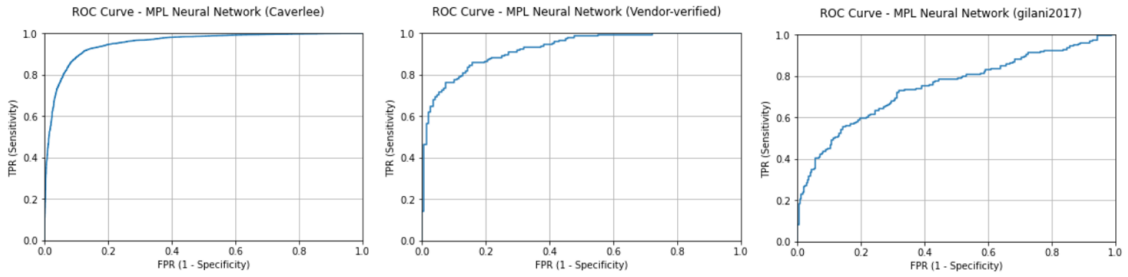

Figure 6.10: ROC Curves obtained from the Multilayer Perceptron models used with all the datasets.

Datasets	AUC
Caverlee 2011	0.953
Vendor Verified Mixed	0.926
Gilani 2017	0.751

Table 6.10: Results of the MLP from Scikit-Learn model in Study 1.

6.2.5 Multi Layer Perceptron (MLP)

This is the last model to be evaluated at the **user level** in this Study. It consists of a standard Dense Neural Network (referenced in Scikit-Learn library as Multi Layer Perceptron). Concretely, the sizes of the hidden layers is 100 with adaptive learning rate and maximum number of iterations (usually known as number of epochs) equal to 200.

Besides, the Activation Function used is the Sigmoid so as to obtain output values within the $[0, 1]$ interval, which can be understood as probabilities. In this way, the ROC Curves can be calculated as it was done with the rest of the models. Once the model has been trained, the acquired ROC curves in the testing process are shown in Figure 6.10, and the respective AUC values in Table 6.10.

In general, the results obtained with the MLP in this Study are notable. For the **Caverlee** dataset, the AUC value is similar to the obtained with Decision Tree and Random Forest. Regarding **Vendor Verified Mixed**, the AUC is somewhat less than the one computed with the Decision Tree and minor than the case in Random Forest. In reference to **Gilani**, is one of the best performances but

	Test Accuracy					AUC				
Caverlee 2011	0.909	0.909	0.910	0.906	0.905	0.966	0.964	0.965	0.962	0.962
Vendor Verified Mixed	0.844	0.847	0.871	0.820	0.847	0.922	0.914	0.938	0.918	0.925
Gilani 2017	0.75	0.721	0.730	0.706	0.728	0.788	0.740	0.773	0.751	0.753

Table 6.11: Cross validation for Multi Layer Perceptron from Scikit-Learn in Study 1.

Datasets	Test Accuracy Average	AUC Average
Caverlee 2011	0.9078	0.9638
Vendor Verified Mixed	0.8458	0.9234
Gilani 2017	0.7270	0.7610

Table 6.12: Mean values of the results in the cross-validation test for Multi Layer Perceptron from Scikit-Learn in Study 1.

still slightly under the Random Forest AUC.

As a final step, **cross validation** has been applied to this model too. The results provided by these test can be checked in Table 6.11. Once more, the similarity in the values obtained in each test ensures the absence of bias in the model due to the dataset train and test splitting. Hence, the model is stable. The final average results can be observed in Table 6.12.

6.2.6 Results Discussion

To sum up, all the models in this set of experiments using Scikit-Learn models have produced acceptable results, and some of them extraordinary. Generally, **Caverlee** and **Vendor Verified Mixed** have produced the best values in their tests, reaching in some cases values of 0.95 AUC. However, **Gilani** has not provoked such fantastic results but are still acceptable, since its AUC oscillates within the interval $[0.70 - 0.75]$ in most of the cases, reaching greater values than 0.75 in Random Forest experiment.

This was expected since **Caverlee** is the dataset with more training examples, so it should be the dataset which better trains the models. In regard to **Vendor Verified Mixed**, it is logic that it is easy to distinguish between Content Polluters and verified Legitimate Users, since the verified accounts are usually more active, may have a greater number of followers on average, etc. Thus, the most difficult case, in comparison, is **Gilani** given that these are Content Polluters and Legitimate Users randomly selected and it contains a limited number of samples in contrast to **Caverlee**.

6.2.7 Dense Neural Network: Keras

This new experiment consists in the analysis of the same user datasets, with a Deep Learning approach, similarly to the Multi Layer Perceptron but using a different Neural Network architecture. Therefore, in this experiment, the library Keras will be used since it is more powerful in terms of performance and possibilities in Artificial Neural Networks. Furthermore, thanks to the fact that this library includes GPU support, the training/testing process is faster.

The concrete network employed is detailed in Figure 6.11. As it is observed, it is composed by 3 Dense layers (formed by neurons which are fully connected to the neurons in both the previous and subsequent layer). The three layers contain 8, 64 and 1 neurons respectively. Moreover, the Sigmoid Activation Function is used in the output of the first layer and in the output (necessary in the output to obtain probabilities).

