UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y
SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DEVELOPMENT OF A COVID-19 MISINFORMATION DETECTION SYSTEM USING MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING TECHNIQUES

ÓSCAR PARRO SAINZ
JULIO 2023

TRABAJO DE FIN DE GRADO

Título:	Desarrollo de un sistema de detección de desinformación sobre el COVID-19 utilizando técnicas de Procesamiento del Lenguaje Natural y Aprendizaje Automático
Título (inglés):	Development of a COVID-19 Misinformation Detection System using Machine Learning and Natural Language Processing Techniques
Autor:	Óscar Parro Sainz
Tutor:	Óscar Araque Iborra
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DEVELOPMENT OF A COVID-19 MISINFORMATION DETECTION SYSTEM USING MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING TECHNIQUES

Óscar Parro Sainz

Julio 2023

Resumen

Desde la aparición del COVID en 2019, este virus no solo se ha propagado rápidamente, sino que también se ha transmitido mucha desinformación en todo el mundo a través de noticias o usuarios de Internet. Estas noticias falsas no solo han causado falta de información, sino también trastornos en la sociedad e incluso consecuencias mortales en problemas de salud. Por lo tanto, ser capaz de entender y mitigar dicha desinformación sobre el COVID-19 no sólo tiene interés académico, sino también un enorme impacto social.

En los primeros años de propagación del COVID-19, la mayor parte de la sociedad no entendía realmente el virus ni cómo actuar contra él. Sin embargo, con el paso de los años, la información y los datos han aumentado, ampliando el conocimiento de los expertos sobre el virus. Aun así, muchos medios de comunicación o internautas siguen difundiendo información falsa. Detrás de estas acciones se articulan en numerosas ocasiones estrategias para manipular a la opinión pública y erosionar la estabilidad de los Estados y sus instituciones.

El objetivo principal de este proyecto, dentro del campo de la Inteligencia Artificial, es utilizar técnicas de Procesamiento del Lenguaje Natural (PLN) y Aprendizaje Automático (AA) para generar un sistema de detección automática de noticias falsas en relación con COVID. El sistema desarrollado se utilizará para escanear toda la información encontrada en las diferentes fuentes de los medios de comunicación, como noticias, artículos o posts en redes sociales relacionados con el COVID-19. Siendo capaz de analizar las tendencias de desinformación encontradas en las mismas y determinar qué información es más probable que sea falsa.

La implementación de un sistema de detección de noticias falsas podría desempeñar un papel crucial en la lucha contra la desinformación en el contexto del COVID-19. Al proporcionar una forma eficiente y precisa de identificar y alertar sobre noticias falsas, se puede ayudar a minimizar su impacto negativo en la sociedad. Además de su utilidad en la detección de noticias falsas, el sistema desarrollado podría tener un impacto significativo en la promoción de la alfabetización mediática y la educación en el uso responsable de la información. Al analizar y categorizar la veracidad de las noticias o publicaciones en redes sociales, los usuarios podrían tener una mayor conciencia de las tácticas y estrategias utilizadas para difundir información falsa. Esto permitiría a las personas tomar decisiones más informadas y críticas al consumir contenido relacionado con el COVID-19.

Palabras clave: COVID-19, desinformacion, Inteligencia Artificial, Aprendizaje Automático, Procesamiento Natural del Lenguaje, noticas falsas.

Abstract

Since the appearance of COVID in 2019, not only this virus has spread rapidly, but also a lot of misinformation has been transmitted worldwide by news or internet users. This fake news has caused not only a lack of information, but also disruption in society and even deadly consequences in health problems. Therefore, being able to understand and mitigate such misinformation about COVID-19 is not only of academic interest, but also of enormous social impact.

In the early years of the COVID-19 spread, most of the society did not really understand the virus and how to act against it. However, over the years, information and data has increased, broadening experts' knowledge about the virus. Even so, many media or Internet users continue to disseminate false information. Behind these actions, strategies are articulated on numerous occasions to manipulate public opinion and erode the stability of States and their institutions.

The main objective of this project, within the field of Artificial Intelligence, is to use Natural Language Processing (NLP) and Machine Learning (ML) techniques to generate a system for the automatic detection of fake news in relation to COVID. The system developed will be used to scan all the information found on the different media sources, such as news, articles or social media posts related to COVID-19. Being able to analyse misinformation trends found in these and determine what information is most likely to be false.

The implementation of a fake news detection system could play a crucial role in the fight against misinformation in the context of COVID-19. By providing an efficient and accurate way of identifying and alerting about fake news, it can help minimise its negative impact on society. In addition to its usefulness in detecting fake news, the developed system could have a significant impact on promoting media literacy and education in the responsible use of information. By analysing and categorising the veracity of news or social media posts, users could be made more aware of the tactics and strategies used to spread false information. This would enable people to make more informed and critical decisions when consuming COVID-19 related content.

Keywords: COVID19, misinformation, Artificial Intelligence, Machine Learning, NLP, fake news.

Agradecimientos

Me gustaria dar las gracias a varias personas que me han ayudado durante los años que he cursado mis estudios en la ETSIT.

A mi tutor, Óscar Araque, por guiarme con mi Trabajo de Final de Grado, dandome un tema que ha sido muy interesante y que me ha permitido descubrir un mundo que no conocia, y por prestarme su apoyo y ayuda en aquellos momentos que he estado más atascado con este trabajo.

A mis compañeros y amigos de la universidad, que durante los cinco años que he estado cursando esta carrera me han ayudado y apoyado a superarla, y a pesar de lo que hemos sufrido, lo hemos conseguido.

A mis amigos del colegio y a mi novia Bárbara, que me han estado apoyando durante mi etapa universitaria, y que en aquellos momentos dificiles han estado a mi lado para ayudarme.

A mis padres, mi hermana y mi familia, que me han dado todas las facilidades del mundo para cursar unos estudios, que en aquellos momentos de bajon, cansancio y de no poder más, han sido los que siempre han estado ahi para impulsarme de nuevo. Tambien agradecer a mi primo Pablo Sainz por todos los consejos y toda la ayuda que me ha proporcionado para superar la carrera.

Y por último a mi, agradecerme a mi mismo todo el trabajo que he puesto durante todos estos años, por no rendirme nunca y por sacar siempre lo mejor de mi para poder lograr mis objetivos.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Context

The coronavirus (COVID-19) first appeared was in the city of Wuhan, China, in late 2019. This virus has had a serious impact on the world, as in a short time it has become a global pandemic affecting millions of people on all continents. As the number of cases increased, information about the virus began to spread through the internet and social media, both true and false. As the number of cases increased, information about the virus began to spread through the internet and social media, both true and false.

COVID-19 has had a significant influence on people's mental and physical health negatively. In terms of mental health, social isolation, uncertainty about the future and fear of illness have led to an increase in anxiety, depression and post-traumatic stress disorder. In addition, the pandemic has exacerbated inequalities in access to health care and mental health services, particularly affecting the most vulnerable communities. In terms of physical health, COVID-19 can have serious consequences, including hospitalisation and death.

Moreover, the pandemic has led to the disruption of health care services and a reduction in physical activity, which has increased the risk of chronic diseases such as diabetes, obesity and cardiovascular disease.

In the age of digital information and social media, the way news are consumed has changed dramatically. The proliferation of social platforms has given rise to a plethora of user-generated

content, making news distribute faster and more globalised than ever before. News spreads through social networks, messaging apps and search engines, and can reach millions of people in a matter of seconds. Today, any user with access to the internet can publish and diffuse information using any of these platforms. This has provided society with an almost immediate speed to keep abreast of all events and information concerning the coronavirus.

However, along with truthful and useful information, there has also been a lot of misleading information , conspiracy theories and fake news related to COVID-19. Many of these stories have been spread by people with political or economic interests, while others have arisen simply out of ignorance or misunderstanding. News verification has become increasingly important to prevent the spread of this and protect the integrity of information.

The lack of information about COVID-19 can have serious consequences, as it can lead people to make wrong or dangerous decisions regarding their health and the health of others. For example, some people have taken extreme measures, such as drinking bleach or taking dangerous medications, based on false information about how to prevent or cure the virus.

In contrast to fake news, the spread of real news about COVID-19 has also been widespread and rapid thanks to digital technology. News channels, government websites and official social media accounts have been a key source of accurate and up-to-date information. Additionally, media and news verification organisations, scientists as well as public health officials collaborating together have played an important role in turning the tables around and sharing accurate and real news.

While challenges remain in the fight against it, the effective use of digital technologies can be a valuable tool in disseminating accurate and reliable information about the pandemic.

The most common ways of communicating news about the coronavirus have been news websites and the use of social networks such as Twitter, Facebook or Instagram. In many of these sites, mainly social networks, the information that is transmitted is not verified, therefore, during this project we will analyse both the news and the publications on social networks to be able to check their veracity, all thanks to the use of Machine Learning and Natural Language Processing techniques.

1.2 Project goals

The objectives of this project will be the following:

1. Search and obtain different sources of information in order to implement text pre-processing techniques.
2. Research and study of different classification models for their development in the searched texts.

3. Upgrade the previously used models to more current and complex techniques, such as the Transformers.

4. Evaluation and analysis of the results obtained during the course of the project.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is as follows:

Chapter 1 explains the context of the work to be carried out together with the objectives to be achieved. It will give an overview of how the project is structured and its information.

Chapter 2 describe the technologies, environments and libraries that have been used to carry out this project.

Chapter 3 explains in depth the technologies, methodologies, classifiers and transformers applied to achieve the objectives of the work.

Chapter 4 shows the process that has been following during the project, such as the analysis of the datasets, the preprocessing, the optimisation of hyperparameters and the results obtained.

Chapter 5 shows the final conclusions of the project, the objectives achieved and the future work.

CHAPTER 2

Enabling Technologies

2.1 Introduction

This chapter will describe the technologies, environments and libraries that have been used to carry out this project. The project focuses on the use of Machine Learning techniques, for which the libraries that will be used to carry out the analysis, pre-processing and evaluation will be described in the following sections. The tools and platforms that have been used during the work will also be included.

2.2 Machine Learning

Machine Learning [5] is an area in the field of artificial intelligence that allows computers and systems to apply statistical learning techniques in order to automatically identify patterns in data, developing learning about processes and techniques in a similar way to how humans do it. Learning in this context means identifying complex patterns in millions of data, it is an algorithm that reviews the data and is able to predict future behaviour. Automatically, also in this context, means that these systems improve autonomously over time, without human intervention.

Machine Learning [6] appeared in the mid-1980s with the application of neural networks and decision trees. It began to be used in complex prediction problems where classical statistical

models were not efficient, such as, for example, voice and image recognition, non-linear time series prediction, financial market prediction, written text recognition, etc.

Nowadays, machine learning models have become a technological resource implemented in everyday tools such as anti-spam filters for emails, automatic car driving or voice recognition software. The main characteristic of this type of algorithms is that they are able to automatically readjust themselves to improve their performance depending on the number of hits and misses produced in a training process prior to its application and during its execution in real time.

Following, different categories of machine learning approaches are presented [7]:

1. **Supervised Learning:** the model uses a training dataset that is made up of examples labelled with the responses corresponding to those data. Its objective is to learn this data, so that when new data is introduced to it that the model has not yet seen, it can make accurate predictions thanks to the previous training it has undergone. In order to carry out this work, we will implement supervised learning.

   There are two types of supervised learning, depending on the label of the data:

   - **Classification models:** these are problems that need to predict a discrete label, within a set of possible labels, as a function of a set of input variables. There are two types of classification problems: binary and multiclass.
   - **Regression models:** these are problems that try to predict a continuous numerical response as a function of a set of input variables.

   This type of learning is incorporated [8] in technological applications such as spam detectors in e-mails, image detectors in captchas or in voice or handwriting recognition applications.

   The best known supervised learning algorithms are [1] : Linear Regression, Logistical Regression, Random Forest, Gradient Boosted Trees, Suppor Vector Machines (SVM), Decision Trees, Naive Bayes and K-Nearest Neighbor. We will talk about the first three later, as they have been chosen to develop the work.

2. **Unsupervised learning:** in this case, the machine will act on its own and will not need labels, but it will need a set of data to give results. The aim is to obtain key or important information without prior knowledge of the reference of the output variables by exploring the structure of the unlabelled data.

   So, the machine must try to integrate or group all the results in labels by relation. In the meantime, the algorithm must find a way to perform measurements and the relationships between all the results found.

   Unsupervised learning algorithms can be divided into two main groups according to their inner workings:

- **Clustering:** are iterative algorithms that use exploratory techniques to analyse the data in which the information is organised into groups without knowing the structure of the data beforehand. They start with an initial allocation of the data into groups and are modified according to an optimisation strategy.

  This is done in order to obtain groups of data with similar characteristics.

- **Dimensional reduction:** are used with highly complex data that require more processing capacity and allows finding relationships between variables in a large dataset. It works by determining correlations between the characteristics that are present in the data sets, reducing redundancies of information and reducing analysis time in order to obtain more efficiently the information considered to be of greater value.

This type of learning is used [8] for anomaly detection, recommender systems in different web applications or eCommerce sites or to identify plagiarism and copyright.

The most common unsupervised learning algorithms are [1]: K-means Clustering, t-SNE, Principal Component Analysis (PCA) and SVD.

