UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND DEVELOPMENT OF A CLASSIFICATION AND DETECTION SYSTEM OF MALICIOUS EMAILS WITH NLP

GUILLERMO CANETE RIAZA

2021

TRABAJO DE FIN DE GRADO

Título:	Diseño y Desarrollo de un Sistema de Clasificación y Detección de Correos Maliciosos mediante Procesado de Lenguaje Natural
Título (inglés):	Design and Development of a Classification and Detection System of Malicious Emails with NLP
Autor:	Guillermo Canete Riaza
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

DESIGN AND DEVELOPMENT OF A CLASSIFICATION AND DETECTION SYSTEM OF MALICIOUS EMAILS WITH NLP

Guillermo Canete Riaza

2021

Resumen

Hoy en día, es increíblemente fácil tener acceso a un dispositivo electrónico, y en la mayoría de casos requieren cuentas de correo electrónico para gestionarlos. Esto se traduce en cerca de 4.000 millones de usuarios de correo electrónico en 2020 [2]. Es muy probable que la mayoría de estos usuarios reciban spam o correos maliciosos diariamente. El spam es el nombre que reciben los correos electrónicos no solicitados, en muchas ocasiones con fines publicitarios.

Teniendo en cuenta este problema, es importante encontrar una solución, que en este proyecto consiste en utilizar Procesamiento del Lenguaje Natural (NLP) y el Machine Learning para clasificar correos electrónicos en las siguientes categorías: ham (correos deseados), spam o correos fraudulentos.

Las tecnologías utilizadas en el proyecto son NLP y Machine Learning, que permiten el procesamiento y la clasificación de textos. Se explican con más detalle en el capítulo de Background, así como se también se describen los principales pipelines utilizados y posibles algoritmos.

La arquitectura propuesta consta de cuatro modelos de preprocesamiento y feature engineering, probados en cuatro algoritmos diferentes de Machine Learning. Los modelos se basan en: obtención directa de features, vectorizadores TFIDF y Count, combinación de características y Doc2Vec.

El capítulo de evaluación se centra en el análisis de los diferentes resultados obtenidos tras la implementación. En cada sección se estudian los diferentes resultados para dos conjuntos de datos, uno para la clasificación de spam y otro para la clasificación de correos fraudulentos, viendo los modelos óptimos de entre los propuestos.

En la conclusión, se presenta una visión más general del proyecto, analizando qué objetivos se han cumplido, cuáles fueron los problemas más importantes que ha habido que afrontar y en qué se puede mejorar el proyecto.

Palabras clave: NLP, Procesamiento de Lenguaje Natural, Machine Learning, clasificación de correos, detección de spam.

Abstract

Nowadays, it is incredibly easy to have access to an electronic device, in many cases needing email accounts to manage them. In addition, multiple platforms, such as social media or video on demand, require email addresses to sign in. This translates to around 4 billion email users in 2020 [2], and it is almost certain that those users already receive spam or malicious emails. Spam is the name given to unsolicited emails, most often used as advertisement.

With this problem in mind, it is important to find a solution. The solution proposed in this project is using Natural Language Processing (NLP) and Machine Learning to classify a set of emails into the following categories: ham (wanted emails), spam or fraudulent.

The technologies used for the project are NLP and Machine Learning, enabling text processing and text classification. They are explained in more detail in the Background chapter, as well as describing the main pipeline used in NLP and the different algorithms that can be used.

The architecture proposed consists of four models of preprocessing and feature engineering, tested on four different Machine Learning algorithms. The models are based on: direct feature obtaining, TFIDF/Count vectorizer, combination of features and Doc2Vec.

The evaluation chapter is focused on analyzing the different results obtained after the architecture is implemented and tested. Each section studies the different findings for two datasets, one for ham/spam classification, and another for ham/fraudulent email classification, finding the optimal models out of the ones proposed.

In the conclusion, a more general vision of the project is presented, analyzing which objectives were accomplished, what were the most important problems faced and were can the project be improved.

Keywords: NLP, Machine Learning, spam detection, email classification.