The training process is formed by several iterations over the training data. Each of these iterations

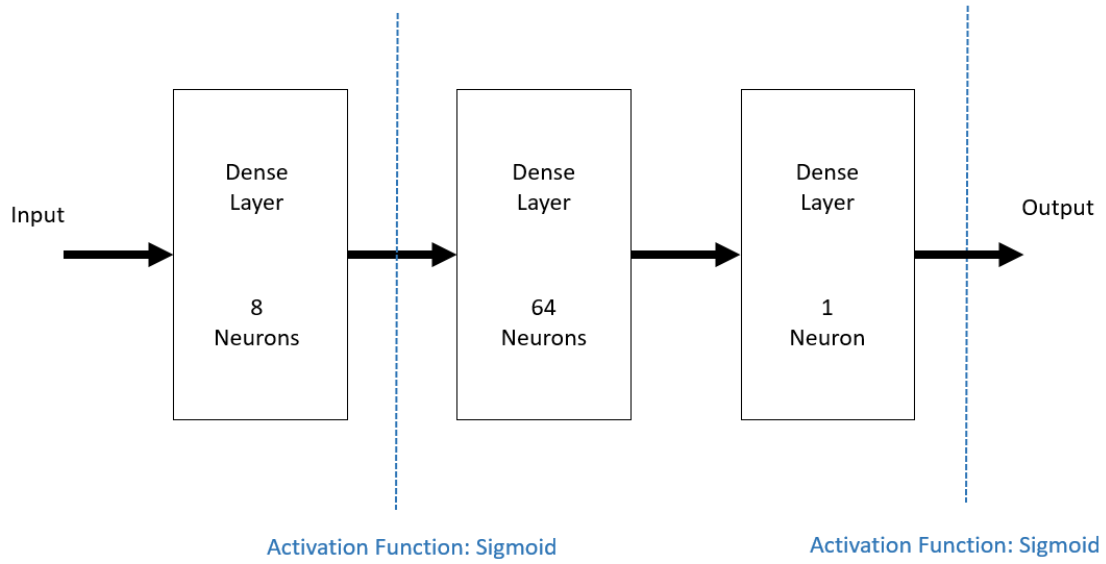

Figure 6.11: Dense Neural Network from Keras library used in Study 1.

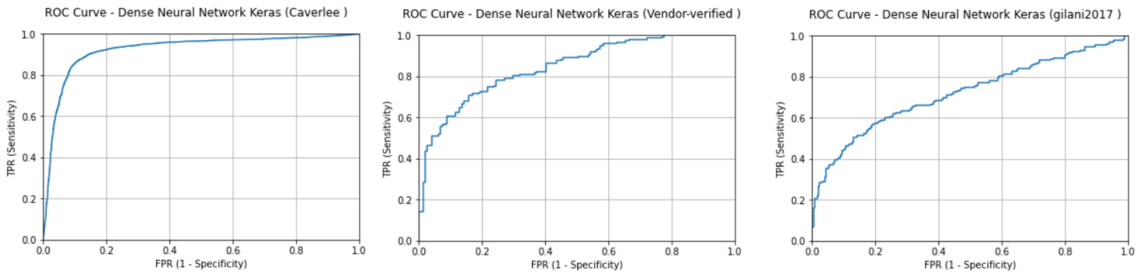

Figure 6.12: ROC Curves obtained from the Dense Neural Network from Keras Library of Study 1.

		Training		Testing		
	Number of Epochs	Loss	Accuracy	Loss	Accuracy	AUC
Caverlee 2011	90	0.0485	0.87	0.0479	0.873	0.923
Vendor Verified Mixed	90	0.0886	0.766	0.0902	0.762	0.846
Gilani 2017	90	0.1026	0.675	0.1040	0.66	0.722

Table 6.13: Results obtained from the Dense Neural Network from Keras Library in Study 1. The values of Loss and Accuracy in training make reference to the last epoch.

	Test Accuracy					AUC				
Caverlee 2011	0.866	0.873	0.873	0.862	0.872	0.921	0.925	0.925	0.918	0.926
Vendor Verified Mixed	0.759	0.793	0.756	0.725	0.735	0.824	0.863	0.841	0.798	0.826
Gilani 2017	0.711	0.693	0.634	0.651	0.675	0.769	0.738	0.653	0.743	0.719

Table 6.14: Cross validation for Dense Neural Network from Keras Library in Study 1.

Datasets	Test Accuracy Average	AUC Average
Caverlee 2011	0.8692	0.9230
Vendor Verified Mixed	0.7536	0.8304
Gilani 2017	0.6718	0.7244

Table 6.15: Mean values of the results in the cross-validation test for Dense Neural Network from Keras Library in Study 1.

are called ”Epochs” in the common nomenclature. During this training process, the parameters Loss and Accuracy have been monitored on each Epoch, and the values from the last epoch have been registered. This is important to compare these values with the Accuracy of the test evaluation. Moreover, getting an Accuracy of 1 in the training process could be a sign of Overfitting. Hence, registering these values is important.

Once each model is trained, two metrics have computed in this Study: the computation of the Accuracy for the whole testing dataset, and the calculation of the ROC Curve and the AUC. These results are displayed in Figure 6.12 and Table 6.13. With respect to the numerical values, **Caverlee** has obtained similar results to the previous experiments, with an AUC of 0.92 and an accuracy in testing of 0.873.