3. **Semi-supervised learning:** is a practice that lies somewhere between supervised and unsupervised learning, combining elements of both. It is used for large datasets so that only a small group of the data is labelled and the majority are unlabelled datasets, as it increases costs, but is useful for meeting objectives.

   Although there is monitoring of how the machine performs, it is not a job that is done throughout the work with the machine. While some results have to be labelled manually, others will be proposed automatically by the machine.

4. **Reinforcement learning:** aims to build models that increase performance based on the outcome or reward that is generated by each interaction performed. This reward is the product of a correct action or set of returned data that falls within a specified measure.

   The model through an agent uses the reward as a parameter to adjust its behaviour for future actions, so that the new action also fulfils the objective or correct action and thus obtains a maximum reward.

   This type of learning is used for [8] video games, robotics, resource management or text mining. A clear example is Aplha Zero [1], an AI trained with this type of learning to play chess.

Figure 2.1 shows a schematic of the three main types of machine learning:

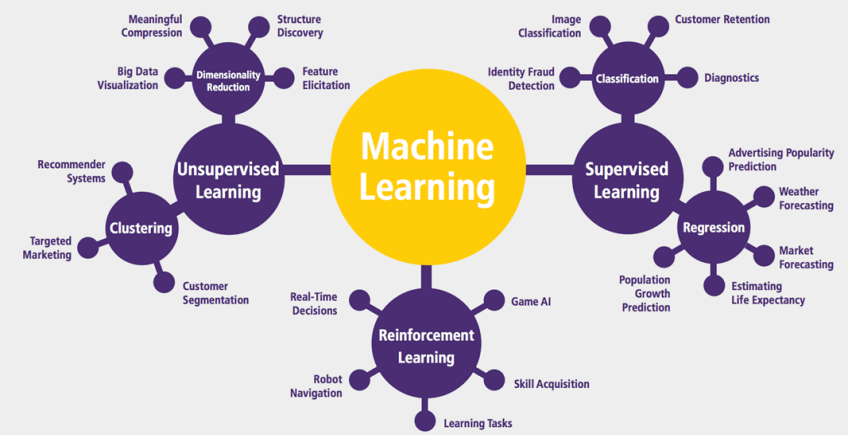

Figure 2.1: Types of Machine Learning [1]

After explaining the different types of machine learning that exist, we are going to explain two aspects that are relevant to take into account when considering the generalization capabilities of the generated prediction models. The problems we consider are under-fitting and over-fitting [9], which can affect the quality of trained models.

- *Underfitting:* occurs when a model is too simple and does not properly fit the training data, which affects the accuracy of the data that is presented. This phenomenon occurs when our model is not able to identify patterns, so it will always have poor results. Some solutions to avoid underfitting are [10]: treat data correctly eliminating outliers and unnecessary variables, use more complex models, adjust the parameters of our models or increase iterations in iterative algorithms.

- *Overfitting:* occurs when a model fits the training data too closely, in turn, tries to memorise all the data patterns. It is in this way that a predictive algorithm will result in a low accuracy on the result test, when producing forecasts with high variance. Some approaches to reduce the over-fitting challenge are [10]: split our data into training, validation and testing, obtain a greater number of data, adjust the parameters of our models, use simpler models or lower the number of iterations in iterative algorithms.

In order to show the difference in a more visual way, the following figure 2.2 compares the problems explained previously, as well as how the model performance should actually be.

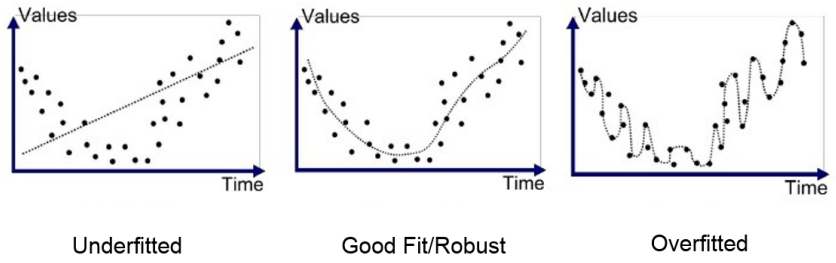

Figure 2.2: Graphical differences between overfitting, underfitting and a robust model

2.3 NLP

Natural Language Processing (NLP) [11] is the area of study focused on how computers understand human language, which focuses mainly on the understanding, handling and generation of natural language by machines. It is a complex field in which different disciplines come into play, such as Artificial Intelligence (AI), big data and linguistics.

NLP [12] is the field of designing methods and algorithms that take as input or produce as output unstructured data in natural language. Human language is highly ambiguous and variable, and at the same time constantly changing and evolving. People are very good at producing and understanding language, and are capable of expressing, perceiving and interpreting elaborate and nuanced meanings.

At the same time, although humans are great users of language, we are also very poor at understanding and formally describing the rules that govern language. Understanding and producing language with computers is therefore a challenge. In fact, the best known methods for dealing with linguistic data are supervised machine learning algorithms, which attempt to deduce patterns of usage and regularities from a set of previously annotated input-output pairs. previously annotated.

Machine learning methods excel in problem domains where it is very difficult to define a good set of rules, but the annotation of the expected outcome for a given input is relatively straightforward. In addition to the difficulties of dealing with ambiguous and variable inputs in a system with an ill-defined and unspecified rule set, natural language has an additional set of properties that make it even more difficult for computational methods, including machine learning: it is discrete, compositional and sparse. The basic elements of written language are characters, which form words that can denote objects, concepts, events, actions and ideas.

Language is also compositional: letters form words, and words form phrases and sentences.

The meaning of a sentence can be greater than the meaning of the individual words that make it up, and follows a series of intricate rules. To interpret a text, one has to go beyond the level of letters and words. of letters and words, and examine long sequences of words, such as sentences, or even entire documents. The combination of these properties leads to data sparsity. The way in which words (discrete symbols) can be combined to form meanings is virtually infinite.

To gain further insights into Natural Language Processing, let's begin with its origin and how it has evolved over the decades up to the present day [13]. NLP started around the 1960s with the first computer analyses of text. A milestone for NLP and linguistics was the publication of Computational Analysis of Present-Day American English in 1967 by authors Henry Kucera and W. Nelson Francis. In the publication they explain how they computationally realised the first NLP corpus, the Brown Corpus of Standard American English. This corpus contained approximately one million words.

The 2000s exploit to the full what has been achieved in the lasts decades. This emerges in the google search system, the robustness of Google Translate or Microsoft applications such as Word, implementing with great precision not only spelling errors but also grammar. They also create CNTK and MSRLM, two toolkits for NLP. The latter already incorporates 40 billion words. Recall the leap from 1967 of 1 million to 2007 of 40 billion.

The last decade was a true technological revolution led by Artificial Intelligence and data analysis. Finally, in 2018 came one of the great innovations of the decade: BERT (Bidirectional Encoder Representation of Transformers).

Nowadays there are many applications for NLP, but in this image we are going to mention some of them in a schematic way:

We are now going to mention and talk about some of the most relevant NLP techniques [11]. One of the best known is tokenisation, which consists of dividing the text into small texts called "tokens", taking into account criteria such as capitalisation, similar words, punctuation, etc. In addition we also have part-of-speech (PoS) tagging, which classifies the words of a sentence according to their grammatical category, surface parsing/chunks which is similar to parsing in terms of classification and representation, the bag-of-words technique which represents documents without taking into account word order or structure, based on frequency, and finally, pragmatic analysis which involves the detection of sarcasm, sentence intentionality, double readings, etc.

Another two very interesting techniques that will be applied throughout this work are normalisation and stopwords. The first one standardises words by converting them into upper or lower case and the second technique consists of omitting or eliminating redundant words, such as articles, which do not contribute much to the comprehension of the text. In later chapters, we will explain how we have implemented these two techniques and how they help us to pre-process our text before implementing the algorithms.

Figure 2.3: NLP Applications [2]

2.4 Python

In this subsection we are going to talk about some of the most popular python libraries that will help us to implement different techniques for organising data, preprocessing our text and applying different classification models. These libraries are Pandas, GSItk and Scikit-learn.

Pandas [14] is a fast, powerful, flexible and easy-to-use open source data manipulation and analysis tool, built on the Python programming language. Development of Pandas began at AQR Capital Management in 2008 by Wes McKinney. In late 2009, Pandas became an open source library, used by many data analysts around the world, who have contributed to its development and evolution to the open source.

Pandas provides efficient data structures for working with tabular data, such as worksheets and relational databases, and can handle data in different formats, including CSV, Excel, SQL and JSON. The library is based on two main data structures:

- **Series:** is a one-dimensional data structure that can contain any type of data such as numbers, strings, Boolean values, etc.
- **Dataframe:** is a labelled and mutable two-dimensional data structure commonly used for data analysis, it can be likened to a table with labelled rows and columns. Each column of the DataFrame is a Series and the rows are identified by an index.

Apart from these two structures that will be useful for structuring datasets, pandas has

numerous functionalities for manipulating and transforming data, like aggregation, merging, selecting and filtering data, and cleaning and transforming missing or incomplete data.

GSITK [15] is a library or framework that is capable of performing a wide variety of sentiment analysis tasks. The framework is aimed at both researchers and professionals, facilitating the replication of previous sentiment models, as well as offering implementations of common tasks. This is accomplished by constructing multiple abstractions on popular libraries like scikit-learn and NLTK. In this way, GSITK allows users to implement complex sentiment pipelines using understandable Python code. Numpy and Pandas and related libraries to make it easy to develop. Some of its features are:

- Dataset Manager interface
- Pre-processing (simple, normalize and pre-process Twitter), Stop word removal and Embeddings trick
- Feature extraction (Word2VecFeatures, Doc2VecFeatures, SIMON and SSWE)
- Classifiers (Lexicon and Vader)
- Evaluation (Basic or Advanced)

Scikit-learn [16] is an open source Python library that offers several simple and efficient tools for users to perform data analysis or data prediction. It is accessible to everyone and is built on top of other Python libraries such as Numpy, SciPy and matplotlib. This library offers the following tools for machine learning tasks:

- **Classification:** Identifying which category an object belongs to
- **Regression:** Predicting a continuous-valued attribute associated with an object
- **Clustering:** Automatic grouping of similar objects into sets
- **Dimensionality reduction:** Reducing the number of random variables to consider
- **Model selection:** Comparing, validating and choosing parameters and models
- **Preprocessing:** Feature extraction and normalization

This is one of the most commonly used libraries in machine learning projects because of the many options and tools it offers. For this project some of the tools that are going to be used to carry out the objectives are the following: Pipeline, Model selection such as GridSearchCV and Metrics, Classification models such as Random Forest, LinearSVC and LogisticRegression.

The functioning of all these tools and how they have been implemented throughout the project will be described in detail in the following chapters.

2.5 HappyTransformer

Happy Transformer [17] is a package built on top of Hugging Face's transformer library, which is a pre-trained deep learning neural network that has been trained in large amounts of natural language data. The library makes use of this architecture to provide a variety of pre-trained NLP models, such as BERT, GPT-2, RoBERTa, DistilBERT, among others. Transformers were developed to solve the problem of sequence transduction, or neural machine translation. That means any task that transforms an input sequence to an output sequence. This includes speech recognition, text-to-speech transformation, etc..These are some of the features that we can use with Happy Transformer:

a) **Text Generation:** automatically generating coherent and relevant text from a given input or context

b) **Text Classification:** assigning one or more tags to a text to categorise it into different predefined classes

c) **Question Answering:** respond to questions formulated in natural language from a set of text or information

d) **Word Prediction:** predicting the next word or words in a sequence of text

e) **Text-to-Text:** transforming one type of text into another type of text, such as machine translation or sentence rephrasing

f) **Token Classification:** assigning a label or category to each token or individual element in a sequence of text

g) **Next Sentence Prediction:** predicting whether a given sentence is next in a sequential order of sentences

For this proyect we will use the Text Classification feature that Happy Transfomers provides, this will be followed by the use of a pre-trained model such as Distilbert. This pre-trained model and the Text Classification feature will be explain in the nexts chapters.

2.6 Environments and platforms

In this section we are going to mention the environments and platforms that have been used to carry out the project, these are: Kaggle, Visual Studio Code, Google Colaboratory and Jupyter GSI.

Kaggle [18] is an online platform where users can collaborate on projects, share and discover datasets for their scientific work with over 50,000 public datasets and over 400,000 public notebooks. This platform also includes a Jupyter Notebooks interface that can be adjusted and customised depending on the needs and requirements of each user. In the same way, Kaggle allows free access to GPUs and to a large number of data and codes published by its extensive user community.

Visual Studio Code (VS Code) [19] is a tool developed by Microsoft that is used as a source code editor. It is open source and has become very popular among software developers because of the wide range of features and extensions it offers to users. At the beginning of the project it was the environment used together with Jupyter GSI for the development of the project.

Google Colaboratory, or "Google Colab" [20] for short, is a product of Google Research. It allows any user to write and execute arbitrary Python code in the browser. It is particularly suitable for machine learning, data analysis and education tasks. From a more technical point of view, Colab is a hosted notebook service from Jupyter that requires no configuration and provides access to computing resources, such as GPUs, at no additional cost.

In this project, when the classification models evolved to use transformers from the Happy-Transfomer library, the execution environment also had to be changed. This is why we switched from using VS Code to Google Colab, as this tool offers the use of GPUs that allowed us to train the transformers more quickly. The problem that arose later is that Google Colab resources are limited, i.e. it has a maximum number of hours of daily use and also for certain models such as BERT or ROBERTA the GPU memory was not able to accommodate them.