Agradecimientos

Me gustaría agradecer a mis amigos y familiares todo el apoyo que me han dado durante la carrera. Especialmente a mis padres, Mari Cruz y Evaristo, y a mis hermanos, Santiago y Rocío.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Introduction to the problem

In recent years, there has been an important growth in the number of people using electronic devices, which has led to an increase in the number of unwanted and malicious emails sent, currently accounting for 47% of the total [15]. This issue may be explained by the ease with which companies and cybercriminals can send a large number of emails at a high rate and very little cost. Emails of this sort are commonly known as spam and usually contain advertisements or some sort of malicious content. On the contrary, ham is the name given to desired emails, so in other words the opposite of spam. Malicious spam can be classified in the following ways:

- Spoofing: it is a kind of attack in which someone claims to be from a legitimate company by disguising things like the email address.
- Phishing: this type of emails try to deceive the receiver into providing passwords or personal information, by making the emails' appearance as the one from a legitimate company. Phishing can use spoofing to make it less obvious that it is an attack.
- Malware Attachments: sending malicious software with an email.

- Fraudulent emails: deceiving people into scams, encouraging transactions or revealing personal information.

Multiple solutions have been proposed to tackle this ongowing issue [12, 27, 28]. One such solution that has gained popularity in recent years relies on the use of Artificial Intelligence (AI). AI is a broad field which, in general terms, enables finely trained algorithms to learn from a set of inputs to perform specific tasks. One of the more relevant fields of AI to the topic of spam classification is known as Natural Language Processing (NLP), with Alan Turing as its precursor [19]. NLP is commonly based on the application of AI to linguistics by modelling human language in a way that computers can "understand".

Human communication is a complex process that involves combinations of words and phrases that can acquire different meanings depending on the context. Through NLP, we are able to accomplish, to a certain degree, text classification [5], language modelling [22], sentiment analysis [17], or information extraction [7]. This set of methods are commonly applied in chatbots, sentiment analysis in social media, customer service, etc. In order to achieve such tasks, we must extract the information a text or sentence is conveying by subdividing it into simpler units (words, morphemes, lexemes, etc).

To further understand the structure of human languages, we may differentiate between syntax and context. 1) Syntax can be defined as the way in which words are arranged to form sentences, 2) context relates to the different meanings a word or a set of words can have depending on when they are used. In other words, language can be ambiguous when the context is not considered, making it difficult for a machine to identify the information contained in a message. For this reason, different approaches for NLP have been proposed: rule-based, neural-network-based, and machine-learning-based [26].

Rule-based NLP is the oldest approach and it relies on building a set of rules to fulfill a task. The machine learning (ML) approach is focused on a statistical analysis of the word vectors that appear in a text. Machine learning based NLP relies on the use of traditional ML algorithms in the same way they are utilized with other types of data. Although this method fails at capturing context, it can still be advantageous in certain applications. The neural network model is used in a similar way to the ML approach but differs in the fact that it does not require feature engineering.

In this project, I will expand on the use of ML based NLP in order to classify emails depending on the intent of the sender. This method is suitable for this application since for email classification context is not necessarily relevant.

1.2 Objectives

The objectives of this project are:

- To perform feature extraction from a pool of emails. By reducing a complex problem to a set of simpler tasks we will be able to better understand the underlying processes.
- To gain knowledge in the application of some of the main NLP libraries and algorithms, such as the Natural Language Toolkit (NLTK).
- To analyse and compare the results from different algorithms in order to combine them effectively.
- To produce a program that combines a set of NLP related methods that achieve the best performance when classifying emails as spam or with a malicious intent.

By successfully completing these tasks I will broaden my understanding of the Machine Learning techniques that are most applicable to Natural Language Processing. In addition, I will produce a basic program capable of classifying emails that will be available in an open source repository.

1.3 Methodology

The procedure will consist of dividing the problem into features, apply a classification technique, and produce an output (spam/ham). For this purpose, features will be extracted by applying different NLP algorithms to the subject and body of the emails.