However, **Vendor Verified Mixed** acquired worse results than in the other experiments. The Accuracy in testing is similar to the one obtained with KNN, but its AUC is still lower (0.846 versus 0.9). Moreover, the rest of values which are outcome of the rest of classifiers in Study 1 are more favourable. On the other hand, the results for **Gilani** are quite similar to the metrics in the general Study 1, as the AUC obtained is 0.722. In contrast, the test accuracy is 0.66, which is slightly lower than all the accuracies present in the rest of this Study, where the minimum accuracy is 0.7 even though the AUC parameter is a more robust evaluation parameter than the Accuracy with fixed threshold of 0.5. In any case, the biggest AUC is still provided by the Random Forest model in Study 1.

Moreover, **cross validation** has been applied identically to the other experiments in Study 1 and the results can be examined in Table 6.14. The numerical values obtained of both accuracy and AUC are approximately similar for all the testes, which ensures the stability, except for **Gilani**, which presents more variability. However, these variations are not alarming, since the maximum variation is of 0.1 in terms of AUC. The average results from the cross-validation test are represented in

Table 6.15.

6.2.8 Results Discussion

In conclusion, the performance and results produced by this Deep Learning approach are quite acceptable and could be a proper classifier. However, in comparison to the rest of the models of Study 1, this technique is not the optimal option, given that Decision Tree and Random Forest present a better behaviour in two of the three datasets (**Caverlee** and **Vendor Verified Mixed**).

This may be due to the fact that we are not using the optimal architecture for carrying out this task. In contrast to Neural Networks, it is easier to find the optimal parameters that define the models when working with KNN, Decision Tree and Random Forest. In order to do a proper comparison, a deeper study on the architecture of the Neural Network would be required, and this is out of the ambit of this project.

6.3 Study 2: Tweets-Level Features

This new Study is focused on the tweet level. The matter is that the features in the original **Caverlee** dataset are reduced. Therefore, some knowledge must be extracted from the tweet content, which makes the Feature Extraction process quite different from the user level approach. In other words, some techniques from the Natural Language Processing (NLP) field must be taken into account.

6.3.1 Word Frequencies Experiment

6.3.1.1 Feature Extraction: Word Frequencies

For this concrete analysis, the frequency of appearance of each word is going to be used. However, each tweet must be cleaned before counting them, since it contains mentions, links and hashtags. If these ”special words” were not removed, they would be considered as different words, since each single mention is different from the others (this also happens with each link and each hashtag). However, if we count them and consider their repetitions as features, they will contribute positively in the study.

Moreover, if a tweet is a retweet (a repetition from a tweet published by other account), then the characters ”RT” appear at the beginning of each tweet. These references have been cleaned too and added as new feature. Otherwise, the model would not be able to distinguish a retweet from a retweet request written by the user. These, together with other new features are shown in the Figure 6.13. If we examine them we have:

1. **URLNum**: Number of links in the tweet.
2. **MentionNum**: Number of mentions in the tweet.
3. **HashtagNum**: Number of hashtags in the tweet.
4. **IsRetweet**: This feature indicates if the tweet in question is a retweet or has been directly published by the user from it was collected.

	ContentPolluter	URLNum	MentionNum	TweetLength	HashtagNum	IsRetweet
0	1	1.125516	-0.535283	0.361302	-0.255592	-0.202913
1	0	1.125516	-0.535283	1.554321	-0.255592	-0.202913
2	0	-0.811869	2.003590	1.387854	-0.255592	4.928227
3	0	-0.811869	-0.535283	-1.331120	-0.255592	-0.202913
4	0	-0.811869	-0.535283	-1.164652	-0.255592	-0.202913

Figure 6.13: Header of the Caverlee tweets dataset with extracted features. The values have been standardised.

5. **TweetLength**: Length of the tweet before removing the links, mentions, hashtags and retweet indicators.

Once the tweets have been cleaned, the word count has been done. Considering each word as a token, using the adequate functions provided by the Scikit-Learn library, the tweets have been ”vectorized”. The result is a matrix whose columns are the hashes of each token and its rows are the number of repetitions of each token in the tweet (each row is a tweet).

Given that there are a lot of tokens which only appear in a few tweets, the outcome matrix contains a vast number of zeroes. Thus, the Sparse Matrix format is used. With this format, which only stores the positions of the non zero elements, it is possible to store information about thousands of tweets, due to its efficiency in memory.

Moreover, a usual Stop Words list has been used in order to reduce the computational cost. This is a common practice in the NLP field, which consists in deleting meaningless words such as prepositions. Given that we only work with the frequencies of the words in this experiments, without taking into account the order, the appearance of such words in a tweet does not give relevant information.

Given that the tweets dataset are much bigger than the user datasets, performing **cross validation** has been avoided due to the high computational cost. Instead, only **hold-out** has been applied.

6.3.1.2 Naive Bayes

The first model to be employed in this Study is the Naive Bayes classifier. Due to its naive assumption, this model has a great performance with huge datasets. This is perfect for this case, where the word frequencies are vast datasets.

Likewise in the other experiments already described, the respective ROC curves and AUC have been computed. The results obtained with this model can be appreciated in Figure 6.14 and in Table 6.16.

Analyzing the results, we find that this technique performed well in terms of the **Caverlee** and **Vendor Verified** with AUCs of 0.85. However, **Gilani** obtained an AUC of 0.63, which is not the best result considering that an AUC of 0.5 implies that the model is classifying randomly.