The last environment we have used in this work has been Jupyter Notebook [21] to be able to deal with the GPU memory problems and the daily limits that Google Colab had. For this, the Intelligent Systems Group (GSI) department provided us with an instance where we could run the project's notebooks. This has an NVIDIA Titan X graphics card with 12GB of memory. Thanks to this we have been able to solve both problems that had previously appeared and successfully carry out the work.

CHAPTER 3

Architecture

3.1 Introduction

In this chapter we address the technologies and methodologies implemented for the achievement of the objectives of the work. To do so, we describe the classification models, the transformers, the extraction of samples and the metrics that will be used to evaluate the results.

3.2 Feature Extraction with n-grams

Feature extraction [22] is a representation computation process by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large datasets is a large number of variables that require a lot of computing resources for processing. Feature extraction is the name given to methods that select and/or combine variables into features, thus effectively reducing the amount of data to be processed, while still accurately and completely describing the original data set.

The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

Representing texts as n-grams is a commonly used approach in natural language processing to represent and analyse text. N-grams are sequences of n elements, where the elements can be words, characters or any other text unit. Sample extraction with n-grams involves splitting the text into contiguous sequences of n elements and using these sequences as features or attributes in the model.

For this feature extraction we will use the Count Vectorizer [23] tool, provided by the scikit-learn library in Python. It serves to transform a given text into a vector based on the frequency (count) of each word that appears in the whole text. This is useful when we have several texts and want to convert each word in each text into vectors (for use in further text analysis).

CountVectorizer takes a set of text documents and performs several stages of preprocessing, then builds a vocabulary of all the unique words in the documents. From this vocabulary, it generates vector representations for each document, where each position in the vector represents the frequency of occurrence of a specific vocabulary word in the document. These vector representations are created in a matrix where each unique word is represented by a column of the matrix, and each text sample in the document is a row of the matrix. Image 3.1 shows a simple example of how it works:

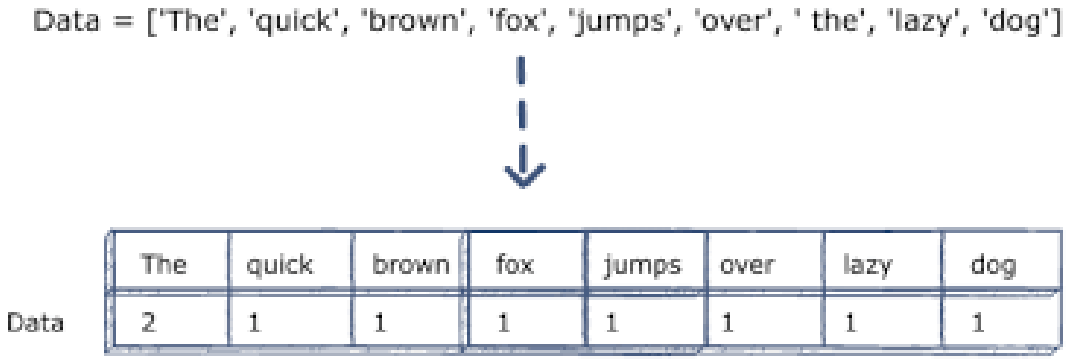

Figure 3.1: Representation of the Count Vectorizer

By configuring CountVectorizer to use n-grams, sequences of consecutive words can be captured as features. For example, if the parameter `ngram_range=(1, 2)` is set, CountVectorizer will generate a feature set that includes both unigrams (single words) and bigrams (pairs of consecutive words). For the purpose of this work, apart from the range (1,2) which allows us to have unigrams and bigrams, we will also use the range (1,3) which will allow us to have trigrams as well.

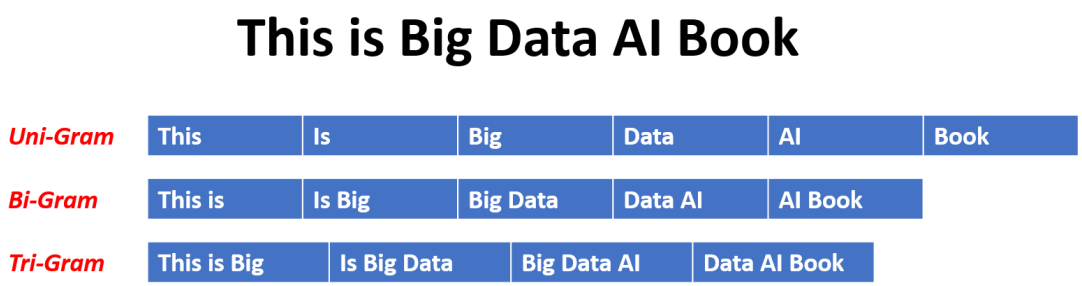

Figure 3.2: Performance of different n-grams [3]

3.3 Classification Models

We now turn to classification models, which we mentioned briefly in the previous chapter. Classification is a supervised learning approach, which is used to predict a set of discrete labels that is, one label within a finite set of possible labels, based on the input data that the model receives. The model attempts to learn the relationship between these inputs or characteristic variables and the labels or target variables. The set of labels or target variables can be of two types: binary or multiclass. This project is based on the classification of binary labels, real or fake.

Classification models in machine learning have both advantages and disadvantages compared to other types of models [24].

One of the advantages of classification models is their ability to achieve high accuracy in data classification. This makes them particularly useful in applications where a high level of reliability is required. Additionally, classification models exhibit flexibility in handling different types of data and labels, allowing them to be adaptable to various applications and problem domains. Moreover, once trained, classification models can quickly and efficiently classify input data, enabling fast decision-making processes.

However, there are also some disadvantages associated with classification models. One such disadvantage is the need for labelled data during the training phase. As a supervised learning model, classification models require data that is already labelled with corresponding categories. Acquiring and preparing such labelled datasets can be a laborious and costly process.

Another drawback is the sensitivity of classification models to changes in the distribution of data. If the underlying data distribution significantly changes, it can affect the classification accuracy of the model. This sensitivity highlights the importance of regularly monitoring and updating the model to maintain its performance.

Furthermore, classification models have limitations in terms of interpretation. Unlike some other types of models in the field of AI, classification models do not provide detailed explanations of how they arrived at their decisions. This lack of interpretability can hinder their usefulness

in applications that require transparent decision-making processes or regulatory compliance.

The most common classification algorithms used for supervised learning are: Decision trees, Logistic Regression, Support Vector Machines (SVM), Naïve Bayes classification, Least squares regression and Ensemble methods. For the development of this project we are going to use the first three mentioned above and in the following subsections we discuss them in detail.

3.3.1 Random Forest Classifier

Random Forest [25] is a widely-used algorithm in supervised machine learning, capable of handling both classification and regression problems. It leverages the power of ensemble learning, which involves combining multiple classifiers to tackle complex problems and enhance the model's performance. Random Forest comprise several decision trees, each trained on different subsets of the provided dataset. By taking the average of these trees, the algorithm improves the predictive accuracy of the dataset, offering robust and reliable predictions.

Before we continue talking about the Random Forest Classifier, let's briefly explain what decision trees are. Decision trees are a machine learning model consisting of hierarchical structures composed of nodes representing features and branches representing possible decisions or outcomes. At each node, a question is asked about a specific feature and, depending on the answer, the tree branches into different paths until it reaches the leaves, where a prediction or final decision is obtained.

Random Forest [26] has almost the same hyperparameters as a decision tree or a bagging classifier. Fortunately, it is not necessary to combine a decision tree with a bagging classifier because the Random Forest Classifier can easily be used. It adds additional randomness to the model as the trees grow. Instead of looking for the most important feature when splitting a node, it looks for the best feature among a random subset of features. This results in a wide diversity that usually translates into better model performance.

Therefore, in a Random Forest Classifier, the algorithm only considers a random subset of features to split a node. It can even make the trees more random by additionally using random thresholds for each feature instead of looking for the best possible thresholds (as a normal decision tree does). We could define it as a collection of decision trees, the difference being that the Random Forest algorithm randomly selects observations and features to build several decision trees and then averages the results.

Another difference is that "deep" decision trees can suffer from overfitting. Most of the time, the Random Forest avoids this by creating random subsets of the features and building smaller trees with these subsets, then combining the subtrees. It is important to note that this does not always work and also makes the calculation slower, depending on how many trees the Random Forest builds. Picture 3.3 bellow shows how the model works:

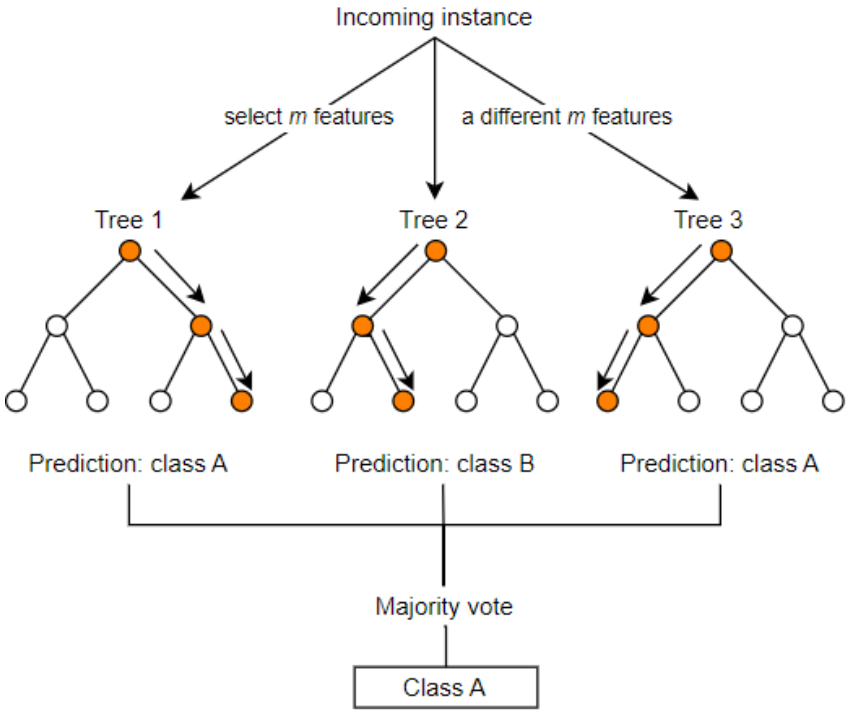

Figure 3.3: How the Random Forest Classifier works

Some of the most important parameters of a Random Forest model are the following:

- **n_estimators:** the number of decision trees in the forest. Increasing this hyperparameter generally improves the performance of the model but also increases the computational cost of training and predicting.

- **max_depth:** the maximum depth of each decision tree in the forest. Setting a higher value for max_depth can lead to overfitting while setting it too low can lead to underfitting.

- **max_features:** the maximum number of features to be taken into account when splitting each tree. Controlling this parameter can help to reduce the correlation between trees and promote model diversity.

- **min_samples_leaf:** the minimum number of samples required in a leaf of the tree.

- **min_samples_split:** the minimum number of samples required for a node to split into additional branches.

The first three parameters are the most important when designing our Random Forest classification models, the reason for the use of these three parameters will be explained when we get to the results section.

3.3.2 Linear SVC

The Linear SVC (Support Vector Classifier) is a machine learning algorithm used for data classification. It is based on the concept of support vector machines (SVM) and is specifically applied to linearly separable classification problems.

Before going into more details about Linear SVC, we are going to explain the basic concept of support vector machines (SVM) [27]. The goal of the SVM algorithm is to create the best decision line or boundary that can segregate the n-dimensional space into classes, so that we can easily place the new data point into the correct category in the future. This best decision boundary is called a hyperplane, asseen in Figure 3.4. SVM chooses the extreme points/vectors that help to create the hyperplane, these extreme cases are called support vectors.

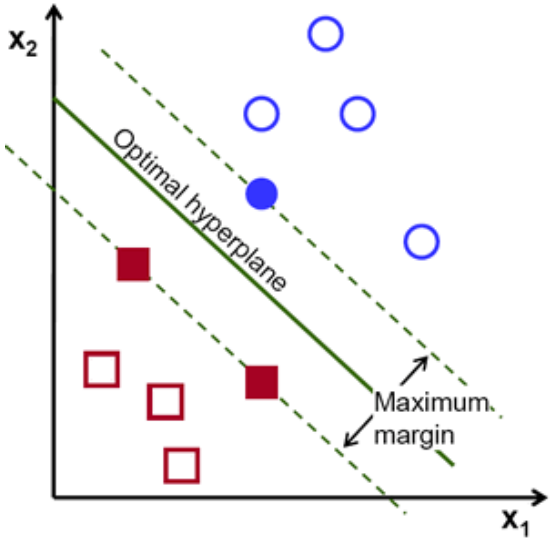

Figure 3.4: Representation of the hyperplane in the SVM

The main idea behind Linear SVC is to find a hyperplane in a high dimensional space that separates the data into different classes in an optimal way. The training process of Linear SVC involves finding the coefficients and bias of the optimal hyperplane that separates the classes, this is achieved by solving a convex optimization problem. Once the optimal hyperplane is found, it is used to classify new data points by assigning them a class label based on which side of the hyperplane they lie.