First, the data set will be managed using the Pandas dataframes. This will be done by extracting the relevant fields and restructuring the data into the Pandas format. Then, the data will be cleaned by removing empty or invalid fields. Once the structure of the data is defined, it will be possible to start the preprocessing phase in order to apply the Machine Learning algorithms of interest. Preprocessing is often structured in the following way:

1. Removing Punctuation
2. Stemming and lemmatizing
3. Removing stop words
4. Tokenizing

5. Vectorizing

Once the preprocessing phase is finished, we can start applying the NLP algorithms to obtain a set of features. Said features can be joined afterwards and processed by Machine Learning algorithms to obtain the output. Subsequently, to ensure the best performance, we will be examining the results from applying these algorithms, by comparing the different outputs obtained from a set of algorithms. The performance measure used will be the classification the accuracy of these algorithms. The project will be programmed in Python 3.0, in the PyCharm IDE environment, and using Anaconda for library management.

CHAPTER 2

Background

In recent years, technology has increasingly become more accessible to the general public, allowing people to use tools that, not too long ago, where only available to a few. These technologies span from programming languages to sophisticated machine learning libraries. These tools, that can be extremely complex, now can be used in a plug and play manner. In this chapter, we will be explaining which technologies made possible this project, and will analyze previous work that has attempted to solve similar problems. For this specific project the following tools have been most helpful:

2.1 Enabling Technologies

There are various technologies that make this type of projects possible. The most important ones in our case are the following:

2.1.1 Python

The first version of Python was released by Guido Van Rossum in 1991. It is an interpreted high-level programming language that stands out for its simplicity and ease to learn. In

addition, it is object-oriented and has memory management. This features make it ideal for newcomers, and fast prototyping and scripting.

Some of the reasons making Python widely extended as the programming language used in Machine Learning are:

- Simplicity: simplicity is something to take into consideration when making a project. In this case, the simple syntax and indentation system makes the code easier to follow and faster to develop.
- Libraries: this is one of the main reasons since having a wide variety of libraries will make it easier to develop programs. In this case, Python has many libraries implementing crucial algorithms and functions for Machine Learning and Natural Language Processing. Some of the biggest libraries are Scikit-learn, Pandas, Keras, etc.
- Documentation: when a programming language or library is not well documented, it can make finding errors and information very difficult. This is not the case with Python.
- Reliability: it has constant updates and the capability of being deployed in many environments. It also has great tools for library management.
- Precompiled: although compiled languages are the norm in any application that requires fast run-times and low storage capacity, precompiled languages require less overhead, making it more accessible to the public.

2.1.1.1 Pandas

Pandas [20] is an open-source Python library that has been under development since 2008. It is used for data analysis and data manipulation. The main advantage it has is that it provides organised and clear data structures.

The two basic data structures are Series, for 1-dimensional arrays, and DataFrames for multi dimensional structures. Dataframes use the object Index as an unambiguous number referring to each row. In these structures, it is very simple to swap, remove, add or modify the data. In the preprocessing stage, it is common to modify the dataframe, removing invalid or empty rows. Pandas allows specifying what kind of data each column has to contain, handling rows that do not meet the requirements. With the function *dropna* we can remove said rows. There is also an option to fill empty values automatically with a

specified value (*fillna*). It can be useful to fill these empty values with the mean value of the column for example. Combining and joining dataframes is quite simple too, using *concat* to join by rows or *merge* to join by columns. There are multiple delimited text formats supported, such as CSV, JSON, HDF5, etc. Another advantage of using Pandas is how easy to read the data sets are when converted into a dataframe.

2.1.1.2 Scikit-learn

Scikit-learn [21] is a Machine Learning library built with support for NumPy, SciPy and Matplotlib, that features a range of common algorithms. These features can be used for the following tasks:

- Classification
- Regression
- Clustering
- Dimensionality Reduction
- Model Selection
- Preprocessing

The main Scikit functionalities used in this project are the Standard Scaler, TfidfVectorizer and CountVectorizer, as well as the different classification algorithms. A great advantage this library has, is the capability of comparing the results of different algorithms with ease, enabling adapting the code to our necessities.