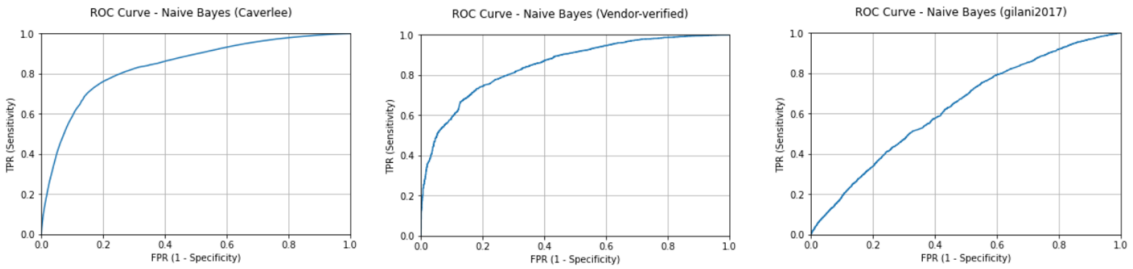

Figure 6.14: ROC Curves obtained with the model Naive Bayes in Study 2.

Datasets	AUC
Caverlee 2011	0.84
Vendor Verified Mixed	0.851
Gilani 2017	0.633

Table 6.16: Results of the Naive Bayes experiments in Study 2.

6.3.1.3 Decision Tree

The next approach is based in a Decision Tree classifier. Considering that the size of the datasets is big (especially **Caverlee**), the study of the optimal maximum depth has not been done in this approach. The chosen values for this parameter has been 15 for **Caverlee** and 20 for the remaining datasets.

The final results are shown in Figure 6.15 and in Table 6.17. In the case of Caverlee the AUC is slightly less than the value obtained with Naive Bayes. The differences are more remarkable in the other classifiers, obtaining a significantly worse AUC in the case of **Vendor Verified**, which has decreased from 0.85 to 0.72. Moreover, the AUC for **Gilani** is even lower (0.588), which is not an acceptable result.

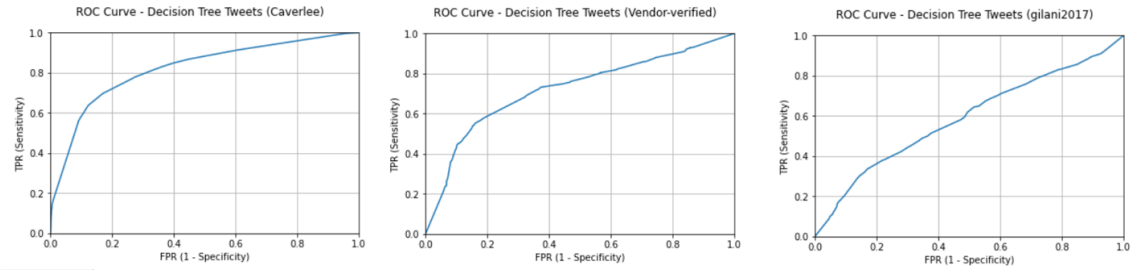

Figure 6.15: ROC Curves obtained with the Decision Tree approach in Study 2.

Datasets	AUC
Caverlee 2011	0.82
Vendor Verified Mixed	0.721
Gilani 2017	0.588

Table 6.17: Results of the Decision Tree experiments in Study 2.

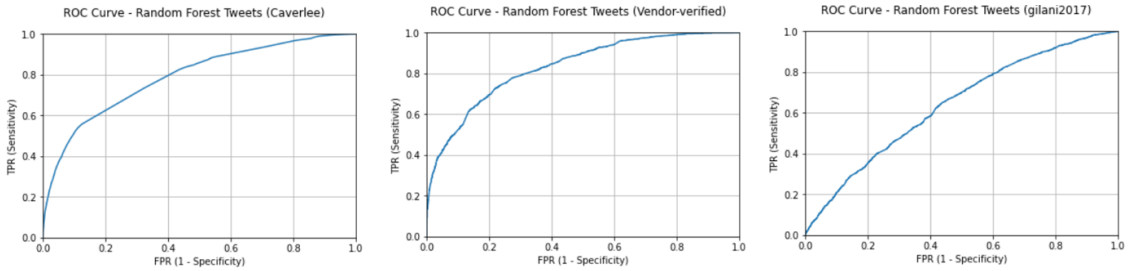

Figure 6.16: ROC Curves obtained with the Random Forest approach in Study 2.

6.3.1.4 Random Forest

Finally, this is the last model used with the word frequencies approach. [1]. Identically to the Decision Tree case, examining the accuracy versus the maximum depth parameter is costly regarding time consuming. Namely, the employed parameters for these experiments have been 65 estimators and maximum depth of 15 for **Caverlee**, and **Vendor Verified**. For **Gilani**, the model used is composed by 100 estimators and maximum depth of 20.

As the reader may have been expected, the results produced by this model are generally better than the Decision Tree outcome. This is clearly appreciated since **Vendor Verified** and **Gilani** obtained AUCs of 0.827 and 0.64 respectively. The only exception is **Caverlee**, which obtained a slightly minor AUC (0.791). However, this is a minimum difference to the value obtained in the previous experiment, so it seems fair to state that the performance on this dataset has been similar to the Decision Tree. All these values and their respective ROC Curves are displayed in Figure 6.16 and Table 6.18.

[1] KNN has not been used due to its inefficiency in making predictions for large dataset such as **Caverlee**. Furthermore, MLP from Scikit-Learn has also been discarded because this library does not include GPU support, so training Neural Networks with enormous datasets is computationally exhausting.