Linear SVC is used exclusively for linearly separable classification problems, whereas traditional SVMs can address both linear and nonlinear classification problems. Linear SVC is computationally more efficient by working directly in the original feature space, while traditional SVMs can be more computationally expensive due to feature transformations. Traditional SVMs offer more flexibility by allowing the use of different kernel functions, while Linear SVC is limited to linear separation of classes. Another important aspect is the difference between SVC and Linear SVC, both are similar, but the second one has the parameter kernel='linear', implemented in terms of liblinear instead of libsvm, so it has more flexibility in the choice of

penalties and loss functions and should scale better to a large number of samples.

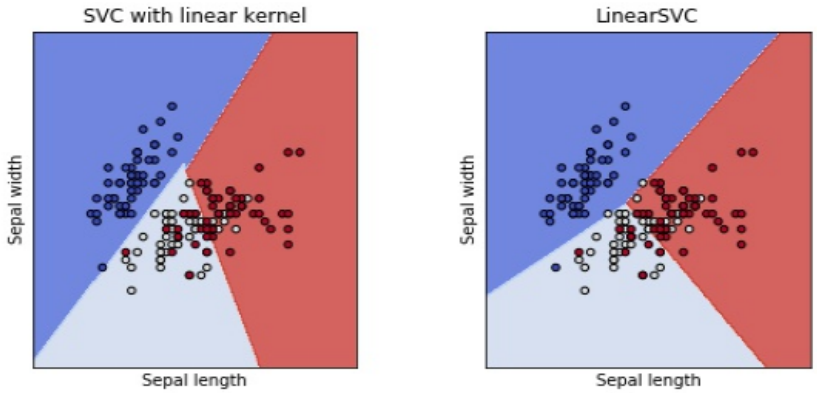

Figure 3.5: Plot different SVM classifiers in the iris dataset

Some of parameters [28] that Linear SVC has are: the regularization parameter (C), which controls the trade-off between maximizing the margin and minimizing the classification error; the error tolerance parameter (tol), which specifies the error tolerance allowed for algorithm completion; the loss function type (loss), which defines the loss function used in the optimization of the model; and the penalty parameter (penalty), which determines the type of regularization applied.

The most important of these parameters is the regularization parameter (C). A low value of C penalizes classification errors more, which may result in a model with a wider margin but with more training errors. This favors generalization ability but may lead to underfitting. On the other hand, a high value of C relaxes the error penalty, which results in a model with a narrower margin and fewer training errors, this may increase the risk of overfitting.

3.3.3 Logistic Regression

Logistic Regression [29] is a machine learning classification algorithm used to predict the probability of certain classes as a function of some dependent variables, and is mainly used for binary classification problems. Logistic Regression is used to predict the probability that an instance belongs to one of two possible classes, the model computes a sum of the input features (in most cases, there is a bias term), and computes the logistic of the result. Unlike linear regression, which is used to predict continuous numerical values, Logistic Regression focuses on predicting probabilities and performing a classification based on those probabilities.

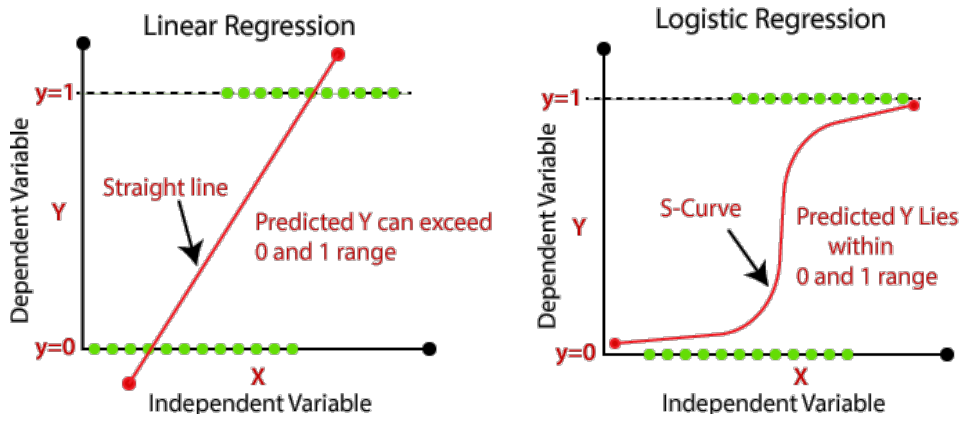

Figure 3.6: Diagram of a visual comparison between Logistic and Linear Regression

In practice, the Logistic Regression algorithm analyses relationships between variables. It assigns probabilities to discrete outcomes using the Sigmoid function 3.1, which converts numerical outcomes into a probability expression between 0 and 1. The probability is either 0 or 1, depending on whether the event occurs or not. For binary predictions, you can divide the population into two groups with a cutoff of 0.5. Everything above 0.5 is considered to belong to group A, and everything below is considered to belong to group B. The function called Sigmoid 3.1 is the following:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

In Logistic Regression, formulas are used to calculate the conditional probabilities of belonging to a specific class. For a binary classification problem with a positive class and a negative class, the probabilities of both classes can be calculated.

The formula for calculating the probability of belonging to the positive class is based on the Sigmoid function. This formula takes the form $P(y = 1|x) = \frac{1}{1+e^{-z}}$ where z is a linear combination of the model coefficients $(\beta_0, \beta_1, \beta_2, ..., \beta_n)$ and the predictor variables $(x_1, x_2, ..., x_n)$ for a specific instance. The Sigmoid function transforms this linear combination into a value between 0 and 1, representing the probability of belonging to the positive class.

The probability of belonging to the negative class can be obtained by subtracting the probability of belonging to the positive class from 1, $P(y = 0|x) = 1 - P(y = 1|x)$. These formulae allow the conditional probabilities of both classes to be calculated for a given instance and are used to perform classifications. It is important to note that Logistic Regression assumes that the classes are mutually exclusive, meaning that an instance can only belong to one of the two classes.

Logistic Regression has several key parameters [30] that influence its performance and ability to fit the model. The regularisation parameter (C) controls the strength of the regularisation to avoid overfitting and the parameter class_weight, which can be balanced/None depending

on whether you want to balance your dataset or not. The tolerance (tolerance) specifies the accuracy required for the completion of the optimisation algorithm, with a lower value indicating higher accuracy but longer training time. In addition, the regularisation type parameter (penalty) defines the type of regularisation applied to the model, with options such as "l1" for L1 regularisation (makes some β's 0) and "l2" for L2 regularisation (estimates small β's).

As in Linear SVC, the regularisation parameter (C) is the most important parameter and the one that allow us to control the balance between the fit to the training data and the generalisability of the model. A lower value of C penalises classification errors and tends to generate a model with a wider margin, which may increase generalisability but may also lead to underfitting. On the other hand, a higher value of C decreases the error penalty and may result in a model with a narrower margin, which may increase the risk of overfitting. Finding the balance in this parameter will allow us to obtain a higher performance in our model.

3.4 Transformers

Transformers [4] appeared as a novel Deep Learning architecture for NLP in a 2017 paper "Attention is all you need" that presented ingenious methods for performing language-to-language translation that outperformed the seq-2-seq LSTM networks of the time [31]. Transformers and their attention mechanism enabled the emergence of the large text-generating models GPT2, GPT3 and BERT, which could now be trained by taking advantage of the parallelism achieved through the use of GPUs. The impact that Transformers have had has made them one of the most important tools in Machine Learning.

This architecture emerged as a solution to supervised learning problems in Natural Language Processing, obtaining great advantages over the models used at the time. The transformer allowed translation from one language to another with the great advantage of being able to train the model in parallel, which made it possible to scale the speed and capacity of sequential deep learning models at unprecedented rates. This also dramatically increased speed and reduced cost, and by using the attention mechanism as an enhancer, which made it possible to track word relationships across very long sequences of text, both forward and backward.

Before Transformers, Recurrent Neural Networks (RNNs) were commonly used for natural language processing. However, RNNs still have limitations, as they processed data sequentially and were slow to train and could not take full advantage of parallel computing power. In addition, RNNs had difficulty capturing long-term relationships between words. Unlike conventional neural networks, Transformers are designed to understand sequences of data, such as text, taking into account the relationship between words. This is important because the meaning of words can change depending on their context in a sentence.

Over time, this architecture proved to be flexible and could be used for tasks beyond NLP

in addition to text generation, content classification and summarisation, it could also be applied to computer vision, audio generation, time series prediction and reinforcement learning. For all these new implementations, it is important that we explain how Transformers work[32] in order to understand them better. Figure 3.7 shows a representation of its architecture.

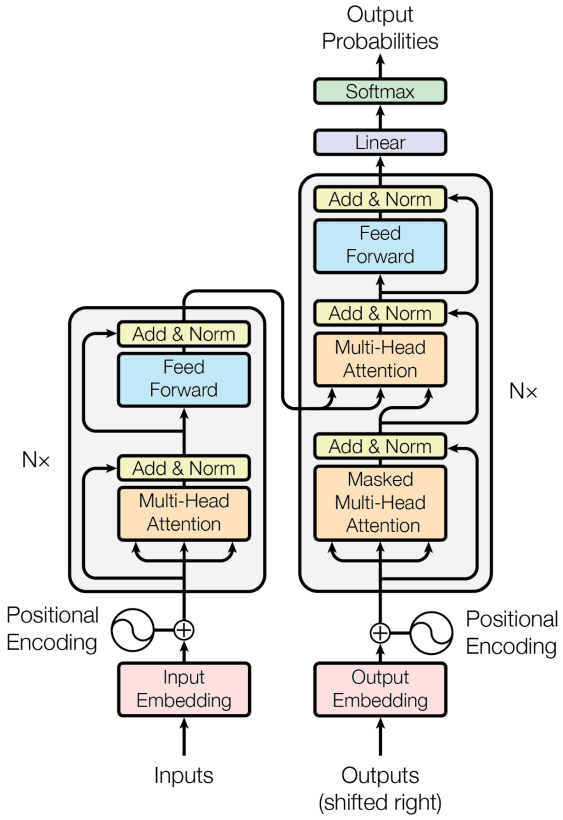

Figure 3.7: Representation of the Transformers architecture [4]

As we can see, Transformers consists of an encoding module and a decoding module. The encoding module transforms a continuous input sequence into a vector representing words and their relations. The decoding module transforms the encoded vector into a text sequence. In the original document, 6 encoders and 6 decoders are used.

The first phase is the text processing. Transformers use a pre-processing phase in which the text is tokenised into smaller units. The tokenisation algorithm may depend on the application; in most cases, each word and punctuation mark counts as approximately one token, while some suffixes and prefixes count as separate tokens. The tokens are then converted into "word embeddings". A word embedding is a vector that attempts to capture the value of words in a multidimensional space. Word embeddings are created using embedding models, which are trained separately from the transformer, and typically have hundreds of dimensions. There are several pre-trained embedding models that are used for linguistic tasks.

Once the text has been transformed into a list of embedded words, it is fed into the encoder module of the transformer. Unlike the RNN and LSTM models, the transformer does not receive

one input at a time, but as explained above, it can receive embedding values of an entire sentence and process them in parallel. To preserve the sequential nature of the words in the sentence, the transformer applies "positional encoding", which basically means that it modifies the values of each embedding vector to represent its location in the text.

The input is then passed to the first encoder block, which processes it through an "attention layer", this attention layer attempts to capture the relationships between the words in the sentence. In other words, the attention layer receives a list of word embeddings representing the values of the individual words and produces a list of vectors representing both the individual words and their relationships to each other. The attention layer contains several "attention heads", each of which can capture different types of relationships between words, so it is called multi-head attention.

The output of the attention layer is fed to a feed-forward neural network that transforms it into a vector representation and sends it to the next attention layer. Transformers contain several blocks of attention and feed-forward layers to capture gradually more complicated relationships.

The task of the decoder module is to translate the encoder's attention vector into the output data. During the training phase, the decoder has access to both the attention vector produced by the encoder and the expected output. The decoder uses the same tokenisation, word embedding and attention mechanism to process the expected output and create attention vectors. The decoder's attention vector passes through a feed-forward layer and its result is mapped into a very large vector, the size of the target data.

Finally as we can see in the figure 3.7 we have the Add & Form block. This is the "post-layer normalization", which is a technique used in Transformers to normalise the output values after each layer. Normalisation refers to the process of fitting and standardising the values of a distribution so that they have a mean close to zero and a standard deviation close to one.

This normalisation is performed by an operation called "Layer Normalization", which calculates the mean and standard deviation of the output values of a layer and uses them to normalise the values. Post-layer normalisation helps stabilise and speed up the Transformer training process by keeping the output values within an appropriate range, facilitates gradient propagation during backpropagation and prevents values from becoming too large or too small.

3.4.1 DistilBERT

The Transformer we have used for the development and evolution of the classification models is DistilBERT [33], but before talking about it in depth we have to talk about its predecessor. BERT (Bidirectional Encoder Representations from Transformers) is a language model based on the Transformers architecture that has revolutionised natural language processing. Developed by GoogleAI in 2018, BERT is known for its ability to understand the context and relationships of words in a text.

Unlike previous language models that processed words sequentially, BERT uses the power of attention and bidirectional learning to capture the context of both the words that precede and follow it. This allows BERT to have a deep understanding of the meaning of words based on their context, which significantly improves its ability for tasks such as text classification, machine translation and sentiment analysis. However, the BERT model had some drawbacks, such as being bulky and therefore a bit slow. To solve these problems, the Hugging Face researchers proposed DistilBERT, which used knowledge distillation for model compression.