Datasets	AUC
Caverlee 2011	0.791
Vendor Verified Mixed	0.827
Gilani 2017	0.64

Table 6.18: Results of the Random Forest experiments in Study 2.

		Training		Testing		
	Number of Epochs	Loss	Accuracy	Loss	Accuracy	AUC
Caverlee 2011	5	0.0637	0.8138	0.0658	0.8052	0.869
Vendor Verified Mixed	120	0.0522	0.8484	0.0738	0.7736	0.8515
Gilani 2017	120	0.0987	0.6792	0.1077	0.6316	0.667

Table 6.19: Results from LSTM Neural Network in Study 2. The values of Loss and Accuracy in training make reference to the last epoch.

6.3.1.5 Results Discussion

Comparing the overall set of metric values obtained so far from this Study, we find that the model with best performance has been Naive Bayes. Nevertheless, Random Forest has obtained approximately similar results and considering that Naive Bayes is based on the ”naive assumption”, then Random Forest seems to be the best model that would be chosen in order to make predictions.

As it was similarly observed in Study 1, **Gilani** has been the dataset with more difficulties to classify not only its users, but also its tweets.

6.3.2 LSTM Neural Network Experiment

This part of the Study consists of an approach to classify at the tweet level using Deep Learning. Moreover, it is totally different from the solution proposed so far in Study 2. In this experiment, the NLP technique which is employed is the use of Long Short-Term Memory (LSTM) Neural Networks. As it is explained in Chapter 4, subsection 4.4.3, this is a type of Recurrent Neural Network (RNN).

RNN are based on the fact that they own feedback connections, which allow the model to memorize the context where some features appear. The advantage of this strategy over the hashing and counting method previously carried out, is that the order of the words in the tweet is considered, and therefore, the context in which every word appear is known, whereas with word frequencies there is no way of evaluating it.

6.3.2.1 Feature Extraction for the LSTM Neural Network

By using this technique, the feature extraction process is also different. Each tweet must be tokenized in order to be able to be the input of the neural network. This means that each word must be encoded with an integer value, unique in the whole dataset. Moreover, once the tweets are tokenized, they must be padded with zeroes so as to ensure that all the tweets have the same length; and therefore, the encoded tweets can be introduced to the network.

The same problem regarding the links, hashtags and mentions is present here, as it was in the word frequencies approach. If they are tokenized together with the rest of the tweet, all the mentions will receive different integers (similar to the links and hashtags). One solution could be removing

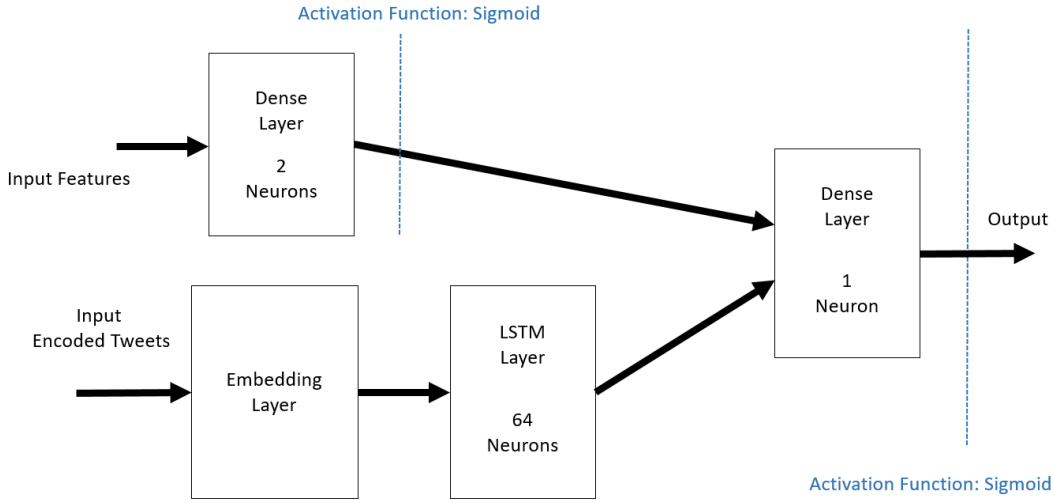

Figure 6.17: Architecture of the LSTM Neural Network used in Study 2.

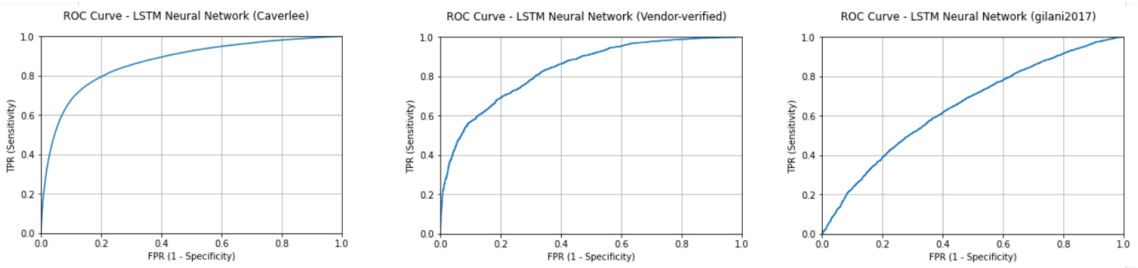

Figure 6.18: ROC Curves for the LSTM Neural Network model in Study 2.

and counting them as it was done in the previous methodology.

However, we can take advantage of the capacity of this model to analyze the order of words by encoding every mention as the same token (and the links and hashtags). Hence, the mentions, links and hashtags have been encoded with the words ”COD_MENTION_COD”, ”COD_HASHTAG_COD”, ”COD_URL_COD”. In this way, these ”special words” will be encoded with the same value. As a final step, the ”IsRetweet” feature has been added similarly to chapter 1 simultaneously to the computation of the length of the tweet.

6.3.2.2 Architecture Employed and Results

In this experiment we have to combine both usual features from the dataset (”**TweetLength** and **IsRetweet**), and the tokenized tweets as the input of the Neural Network. Therefore, the architecture in Figure 6.17 has been implemented. As it can be observed, the features are introduced to a Dense Layer of 2 neurons and the encoded tweets will enter the Embedding Layer. The Embedding layer processes the inputs as Dense Vectors which represent the projection of each word in a continuous space. Then, the LSTM layer processes the outputs of the Embedding Layer. Finally, the last Dense Layer evaluates both outputs from the Dense Layer for features and the LSTM Layer, producing the outputs of the model.

The final results are observed in Table 6.19 and the respective ROC Curves in Figures 6.18. We can observe that **Caverlee** acquired an AUC of 0.87 with a testing accuracy of 0.8. All this using only 5 epochs (the number of epochs is more reduced for this dataset due to its enormous size).

Similarly, **Vendor Verified** generated an AUC of 0.85, with a testing accuracy of 0.77. These two results are exceptional, considering the assumption of labeling all the tweets by a Content Polluter as ”polluting”. Similarly to the rest of the tests, **Gilani** acquired a lower AUC (0.66) and a lower accuracy.

6.3.2.3 Results Discussion

According to our suppositions, this model has provided a better performance in comparison to the techniques employed in the word frequencies experiment. Thus, this means that the LSTM Neural Network has taken advantages of extracting knowledge from the order of the sentences.

Although all the AUC values are greater than the overall results obtained in the other approach, these are still quite similar to the outcome obtained from Naive Bayes. As stated in the word frequencies approach, other models are preferred when facing similar results due to the fact that this model relies on a naive assumption.

6.4 Study 3: Users and Tweets

6.4.1 Methodology and Results

The final method to be used in this project consists of a final study combining the techniques of both approaches: **the user level and the tweet level**. Concretely, it has been decided to use the tweet level in this final approach, since our tweet level approaches generated worse results than the user level approach which were outstanding.

In order to carry out this experiment, new datasets have been created by adding the features of each user to their respective tweets. In this way, the user features will add information to the tweet content, hopefully improving the classification.

The model selected to this experiment has been the LSTM Neural Network used in Study 2, whose architecture is illustrated in Figure 6.17. The only modification has been the increase in the number of neurons of the Dense Layer corresponding to the features input from 2 to 8, in order to improve the capacity of the model since the number of input features has grown.

This model has been chosen due to its performance in the previous experiments, since its results were rated as the best regarding the tweet level. In terms of evaluation, it has been carried out identically to the Study 2. **Hold-out** has been used, discarding the possibility of using **cross validation** due to the computational costs.

6.4.2 Results Discussion

The ROC curves are shown in Figure 6.19 and the accuracies and AUC values can be appreciated in Table 6.20. As we can notice, the results for each dataset have improved with respect to the

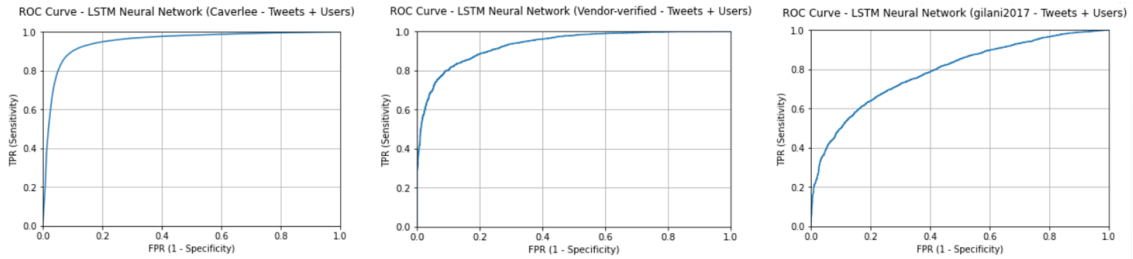

Figure 6.19: ROC Curves for the models in Study 3: Users and Tweets.

		Training		Testing		
	Number of Epochs	Loss	Accuracy	Loss	Accuracy	AUC
Caverlee 2011	5	0.0366	0.9005	0.0364	0.8992	0.949
Vendor Verified Mixed	150	0.0374	0.9007	0.0486	0.8546	0.932
Gilani 2017	180	0.0744	0.7759	0.0858	0.7279	0.796

Table 6.20: Results obtained from the LSTM Neural Network in Study 3.

values in Study 2: LSTM approach. The AUCs for **Caverlee** and **Vendor Verified** have improved from 0.85 to 0.95. Moreover, the AUC of **Gilani** has also risen from 0.66 to 0.796. Ultimately, this model has exceeded all the models regarding the tweet level approach.

In comparison to the user level approach, these results surpass the values obtained by all the models, excluding only the Random Forest at the user level in Study 1, which has similar values for **Caverlee** and **Gilani** and greater values for **Vendor Verified**.