DistilBERT is built upon the principles of knowledge distillation, also known as teacher-student learning, which involves compressing knowledge from a larger model into a smaller one. In this approach, the smaller model, referred to as the learner model, is trained to replicate the output distribution of a larger model or a group of models known as the teacher model. Rather than using hard targets where the model assigns high probability to one class and near-zero probabilities to others, knowledge distillation employs soft targets derived from the teacher model's estimated probabilities.

By training the learner model to imitate the teacher's output distribution, the learner can leverage the teacher's knowledge and enhance its own performance. A temperature parameter is utilized to control the smoothness of the output distribution. The training objective combines supervised training loss with distillation loss.

In the case of DistilBERT, the student model, it shares a similar architecture with BERT, the teacher model, but with specific modifications such as the removal of token embeddings and pooler, as well as a reduction in the number of layers. The DistilBERT training process incorporates the best practices employed in training BERT models.

Having understood all this, we need to look at the advantages of using DistilBERT, that makes it the ideal model to carry out this project, over BERT, these advantages are as follows:

1. **Reduced size model:** DistilBERT is a smaller model compared to BERT because it undergoes knowledge distillation, shrinking by 40%, while retaining 97% of its linguistic understanding capabilities and being 60% faster.

2. **Faster inference:** The reduction in the number of layers and the elimination of token embedding and pooling contribute to faster processing of input data.

3. **Lower memory requirements:** DistilBERT's smaller size allows it to be deployed on devices with limited memory capacity, such as mobile devices or edge devices.

4. **Cost-effectiveness:** Training and utilisation of DistilBERT requires fewer computational resources and less training time. This advantage is especially significant when working with large datasets or in scenarios where fast model iterations are necessary.

CHAPTER 4

Evaluation

4.1 Introduction

In this chapter we are going to analyse the results obtained, for two different datasets, due to the use of the different classifiers and transformers mentioned in the previous chapter. The idea is to show the process followed for text cleaning, processing, the development of pipelines for the optimisation of hyperparameters of the classification models, the metrics obtained with these and finally the evolution of the project thanks to more complex models such as the Transformers.

4.2 Datasets

In order to realise the objectives of this project, two different datasets have been selected from the Kaggle platform. These datasets include both news articles with a title and a content, and resources from different social platforms, in order to be able to analyse not only simple sentences, but also more complex text structures.

The first one, the COVID-19 Fake News Dataset [34], contains news, claims and tweets, but for the development of this project only the news were used, because the tweets and claims did not offer enough information to be analysed and were in a format that was not appropriate for the objective of the project. The second dataset used, COVID19 Fake News Dataset NLP [35], contains more than 10,000 sources of various social-media platforms such as Twitter, Facebook,

Instagram, etc. Becouse of these two datasets found in Kaggle, it is possible to perform an analysis of news and social media posts.

4.2.1 News Dataset

This dataset is made up of six different files that contain fake and real news. Therefore, we have unified it into one, adding a new column called label that indicates whether the text is real or fake. Within this news dataset, we have to analyse and see the fields it has, these are the following: *type, fact_check_url, archieve, news_url, news_url2, news_url3, news_url4, news_url5, title, newstitle, content, abstract, publish_date, meta_keywords and label.* Of all these fields, we are going to focus on those that can provide us with information for the development of the work in **title, news_url, content** and **label**.

Once we have analysed the columns, next step is to look at the number of rows we have. In total there are 4,504, of which we have to clean up as there are many that contain information that is not useful or that has not been loaded properly in the Kaggle dataset. On the other hand, it's important to view if our dataset is balanced. In the following graph you can see how many instances each type of label has:

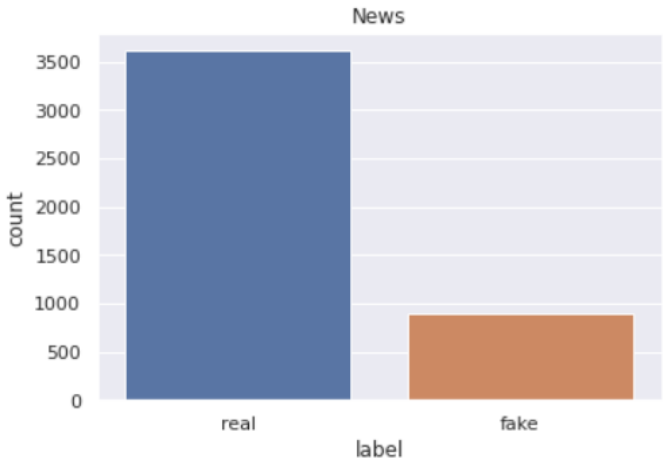

Figure 4.1: News Dataset Labels Count

As we can see, the number of instances for each class is highly unbalanced. Ideally, our dataset should be balanced, having similar distribution for each class, but in this dataset the percentage of rows for real labels is 80,15% and for the fake 19,85%. Therefore, we are going to work with this dataset in two different ways, on the one hand, we will use the title and content column together, and on the other hand, only the title column, so we can analyse the news taking into account all the information we have in this dataset.

4.2.2 Social Media Dataset

At the same time, we have the social media dataset. This dataset looks better than the previous one as it is sorted by different sets: training, validation and test. This dataset only has two columns: tweet and label (although we have also added a third column to refer to the set to which it belongs). The tweet column represents text strings from different social platforms such as Instagram, Twitter or Facebook.

This dataset also has 10,700 rows of which 60% are training, 20% validation and 20% test rows. This distribution will be maintained throughout the project in order not to alter the data. As seen in Image 4.2 the number of instances that belong to each set.

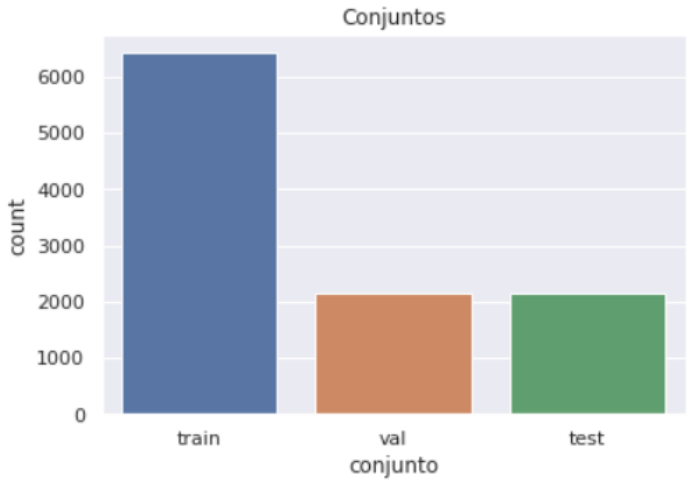

Figure 4.2: Social Media Dataset Groups

As in the news dataset, we must analyse if it is balanced or not. In this case, as shown in the graph below, this dataset is balanced for the two classes we have. It has 52.33% of rows belonging to the real class, and the rest, 47.67%, belonging to the fake class.

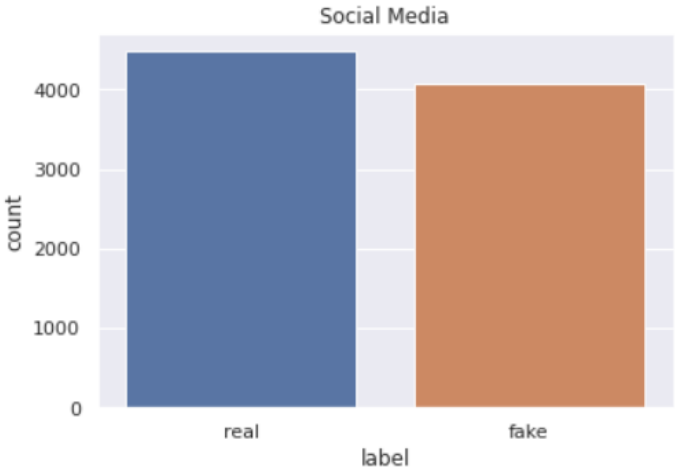

Figure 4.3: Social Media Dataset Labels Count

Although we will not have the same problems of unbalance as with the news, this dataset presents other problems that we will cover in the following sections.

4.3 Pre-processing

Once we have our datasets loaded and organised, then starts the pre-processing. Although news and social media datasets have different formats, the way to apply preprocessing is very similar. To do this we start from a general level, eliminating instances that do not have certain characteristics and then we will apply a cleaning to the text using the *Normalize* and *Stopwords* methods provided by GSItk, as exaplained in Section 2.4.

With the dataset of news, as we have mentioned previously we are going to analyse it in two different ways, that means that we would have two datasets at the same time, so the preprocessing will be done in different ways. The first thing is to search, for both datasets, through the column news_url all possible rows that contain the same url, this will help us to find those rows that are duplicated and eliminate them. At the same time, for the dataset that will be formed by the title and the content, we have to eliminate those rows whose content is null.

The next step regarding the news dataset, composed by titles and content, is to clean those rows whose content has not been loaded or does not provide relevant information. To do this we will search and delete these rows in the content and newstitle columns. Then it is time to filter by the character size of the content field. To do this it is important to count the number of characters in each line and visualise it in graphs. Images 4.5 below shows the full range of characters length and a graph whose range has been narrowed down to see at which point we should filter the content. The x-axis represents the total number of characters and the y-axis the number of rows that have that amount of characters.

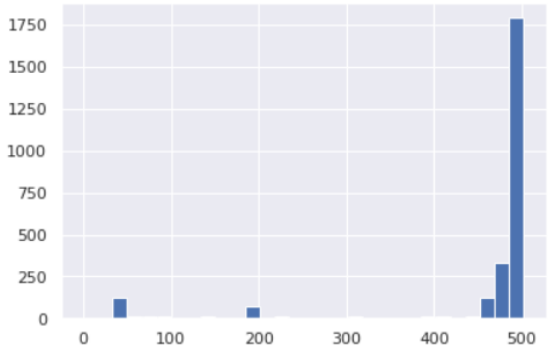

Figure 4.4: Graph with all character lengths

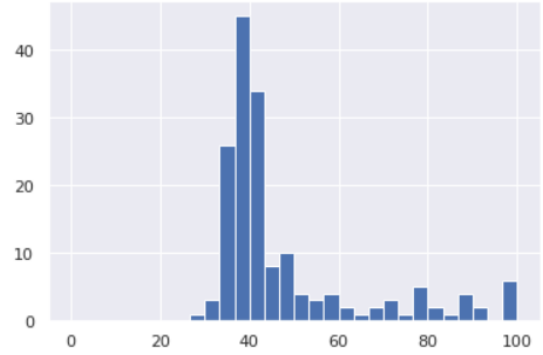

Figure 4.5: Dimensioned graph for lengths between [0,100]

Analysing these graphs and some of the rows that can be visualised, the option chosen has been to eliminate all those rows whose content is less than 70 characters.

Social Media Dataset 4.6.3, which as we have mentioned previously, has different characteristics than the news dataset, so its preprocessing will be different. Unlike the other one, this dataset does not contain empty or irrelevant instances, but it is important to modify one thing in it. In this dataset, the urls where the data come from are embedded in the same column as the text they contain, we can extract those urls and take them to another column. Once this is done, we will have our data ready for the last stage of preprocessing and cleaning.

The last step is going to be the same for the three datasets (two for the news and one for the social media), in this one we are going to apply a function that will allow us to eliminate some characters within each instance. It help us to remove whitespace, tabs, punctuation marks, emojis and some elements that mess up our data.

Once we apply this function, the last two steps we have to apply are the *Normalize* to split our text into words separated by spaces and the *Stopwords* to remove those words that have no value to perform the feature extraction. Both methods are provided by the GSItk library 2.4. Table 4.1 shows the number of values we have left for the dataset.

	Label		
	Real	Fake	Total
News titles and content	2,277	180	2,457
News with only titles	3,371	819	4,190
Social media	5,600	5,100	10,700

Table 4.1: Table of values belonging to each class for each dataset

4.4 Metrics

Before start to seen and analyze the results of the models, we have to discuss the most important metrics for evaluating machine learning algorithms. These metrics will help us to compare the results obtained with each model and to see which one best suits our needs.

The most basic metric that help us to define the others is the confusion matrix. The confusion matrix is a matrix representation of the predicted results of any binary test that is often used to describe the performance of the classification model on a set of test data whose true values are known.

For our project we are going to analyse and dig deeper into a binary classification matrix. Each prediction can be one of four results, based on how well it matches the true value:

- True Positive (TP): it is predicted as True and is True in reality.

- True Negative (TN): It is predicted to be False and is False in reality.
- False Positive (FP): predicted True and is False in reality.
- False Negative (FN): predicted False and is True in reality.