In conclusion, the combination of both approaches has resulted in an ameliorated model in terms of the tweet level classification. This suggests that the user features when analyzing individual tweets are quite relevant.

Chapter 7

Conclusions and Future Projects

7.1 Conclusions

In this project, different Machine Learning models have been evaluated to discriminate between Content Polluters and Legitimate Users. To do this, firstly a social analysis of Twitter as a social network platform has been done, this includes an study of its functioning and its context. Next, a proper examination of the Machine Learning field at a theoretical level has been carried out, in order to identify which techniques are available. The following step has consisted in doing some research to find public datasets related to Content Polluters on the web. Given that it was possible to find appropriate datasets, it was decided not to obtain our own datasets. Subsequently, different experiments have been executed in order to test the possibilities of Machine Learning classifiers in the context of the task defined for this project. Moreover, all the results have been processed, obtaining different metrics which allow us to perform a critical evaluation of them. Finally, the conclusions of the project have been acquired according to the results obtained. Basically, it seems fair to state that all the objectives suggested in the introduction of this project have been properly achieved.

Once finished all the experiments and obtained all the results, it is possible to extract the conclusions of this project in a critical way: Thus, it is feasible to declare that:

1. **It has been possible to differentiate Content Polluters from Legitimate Users at the user level by using Supervised Learning approaches in this project:** This has been demonstrated by the outstanding results obtained in Study 1, since the values of the metrics for **Caverlee** and **Vendor Verified** were spectacular. At the same time, the results for **Gilani** were acceptable.

2. **The best model found in this project for classifying Content Polluters at the user level is the Random Forest:** According to the results, the performance of Random Forest surpasses the metric values corresponding to KNN, Decision Tree and Dense Neural Networks in Study 1 in terms of user level classification.

3. **It has been possible to classify Content Polluters tweets by only using its content and relative features to it:** The experiments at the tweet level have shown that most of the Machine Learning models produced more than acceptable results.

4. **LSTM Neural Networks generated, in this study, slightly better results than usual Machine Learning models using only word frequencies:** The results acquired by the LSTM Neural Network used in Study 2 got better values than the word frequencies approach. However, the difference is not as big as expected, so the word frequencies technique should not be underrated.

5. **It has been easier to classify at the user level than at the tweet level in this project:** Generally, the classifiers used in the user level approach provided better results than the models trained at tweet level. This may be caused by either the difficulty of evaluating the contents of the tweets by NLP techniques or the high relevance of the user level features. Of course, it could also be due to undetected bias in the datasets or similar reasons.

6. **Distinguishing Content Polluters from Legitimate Users verified by Twitter has been easier when facing reduced number of samples in the dataset.** The experiments have demonstrated that, according to our initial thoughts, the results have been better for **Vendor-verified** than for **Gilani** dataset. However, when dealing with a large enough dataset, such as **Caverlee**, it is perfectly possible to carry out the classification. This is applied to both the user and tweet level classification.

7. **Combining the user level and tweet level approaches, the tweet level detection noticeably improves:** The results from Study 3 obtained a better outcome than the rest of the tweet level approaches. That is the optimal solution for the tweet level classification.

8. **It is possible to classify current Content Polluters by Supervised Learning methods if training datasets are available:** It has been feasible to carry out the classification in all the three datasets used in this project, some with better results than others. Given that the datasets belong to the years 2011, 2017 and 2019, it seems fair to state that it is possible to detect them by using Supervised learning approaches, even though they are constantly evolving. However, the difficulty may be to get datasets of evolved Content Polluters.

7.2 Future Projects

This project has been focused in a pure Supervised Learning approach. However, in the recent years, some unsupervised approaches have been proposed. For instance, Cresci et. al. evaluated this possibility by means of clustering in [19]. This approach is totally different that the one carried out on this project because it needs distinct datasets, which contain sequences of actions done by each account rather than numerical features such as the number of followers. Thus, it would be interesting in a future project to try both approaches on the same set of accounts and do a comparison.

Another project could be focused on studying the concrete relevance of the features used in this analysis in terms of classifying Content Polluters. Moreover, the possibility of adding new features and collecting new datasets can be also explored.

A similar project based on the same basic idea of identifying Content Polluters could be carried out with other social networks such as Facebook, Instagram, etc. A comparison could be done so as to find out in which sites Content Polluters have a greater impact or where they are more difficult to be detected.

Another option could be the development of an application which uses the conclusions of this project to implement an optimal model, dedicated to analyze the chances of a Twitter account, introduced by the user being, Content Polluter. In other words, something similar to the Indiana University Project, but oriented to smartphones.

Similarly to the smartphone application suggestion, a Content Polluter classification system could be integrated in a plug-in/extension for some web browser. In this case, this extension could extract the required information from an account in Twitter and classify it by using the models studied in this project.

References

[1] *Digital revolution - Nature*. https://www.nature.com/articles/d41586-018-07500-z. Accessed: 2020-05-18.

[2] Kai-Cheng Yang et al. "Arming the public with artificial intelligence to counter social bots". In: *Human Behavior and Emerging Technologies* 1.1 (2019), pp. 48–61.

[3] *Social media is failing miserably at battling the spread of coronavirus misinformation - Independent UK*. https://www.independent.co.uk/voices/social-media-coronavirus-fake-news-misinformation-5g-conspiracy-theories-a9497926.html. Accessed: 2020-05-19.

[4] *5G Virus Conspiracy Theory Fueled by Coordinated Effort - Bloomberg*. https://www.bloomberg.com/news/articles/2020-04-09/covid-19-link-to-5g-technology-fueled-by-coordinated-effort. Accessed: 2020-05-19.

[5] *Facebook investigates army of bots giving massive "likes" to Spain's health ministry - El Nacional*. https://www.elnacional.cat/en/news/coronavirus-facebook-bots-likes-spain-health-ministry_494676_102.html. Accessed: 2020-05-19.

[6] Kyumin Lee, Brian David Eoff, and James Caverlee. "Seven months with the devils: A long-term study of content polluters on twitter". In: *Fifth international AAAI conference on weblogs and social media*. 2011.

[7] *12 ways Twitter changed our lives*. https://www.theguardian.com/technology/2016/mar/21/12-ways-twitter-changed-our-lives-10th-birthday. Accessed: 2020-06-01.

[8] Andrew G Reece and Christopher M Danforth. "Instagram photos reveal predictive markers of depression". In: *EPJ Data Science* 6.1 (2017), pp. 1–12.

[9] *Leading countries based on number of Twitter users as of April 2020*. https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries/. Accessed: 2020-06-01.

[10] *Six ways Twitter has changed the world*. https://theconversation.com/six-ways-twitter-has-changed-the-world-56234#:~:text=Twitter%20has%20changed%20celebrity%20culture,strict%20control%20from%20their%20management.. Accessed: 2020-06-02.

[11] *Distribution of Twitter users worldwide as of April 2020, by age group*. https://www.statista.com/statistics/283119/age-distribution-of-global-twitter-users/. Accessed: 2020-06-01.

[12] *Developer Agreement and Policy*. https://developer.twitter.com/en/developer-terms/agreement-and-policy. Accessed: 2020-06-02.

[13] Chao Yang, Robert Harkreader, and Guofei Gu. "Empirical evaluation and new design for fighting evolving twitter spammers". In: *IEEE Transactions on Information Forensics and Security* 8.8 (2013), pp. 1280–1293.

[14] VS Subrahmanian et al. "The DARPA Twitter bot challenge". In: *Computer* 49.6 (2016), pp. 38–46.

[15] Clayton Allen Davis et al. "Botornot: A system to evaluate social bots". In: *Proceedings of the 25th international conference companion on world wide web*. 2016, pp. 273–274.

[16] Gang Wang et al. "Social turing tests: Crowdsourcing sybil detection". In: *arXiv preprint arXiv:1205.3856* (2012).

[17] Stefano Cresci et al. "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*. 2017, pp. 963–972.

[18] Sneha Kudugunta and Emilio Ferrara. "Deep neural networks for bot detection". In: *Information Sciences* 467 (2018), pp. 312–322.

[19] Stefano Cresci et al. "DNA-inspired online behavioral modeling and its application to spambot detection". In: *IEEE Intelligent Systems* 31.5 (2016), pp. 58–64.

[20] Stuart J. Russell and Peter Norvig. *Artificial Intelligence. A Modern Approach - Third Edition*. California Technical Pub, 1995, p. 1132. ISBN: 0-13-604259-7.

[21] *Underfitting and Overfitting in Machine Learning*. https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/. Accessed: 2020-06-16.

[22] *Understanding AUC - ROC Curve*. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5/. Accessed: 2020-06-16.

[23] *Machine Learning Classifiers*. https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623. Accessed: 2020-06-19.

[24] *Classification and Regression Analysis with Decision Trees*. https://towardsdatascience.com/https-medium-com-lorrli-classification-and-regression-analysis-with-decision-trees-c43cdbc58054. Accessed: 2020-06-20.

[25] *Understanding Random Forest*. https://towardsdatascience.com/understanding-random-forest-58381e0602d2. Accessed: 2020-06-22.

[26] *5 Regression Loss Functions All Machine Learners Should Know*. https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0. Accessed: 2020-06-23.

[27] *Building A Deep Learning Model using Keras*. https://towardsdatascience.com/building-a-deep-learning-model-using-keras-1548ca149d37. Accessed: 2020-06-23.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[29] *Understanding LSTM Networks*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2020-06-24.

[30] *Naive Bayes Explained*. https://towardsdatascience.com/naive-bayes-explained-9d2b96f4a9c0. Accessed: 2020-06-24.

[31] Zafar Gilani et al. "Of bots and humans (on twitter)". In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. 2017, pp. 349–354.

[32] *Bot Respository - Indiana University*. https://botometer.iuni.iu.edu/bot-repository/index.html. Accessed: 2020-06-26.

[33] *About Verified Accounts*. https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts. Accessed: 2020-06-29.

[34] *Kaggle*. https://www.kaggle.com/. Accessed: 2020-06-29.

[35] *Popular Twitter bots*. https://www.kaggle.com/fourtonfish/popular-twitter-bots/data. Accessed: 2020-06-29.

[36] *Pandas Documentation*. https://pandas.pydata.org/docs/. Accessed: 2020-06-29.

[37] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[38] *Scikit-Learn Documentation*. https://scikit-learn.org/stable/. Accessed: 2020-06-29.

[39] *Keras Documentation*. https://keras.io/. Accessed: 2020-06-29.

[40] *Matplotlib Documentation*. https://matplotlib.org/3.2.1/contents.html. Accessed: 2020-06-29.

[41] John D Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in science & engineering* 9.3 (2007), pp. 90–95.