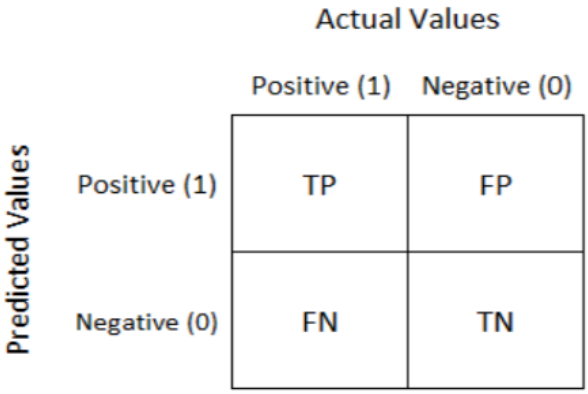

Figure 4.6: Confusion Matrix

Once we have defined these four possible predictive results, we can define various metrics that will help us to assess the quality and performance of our models. First we have the accuracy, this is a basic metric that measures the proportion of correct predictions made by the model relative to the total number of predictions.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{4.1}$$

Then we have precision and recall. The first one refers to the proportion of correct positive predictions relative to the total number of positive predictions, it is useful when the cost of false positives is high. The second measures the proportion of positive instances that are correctly identified by the model, it is useful when the cost of false negatives is high.

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4.2}$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4.3}$$

On the other hand, the F1 Score is a metric commonly used in classification problems, particularly when the data is unbalanced. This metric combines precision and recall in a single value, which provides an overall measure of the model's performance. F1 Score can be calculated as follows:

$$F1\ Score = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{4.4}$$

Having defined the F1 Score, we must understand that there are several variants that allow us to analyse our models in different ways. These will be very useful, mainly the F1 Macro, to evaluate the results of our models.

Firstly, F1 Macro calculates the F1 Score for each class and then averages these values without considering class unbalance, implying that each class has the same weight in the calculation of the average. This variant is useful when all classes are of equal importance and you want to evaluate performance equally.

$$\textit{F1 Macro} = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{4.5}$$

On the other hand, the F1 Micro calculates the overall precision and recall metrics by aggregating the true positives, false positives and false negatives across all classes, and then calculates the F1 Score using these aggregated metrics. F1 Micro gives more weight to dominant classes.

$$\textit{F1 Micro} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \tag{4.6}$$

Finally, F1 Average calculates the F1 Score for each class and then performs a weighted average of these scores, using the weight determined by the proportion of examples in each class. This balanced measure takes into account class imbalance and is useful when you want to assess the overall performance of the model by considering the distribution of classes in the data.

$$\textit{F1 Average} = \frac{1}{N}\sum_{i=1}^{N}\text{F1 Score}_i \tag{4.7}$$

The last two metrics we have are the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC). The ROC curve shows the relationship between the true positive rate and the false positive rate as the classification threshold of the model is varied. Each point on the ROC curve represents a different threshold and provides information on how the model balances the correct classification of positive samples and the misclassification of negative samples. An ideal ROC curve would be close to the upper left corner of the graph, indicating a high rate of true positives and a low rate of false positives.

The AUC, which is the area under the ROC curve, provides a quantitative measure of overall model performance. It varies between 0 and 1, where 1 indicates a perfect model and 0.5 indicates a model that classifies randomly. Ideally, a higher AUC value would indicate a better ability of the model to distinguish between positive and negative samples.

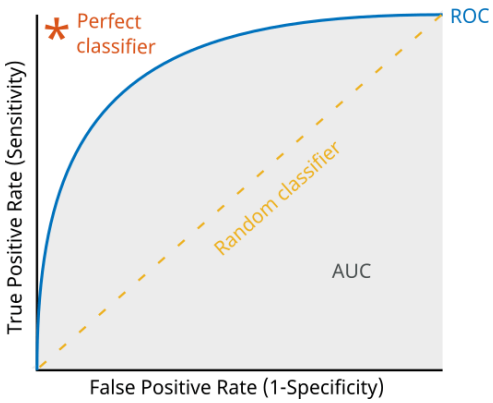

Figure 4.7: ROC and AUC Representation

All these metrics will be used in the following sections to check the performance and efficiency of our models.

4.5 Optimisation of Hyperparameters and Pipelines

Once we have our text pre-processed and cleaned, next step is extract the samples and train our models, but this is not as simple as it sounds. The problem arises when it comes to defining which parameters we are going to use for the classification models we have previously discussed such as extracting the n-grams. If we were to do this by simple trial and error, using all the possible combinations we could think of, it would be a very long and tedious process, and obviously not as effective. That is why we have to implement the optimisation of hyperparameters and pipelines.

Hyperparameter optimisation is a mechanism that allows us to automatically explore a search space for possible hyperparameters, generate a set of models and compare our models using metrics of interest. It is a fundamental process in the development of machine learning models, as these are values that are not learned directly from the dataset, but have a significant impact on the performance and generalisability of the model. These parameters control aspects such as model complexity, regularisation and the configuration of specific algorithms.

The goal of hyperparameter optimisation is to find the optimal combination of values to maximise model performance on a given data set. This process can be complex and computationally expensive, as it involves exploring a multidimensional search space and evaluating model performance for each combination of hyperparameters. To address this challenge, techniques such as grid search, random search and Bayesian optimisation are used to search more efficiently for the best combinations of hyperparameters.

On the other hand, pipelines are a structured way to organise and automate the workflow. A pipeline consists of a sequence of steps that are applied in order to the data, such as pre-

processing, feature extraction and model training. Pipelines are especially useful when working with complex datasets and a series of transformations are required before building and evaluating a model.

A key benefit of pipelines is that they facilitate reproducibility and portability of experiments. By defining and organising the workflow in a pipeline, it ensures that each step is applied consistently and that the results are comparable between different runs. In addition, pipelines allow you to make adjustments and improvements more efficiently, as you can modify and add steps without having to rewrite the entire code.

Prior to the implementation of the pipelines, we have to take into account that our data have to be divided into the train set and the test set. For the news datasets we divide the data into 80% for training, and 20% for test. It is important to set a random state with a fixed value to control the randomness and guarantee the reproducibility of the results.

On the other hand, for the social media dataset, as our data was previously split, it is not necessary to split it again. We will merge the training and validation sets into one set, which will account for 80% of the total data, and on the other hand we have the test set, which will account for 20% of the total data. At the same time, for all datasets, the annotation "X" and "y" will be used. The variable "X" represents the characteristics or attributes of the input data (text), while the variable "y" represents the labels or output values associated with the input data.

After briefly explaining how hyperparameter optimisation works, the pipelines and the division of the data into training and test sets, we follow with the pipelines implemented. As we have three models and three datasets we will need nine pipelines. Each pipeline will have specific parameters for each classifier, as the number of maximum samples for the headline news will not be the same as the social media dataset. All pipelines will use CountVectorizer with unigrams, bigrams and tigrams. In both Linear SVC and Logistic Regression, we use a logarithmic scale for the parameter C instead of a linear scale. This logarithmic scaling facilitates a more comprehensive exploration of the C parameter space, allowing us to assess the model's performance across a wide range of C values.

After defining our parameters for the CountVetorizer and the classifiers, we use GridSearchCV to find the best hyperparameters of a model. This technique divides the dataset into multiple subsets or "folds" to evaluate model performance more robustly and avoid overfitting. GridSearchCV combines grid search with cross-validation, which means that it performs an exhaustive search for hyperparameters in a predefined grid and evaluates the model performance for each combination of hyperparameters using cross-validation.

Subsequently, we fit and evaluate different hyperparameter combinations of the classifier using cross-validation. Finally, the pipeline when finished will return the best selected parameters and the score obtained. Also in this pipelines we have implemented the use of dictionaries to be

able to store the obtained parameters and not have to add them manually later. Below are the tables for each classifier with the results and the parameters obtained for each dataset.

	Count Vectorizer		**Random Forest**		
	max_features	**ngram_range**	**n_estimators**	**max_depth**	**Score**
News titles and content	20000	(1, 2)	200	30	0.96
News with only titles	8000	(1, 3)	300	55	0.87
Social media	2000	(1, 3)	250	65	0.91

Table 4.2: Best parameters and scores for Random Forest with each dataset

	CountVectorizer		**Linear SVC**	
	max_features	**ngram_range**	**C**	**Score**
News titles and content	20000	(1, 3)	6.8273	0.96
News with only titles	6000	(1, 2)	1.0023	0.90
Social media	27500	(1, 2)	1.0023	0.93

Table 4.3: Best parameters and scores for Linear SVC with each dataset

	CountVectorizer		**Logistic Regression**	
	max_features	**ngram_range**	**C**	**Score**
News titles and content	27500	(1, 3)	1.0023	0.96
News with only titles	6000	(1, 2)	17.8186	0.91
Social media	25000	(1, 2)	2.6159	0.94

Table 4.4: Best parameters and scores for Logistic Regression with each dataset

4.6 Classification Models

Once we have obtained the best parameters for our classifiers, we start training the models, obtaining the predictions and evaluating the metrics they return. The first point for each of the models we are going to train is to create its own CountVectorizer, which contains the previously obtained parameters. After this we have to to obtain the vectors, both for the training set and

for the test set.

After this, we have to create our X's and y's for each model and for each dataset, and associate to the X's the vectors obtained after the CountVectorizer and to the y's the corresponding labels. Next, we have to create our models and give them the corresponding train set, this will train the model. Now we simply have to take out the predictions and the predictions of the probabilities and we can start evaluating the results of our models, comparing them with the true values of our datasets.

4.6.1 News title and content results

First we will start with the news dataset consisting of titles and contents. The following table 4.5 shows the results obtained for this dataset with the different classifiers. All results shown bellow in the tables are in percentage.

Model	Accuracy	Recall	Precision	F1-Score	AUC
Random Forest	7,31	7,31	93,92	2,41	67,39
Linear SVC	67,27	67,27	86,84	75,38	46,48
Logistic Regression	36,99	36,99	89,46	48,06	54,43

Table 4.5: Results for News title and content dataset

As we can see, for these data with Random Forest the model is having problems when classifying the true samples, the model is mostly biased towards the majority class (false) due to the unequal distribution of the classes, hence having an accuracy of 93% and a recall of 7%. On the other hand, the data with Logistic Regresssion and Linear SVC behave much better, the latter being the one with the highest F1-Score with 75.38%.

As we can see in the figure 4.10 below, the ROC of the Random Forest reflects that the model may have a tendency towards the dominant class at the beginning, which affects the initial position of the ROC curve. On the other hand, the models trained with Linear SVC and Logistic Regression show ROC curves and AUC very similar to those of a random model.

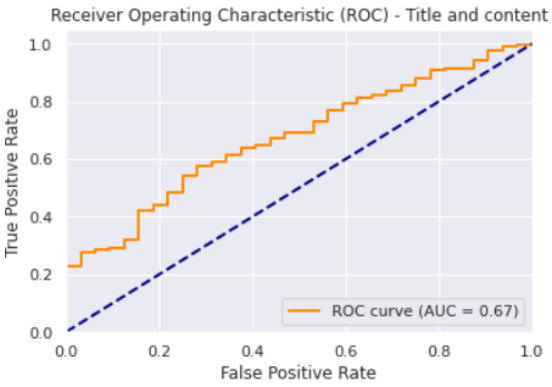

Figure 4.8: ROC for Random Forest

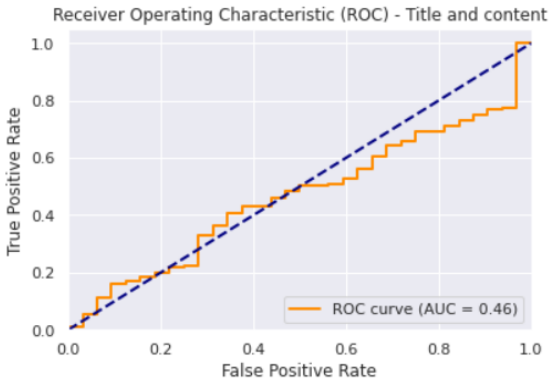

Figure 4.9: ROC for Linear SVC

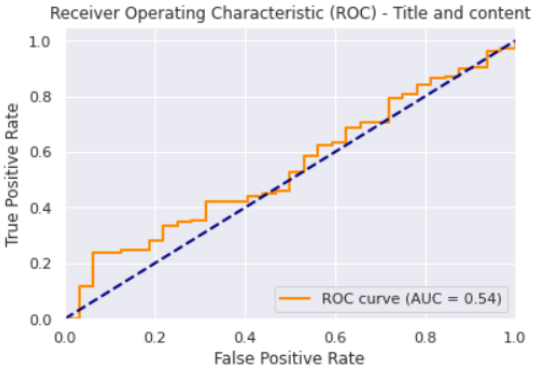

Figure 4.10: ROC for Logistic Regression

4.6.2 News with only titles results

Next we will continue with the news dataset consisting of titles only. The following table 4.6 shows the results obtained for this dataset with the different classifiers.

Modelo	Accuracy	Recall	Precision	F1-Score	AUC
Random Forest	82,93	82,93	80,81	77,37	65,49
Linear SVC	75,77	75,77	72,56	73,96	47,20
Logistic Regression	75,89	75,89	72,93	74,22	48,52

Table 4.6: Results for News only with titles dataset

For this datset we obtain much better results compared to the results obtained for the other dataset pertaining to news. In this case the Random Forest is the classifier that gives us the best results with an accuracy of 80.81% and a recall of 82.93%. The other two classifiers also provide fairly decent results, with all their metrics being above 70%.

We will now visualise the different ROC curves for each of the models. The ROC curve for Random Forest has improved quite a lot, since in this case there is no learning trend on the dominant class and its AUC is higher than 0.5, so it is far from being a random model. On the other hand, as in the previous dataset, the ROCs for Linear SVC and Logistic Regression are still quite similar and look like a random model.

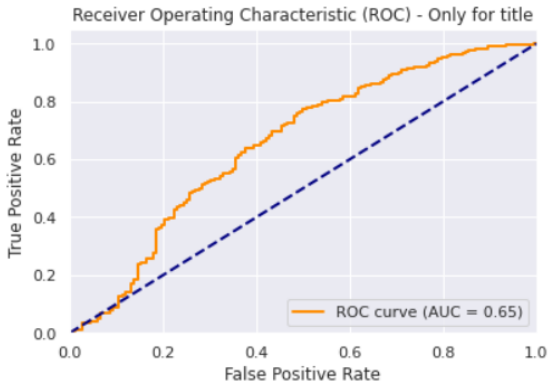

Figure 4.11: ROC for Random Forest

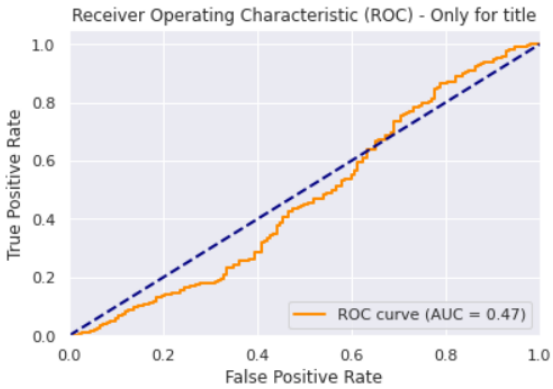

Figure 4.12: ROC for Linear SVC

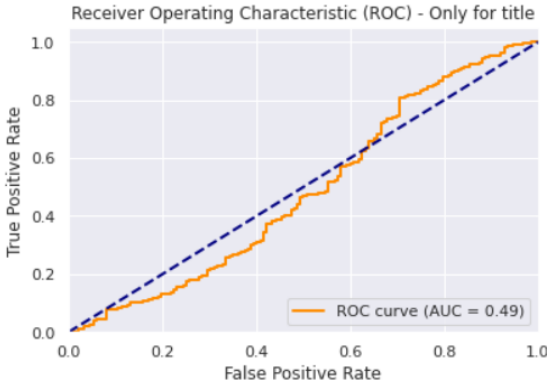

Figure 4.13: ROC for Logistic Regression

4.6.3 Social media results

Finally, we have the social media dataset. Table 4.7 shows the results obtained for this dataset with the different classifiers.

Modelo	Accuracy	Recall	Precision	F1-Score	AUC
Random Forest	65,42	65,42	74,90	62,59	81,92
LinearSVC	52,61	52,61	58,85	46,04	54,29
Logistic Regression	52,52	52,52	61,68	44,17	59,61

Table 4.7: Results for Social Media dataset

As mentioned at the beginning of this chapter, this dataset did not have the same problems as the news as it was not unbalanced. We observe that the model trained with the Random Forest is the one that obtains the best results, with an accuracy of 74.90% and a recall of 65.42%. The metrics obtained with the Linear SVC and Logistic Regression classifiers are only 50% more accurate, which means that these models are highly random when classifying the test data.

Finally, the following figures show the different ROC curves of each model. Similar to the other datasets, the curves for Linear SVC and Logistic Regression are very similar to those of a random model, being scarcely higher than the 50% that represents a random model. On the other hand, for the Random Forest classifier, its ROC curve and its AUC give us a good result for this model, as they are closer to a perfect model, which indicates that this model has been able to produce a good result for the Random Forest classifier, as they are closer to a perfect model.

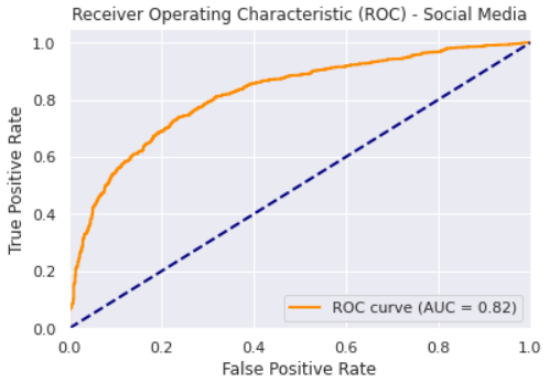

Figure 4.14: ROC for Random Forest

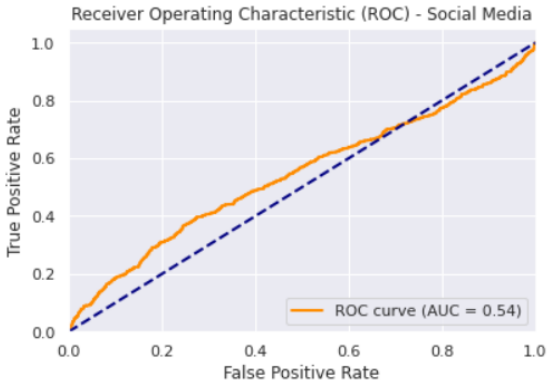

Figure 4.15: ROC for Linear SVC

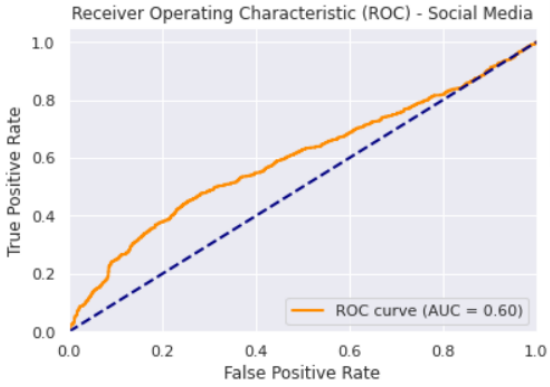

Figure 4.16: ROC for Logistic Regression

4.6.4 F1 Macro Average Results

As a last point in the evaluation of the classification models, we have to analyse and compare the different F1 Macro Average we obtain. As we mentioned before, this metric is very important because it calculates the F1 score for each class and then averages these values without taking

into account the imbalance between classes. Table 4.8 shows all the F1 Macro Averages for each of the datasets and the classification models.

	Random Forest	**LinearSVC**	**Logistic Regression**
News titles and content	7,01	43,54	31,94
News with only titles	54,37	53,35	54,02
Social media	63,03	46,86	45,12

Table 4.8: F1 Macro Score for all the datasets

In the News titles and content dataset, it is observed that the Random Forest model obtains the lowest F1 Macro Average value (7.01%). This suggests that the model has difficulties when dealing with the class unbalance present in this dataset, which may lead to a biased classification towards the dominant class. On the other hand, the LinearSVC and Logistic Regression models achieve higher F1 Macro Average values, although still relatively low.

In the News with only titles dataset, all three models show more similar F1 Macro Average values compared to the previous dataset. This suggests that these models perform more consistently when classifying news based on titles alone. The Random Forest model stands out by obtaining the highest F1 Macro Average value (54.37%), indicating that it is the most effective model for this particular dataset.

On the other hand, in the Social media dataset, the Random Forest model again obtains the highest F1 Macro Average value (63.03%). This suggests that this model is the most suitable for classifying data from social networks. However, both LinearSVC and Logistic Regression show lower F1 Macro Average values compared to the Random Forest model. This may indicate that these models may have difficulties in capturing the peculiarities and unique characteristics of social network data.

We can see that for all datasets, regardless of whether they are balanced or not, the classification models still have problems in obtaining good results. Therefore, we have to evolve our project to obtain better results, which we will achieve thanks to the Transformers and the DistilBERT model.

4.7 DistilBERT

As we have seen in the previous section, the results obtained with the classification models have not been entirely satisfactory. The classification models are basic, and do not offer any new features at present, therefore we have to use the Transformers, which are more innovative and

offer better results. Therefore we are going to implement DistilBERT, described in Section 3.4.1, in each of the datasets we had.

The first step for the three datasets, after doing the preprocessing and dividing the data into training and test sets, is to create our model, for this we are going to use the HappyTransformers library, described in Section 2.5. We start by creating an instance of the model, we have to specify the model to use, DistilBERT in this case, and the number of labels we have, which is two, real or false. Once this is done, we have to define the arguments of our model, which are: the number of epochs or training cycles to be carried out and the batch size or number of training examples to be used in each iteration during training.

During the development of the project, choosing the arguments for the DistilBERT models has been a trial and error process. Just as in the Pipeline Section 4.5, we commented that it was not entirely efficient to do this, in this case it is the other way around, since implementing a pipeline for this is not efficient. The difference in the score obtained with some arguments or others is not going to be excessively significant, since with most of the parameters the results were quite good. After defining these arguments, the next step is train the model. It is important to take into account that this library has to receive the data in csv form, so when we do the split to have the train and test sets, we must create the corresponding csv for each one.

Once the model is trained it is time to get the predictions. HappyTransformers returns predictions as "LABEL_0" and "LABEL_1", for those results that are fake and real respectively. Therefore, we must apply a few lines of code so that they correspond to 0 and 1 to be able to make the predictions. It should also be noted that for these models, the format of the labels in the datasets has been changed from real and fake to 1 and 0, although they still have the same meaning. This has been done to facilitate the work, as this is the format that this library interprets.

In addition, for social media text, we need to encode the text as a tensor, which involves converting the text into a numerical representation suitable for the model. In this case, the maximum tensor length is adjusted to address specific problems that arose during the development of the model.

The maximum tensor length refers to the maximum number of tokens or words allowed in the input text. If a text exceeds this length, it will be truncated or trimmed to meet the limit, this is important because natural language processing models, such as DistilBERT, have a limitation on the number of tokens they can efficiently process. By adjusting the maximum tensor length, one seeks to balance the performance of the model and the quality of the text representation.

Once all this is done the next step is to evaluate the performance of our new models.

4.7.1 Results of DistilBERT

Finally we have to evaluate the performance of DistilBERT. Table 4.9 shows the results obtained for each one of the datasets, these are out of 100, i.e. in percentage.

	Accuracy	**Recall**	**Precision**	**F1-Score**
News titles and content	96,95	96,95	96,83	96,88
News with only titles	94,39	94,39	94,31	94,33
Social media	90,56	90,56	90,60	90,54

Table 4.9: DistilBERT results for each dataset

As can be seen in the table, these results are much better than those obtained using the previous classifiers. First, the set of news items consisting of headlines and content shows surprisingly good metrics, as all of them are above 96%. The F1-Score value, which combines sensitivity and accuracy, was 96.88%, indicating a good balance between both metrics. We should also keep in mind that these metrics are very good knowing that our dataset is quite unbalanced, which is usually a problem as we saw in the previous section.

In the News with only titles dataset, the model also performed well with metrics above 94%, although slightly lower compared to the previous dataset. These results indicate that the model is able to perform accurate classification on this dataset, although slightly less effective compared to the News titles and content dataset, contrary to what was previously the case. This suggests that the more information we provide, the better the performance of the model.

Finally, the social media dataset shows a performance and results above 90% in all metrics. The improvement is very high compared to the classification models, where Random Forest offered the best results. This dataset is the most balanced and has the highest number of samples and we can see that DistilBERT also offers us very good results.

As the HappyTransformers library does not yet have the option to generate the predicted probabilities, which allows us to illustrate the ROC curve and the AUC, nor can we obtain graphs of the learning curve, one option to visually see the performance of this model is by means of the confusion matrix. In it we can see, for the three datasets, that the number of correctly classified samples is much higher than those that are not, and that gives us good results.

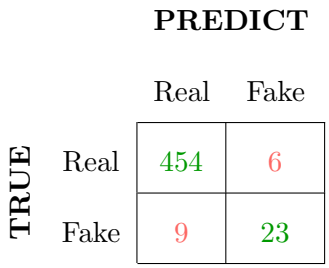

Figure 4.17: Confusion Matrix for News titles and content

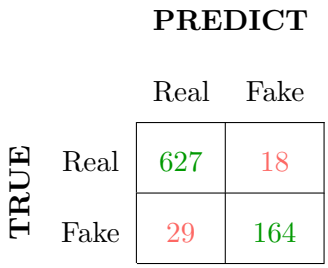

Figure 4.18: Confusion Matrix for News with only titles

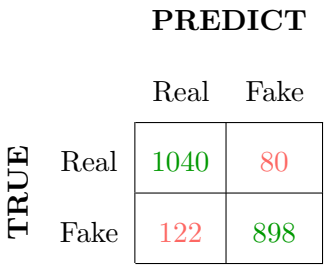

Figure 4.19: Confusion Matrix for Social Media

Lastly, Table 4.10 shows all the F1 Macro Average scores obtained. The scores obtained with the DistilBERT model are better and show the true performance of the Transformers compared to the previous classifiers. In the set of news, although in the previous table we saw that the metrics were much better for the dataset of titles and content, when we do not take into account the class imbalance, the performance is 5% better for the news with only the titles using DistilBERT. Also if we compare the F1 Macro Average for the social media set, it is superior to those previously shown, as it increases by 27.51% compared to the best model, Random Forest.

Comparing these metrics with those in the table above, it can be seen that the F1 Macro scores are lower than the F1-Score previously reported. This is because the F1 Macro metric averages the F1 scores of all classes, without considering the imbalance between classes. On the other hand, the F1-Scores above were calculated for each individual class, providing a more specific and detailed assessment of performance for each class. In summary, the results in the table indicate a reasonable average performance of the DistilBERT model in class ranking across different datasets.

	Random Forest	**Linear SVC**	**Logistic Regression**	**DistilBERT**
News titles and content	7,01	43,54	31,94	86,89
News with only titles	54,37	53,35	54,02	91,92
Social media	63,03	46,86	45,12	90,52

Table 4.10: F1 Macro Score for all the datasets

CHAPTER 5

Conclusions and future work

In this chapter we describe the final conclusions related to the work that has been carried out, an analysis of whether the initial objectives have been achieved and the future work that could be developed with this project.

5.1 Conclusions

The main objective of this project is to classify all types of internet publications or news about COVID-19, in order to detect which are false and which have true information, all by using Machine Learning and Natural Language Processing techniques.

Since the appearance of COVID-19, information about the virus has been limited, and both internet users and the media have been spreading news that is not entirely true. Despite the fact that research was carried out over time, and experts discovered how the virus worked, fake news and messages continued to be distributed online.

All this spread of false news during the years of pandemic that we have had, not only caused the population not to be well informed and not to know exactly how this virus affected humans, but it has also caused many people psychological problems and other types of disorders. The problem that arises is that any kind of news, whether true or false, is distributed over the internet very quickly, and both those who publish it and those who read it often do not check its authenticity.

With the development of this project, we advance in developing a system that classifies news and publications to determine their veracity with a high degree of accuracy. To achieve this work, two different types of datasets have been used: the first one formed by news extracted from different media and composed by a title and a content; and the second one formed by users' publications in different platforms. Moreover, for a better study and analysis of the news, this dataset has been divided in two, one formed by the combination of titles and contents, and the other one only with the titles. Using these two datasets, we can cover the main sources through which information is distributed today.

The first steps of this project, apart from obtaining the data, have been the pre-processing of each of the datasets. As each one has different formats, we have had to treat them in different ways. Subsequently, the hyperparameter optimisation stage was carried out for the basic classification models such as Random Forest, Linear SVC and Logistic Regression. This hyperparameter optimisation has been carried out using different pipelines for each of the datasets, allowing us to obtain the best parameters that fit each of our models.

After this last step, the selected models were trained, evaluated and analysed. Very different results have been obtained, from models with very low metrics, models that classified the samples randomly, to models that obtained a quite decent performance for the selected data. For this reason, it was decided to implement the use of more advanced classifiers such as the Transformers, specifically with DistilBERT.

Transformers allow us to obtain better results with the use of attention, a mechanism that allows models to relate the words in a sentence to all the other words in the sentence, as well as to model context and coherence in the text. The results obtained with DistilBERT have exceeded all my expectations, being in multiple of the metrics above 90% in each of the selected datasets. These new models trained with DistilBERT have allowed me to observe the real potential that Transformers have nowadays for Machine Learning tasks.

To conclude, on a personal level, this project has helped me to enter a world that I was totally unaware of, which has infinite possibilities today, being able to be involved in important issues such as health, finance, information and many more applications. Although at the beginning of the project it was difficult for me to understand and get into this world, today I feel that the work I have done has helped me a lot to know more about the world of artificial intelligence, generating curiosity and desire to continue investigating more.

5.2 Achieved goals

In Section 1.2 the main objectives of this project were presented, therefore the following list shows how they have been successfully completed:

- Objetive 1 has been completed as we have been able to obtain different datasets with different structures, both news and social media, and to apply preprocessing techniques.
- Objetive 2 has been achieved as we have found and studied three basic classification models, Random Forest, Linear SVC and Logistic Regression, in order to implement them on the obtained texts and evaluate their performance.
- Objective 3 has been achieved as the use of Transformers and specifically the DistilBERT model has reduced time to results and improved performance.
- Objective 4 has been accomplished as an evaluation and analysis of the results obtained has been carried out, comparing metrics between classification models and concluding with results above 90% for each of the datasets used.

5.3 Future work

This section focuses on possible improvements and extensions that can be made to the current work. In the following, some relevant tasks that could be addressed in the future are presented.

One important task would be to improve existing models. Although the results obtained have been good and are in many metrics above 90%, in the future it would be good to increase the performance and obtain better results from the models to a near perfect state, as it is about the classification of false or true news in such an important topic as COVID-19.

A possible task of this work to be done in the future could be to train models with a dataset that unifies news and social media data. This would allow us to develop a classification model capable of achieving high performance when classifying texts regardless of where they have been obtained.

In addition, it will also be helpful create different applications as a way to educate the users about COVID-19 fake news, as well as show them how to spot it. This could be, through a public online platform where users can submit information about the virus and receive a response as to whether the facts are true or false. Another implementation could be an application that sends alerts when false information has been leaked on social networks or news sites.

APPENDIX A

Impact of this project

This appendix reflects, quantitatively or qualitatively, on the possible impact on society, economics, environment and ethics.

A.1 Social impact

The development of a fake news detection system in relation to COVID-19 has a significant societal impact. The whole of society is affected, as the spread of fake news can cause confusion and disruption of informed decision-making about the virus. In addition, this system aims to improve accessibility to reliable and accurate information on COVID-19, which contributes to people's education and their ability to detect and avoid misinformation. By analysing and classifying news stories, common patterns and characteristics of misinformation can be identified, enabling users to develop critical skills to assess the veracity of information and recognise indicators of possible fake news.

In terms of security, the identification and early warning of fake news can help prevent the spread of information that is harmful or dangerous to public health and also reduce panic, uncertainty and mistrust in institutions and scientific information. Promoting well-being is also achieved by providing users with tools that allow them to filter and evaluate information more efficiently, avoiding psychological problems, such as anxiety and stress, caused by exposure to fake news.

A.2 Economic impact

From an economic perspective, the development of this fake news detection system can have a positive impact in several respects. Firstly, this system can contribute to improved productivity in various industries and organisations by providing users with a reliable tool to identify and filter out fake news, reducing the time and resources spent on verifying the veracity of information. This allows companies and organisations to focus on more relevant tasks and make more informed decisions based on reliable data, and by preventing the spread of false information, it reduces the risks and costs associated with making decisions based on misleading information.

In addition, the development of this system can generate business and employment opportunities in the field of artificial intelligence and natural language processing. The creation, implementation and maintenance of this technology requires the involvement of algorithm development experts, data scientists, software engineers and related professionals.

A.3 Environmental impact

There are positive aspects of the development of this system in terms of environmental impact related to sustainability and the consumption of natural resources. This project can contribute to the reduction of energy consumption, as by detecting fake news, the need to access multiple sources of information is reduced, which in turn reduces the demand for energy used in data transmission, processing and storage. By optimising the efficiency of information classification and analysis processes, the load on servers and IT systems is minimised, which can result in reduced energy consumption in the long term.

In addition, by promoting awareness of misinformation and encouraging fact-checking, it contributes to reducing the spread of misleading content that can generate confusion and lead to inefficient use of resources. By preventing people from relying on false information, it reduces unnecessary or inappropriate actions that could have a negative impact on the environment. For example, the spread of false news about possible cures or treatments for COVID-19 can lead to unnecessary consumption of products or medicines that are neither effective nor safe.

A.4 Ethical impact

Regarding ethical and professional responsibility, it is essential that the development and implementation of this project complies with applicable professional and legal rules and regulations. This involves the proper management and control of risks associated with the detection of fake news and the processing of sensitive health-related data. It is also necessary to respect intellectual property rights and comply with data protection legislation when handling users' personal

information. In addition, the development of this system is expected to be guided by engineering codes of ethics, ensuring transparency, impartiality and fairness in its operation. Ethics also involves considering the impact of these technologies on people's well-being. It is essential to ensure the security and privacy of users when collecting, storing and processing data related to fake news detection, involving the implementation of appropriate security measures and guaranteeing the anonymity and confidentiality of personal information.

APPENDIX B

Economic budget

This appendix details an adequate budget to bring about the project.

B.1 Physical resources

The physical equipment used for the development and implementation of this project was a computer equipped with an Intel Core i7-10510U 1.80GHz x 4 CPU processor, 16GB of RAM, 512GB of SSD storage and an NVIDIA GeForce GTX 1050 graphics card. The approximate price of this computer is 1130 euros.

B.2 Human resources

This project has been carried out on an individual basis, so only the salary needed for one person is considered.

The time spent was the equivalent of 12 FTEs, which means 360 hours of work during the project.

Considering the number of hours, which has been carried out by one person, taking into account that the average salary for a part-time trainee is 500 euros and that the monthly dedication is 4 hours per day for 22 working days, the total cost is approximately 2045 euros.

B.3 Software licenses

All software licenses for the programs that have been used for this project are open source, which means that there is no additional cost to the budget.

B.4 Total budget

Taking into account physical and human costs, the total budget of this project is approximately **3175 euros**.

Bibliography

[1] F. Gonzalez. Machine Learning: La base que tenes que tener. https://somospnt.com/blog/53-introduccion-a-machine-learning.

[2] A. Saleem. Applications of Natural Language Processing. Data Science Dojo. https://datasciencedojo.com/blog/natural-language-processing-applications/, September 2022.

[3] Arvindpdmn. N-Gram Model. Devopedia. https://devopedia.org/n-gram-model, 2023.

[4] ¿Cómo funcionan los Transformers? en Español. Aprende Machine Learning. https://www.aprendemachinelearning.com/como-funcionan-los-transformers-espanol-nlp-gpt-bert, 2022.

[5] GraphEverywhere. Machine Learning. Qué es, tipos, ejemplos y cómo implementarlo. https://www.grapheverywhere.com/machine-learning-que-es-tipos-ejemplos-y-como-implementarlo/, January 2022.

[6] P. P. Torralba. ¿Qué es Machine Learning (aprendizaje automático)? Thinking for Innovation. https://www.iebschool.com/blog/que-machine-learning-big-data/, March 2023.

[7] KeepCoding Bootcamps. 4 tipos de aprendizaje automático. https://keepcoding.io/blog/tipos-de-aprendizaje-automatico, April 2023.

[8] KeepCoding Bootcamps. 12 aplicaciones de los tipos de aprendizaje automático. https://keepcoding.io/blog/aplicaciones-de-los-tipos-de-aprendizaje-automatico/, April 2023.

[9] KeepCoding Bootcamps. Diferencias: underfitting vs overfitting. https://keepcoding.io/blog/underfitting-vs-overfitting/, April 2023.

[10] Rubiales. Alberto. ¿Qué es Underfitting y Overfitting?. https://rubialesalberto.medium.com, December 2021.

[11] V. Unir. ¿Qué es el NLP y para qué sirve? UNIR. https://www.unir.net/marketing-comunicacion/revista/nlp-procesamiento-lenguage-natural/, October 2021.

[12] Y Goldberg. Neural network methods for natural language processing. synthesis lectures on human language technologies. https://sci-hub.se/10.2200/s00762ed1v01y201703hlt037, 2017.

[13] F. Murzone. ¿Qué es el Natural Language Processing o NLP? De dónde viene y hacia donde vamos, October 2020.

[14] Pandas documentation. https://pandas.pydata.org/docs/, April 2023.

[15] C.A. Iglesias O. Araque, J.F. Sánchez-Rada. GSITK: A sentiment analysis framework for agile replication and development, January 2022.

[16] Scikit-learn: machine learning in Python. https://scikit-learn.org/stable/, June 2023.

[17] Happy Transformer. https://happytransformer.com/, June 2023.

[18] Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/, June 2023.

[19] Visual Studio Code - Code Editing. https://code.visualstudio.com/.

[20] Google Colaboratory. https://colab.research.google.com/.

[21] Jupyter Notebook. https://gpu2.gsi.upm.es/.

[22] DeepAI. Feature Extraction. https://deepai.org/machine-learning-glossary-and-terms/feature-extraction, 2020.

[23] GeeksforGeeks. Using CountVectorizer to Extracting Features from Text. https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/, 2022.

[24] Jairo. Modelos de clasificación en IA: cómo funcionan y en qué se utilizan. Las cosas de internet. https://lascosasdeinternet.com/tech-development/modelos-de-clasificacion-en-ia-como-funcionan-y-en-que-se-utilizan/, December 2022.

[25] Machine Learning Random Forest Algorithm. Javatpoint. https://www.javatpoint.com/machine-learning-random-forest-algorithm.

[26] N. Donges. Random Forest: A Complete Guide for Machine Learning. Built In. https://builtin.com/data-science/random-forest-algorithm, 2021.

[27] Support Vector Machine (SVM) Algorithm. Javatpoint. https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm.

[28] sklearn.svm.LinearSVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.linearsvc.html.

[29] S. Jessica. How Does Logistic Regression Work? KDnuggets. https://www.kdnuggets.com/2022/07/logistic-regression-work.html.

[30] R. Alberto. Regresión Logística con Sklearn. Medium. https://rubialesalberto.medium.com/regresion-logistica-con-sklearn-4384c707075d, December 2021.

[31] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[32] B. Dickson. Machine learning: What is the transformer architecture? TechTalks. https://bdtechtalks.com/2022/05/02/what-is-the-transformer/, August 2022.

[33] D. Sharma. Introduction to DistilBERT in Student Model. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/11/introduction-to-distilbert-in-student-model/, 2022.

[34] COVID-19 Fake News Dataset. Kaggle. https://www.kaggle.com/datasets/arashnic/covid19-fake-news, November 2020.

[35] COVID19 Fake News Dataset NLP. Kaggle. https://www.kaggle.com/datasets/elvinagammed/covid19-fake-news-dataset-nlp, March 2021.