2.1.1.3 Matplotlib

This Python open-source library [13]is used to produce publication-quality 2D and 3D graphs, showing visual results for programs. It makes creating and customizing plots very easy, being able to generate different types of plots to provide the best visual aid. Some of these graphs are, for instance: curves, histograms and scatter plots.

2.1.2 Natural Language Processing

Language is the combination of symbols and words used to communicate between humans. As it was explained in the Introduction, Natural Language Processing is a mixture of

linguistics and Artificial Intelligence.

2.1.2.1 Artificial Intelligence

Artificial Intelligence is the field of computer science that focuses on making computers "intelligent". In its more basic form, AI is the combination of computer science concepts with robust data sets, enabling problem-solving. In addition, the algorithms used in AI try to make predictions or classifications based on input data. The two types of Artificial Intelligence are: 1) Weak AI focused on performing a specific task, 2) Strong AI a theoretical concept where machines have a broader sense of intelligence.

2.1.2.2 Linguistics

Linguistics is the scientific study and modelling of language [18]. Language can be divided into five levels:

- Phonology: it is the study of sounds humans can make, that are usable in language. Each language has shared sounds and characteristic ones, making it a different case of study for each. Phonemes are the smallest units of language and they do not have meaning by themselves. The English language is composed of a total of 44 phonemes.

- Morphology: it is the study of morphemes, the smallest units of language with meaning and lexemes, structural variations applied to morphemes modifying their meaning. It takes into consideration single words only, it does not study the structural relationship between words in a sentence.

- Syntax: sentences are structured groups of words that follow a set of rules, these are the area of study of syntax.

- Semantics: whilst the other fields of linguistics are more focused on the structure of language, semantics is focused on the meaning of words and sentences. Furthermore, sentences are can be statements (true or false), questions or requests.

- Pragmatics: it is similar to semantics, but it centres on what sentences and words imply in different situations.

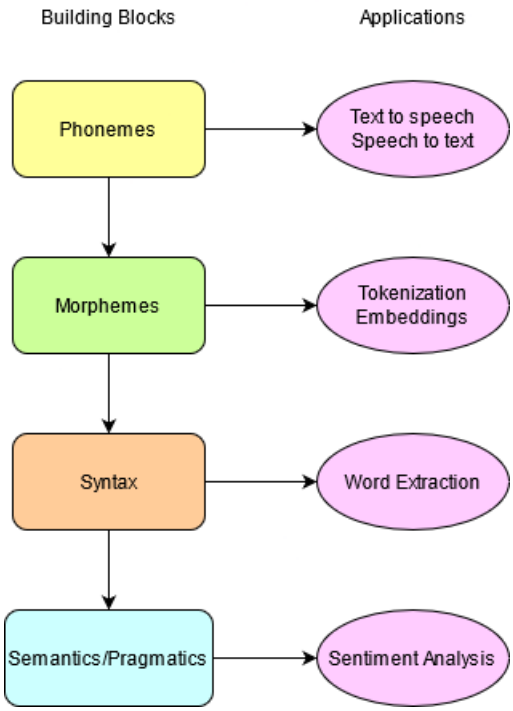

Figure 2.1: Building Blocks and Applications

2.1.2.3 Building Units

Natural language is the ordinary language that people use, originated from years of evolution and adapting to human needs. The main goal of NLP is making computers able to interact with humans by either processing or generating language, normally both.

As it can be seen in Fig. 2.1, for analysing and generating language we need to divide it into building blocks that computers can use. These blocks are the ones discussed in the Linguistics section, and each can be used for different tasks inside NLP.

2.1.2.4 Context

A factor to take into consideration when building an NLP project is context, as words have more than one meaning and it depends on where and when they are used. So, a broad definition of context can be the meaning a word or phrase has depending on what it is surrounded by. Generally, this is studied by the combination of semantics and pragmatics. Although humans tend to understand context easily, computers can have a hard time capturing certain meanings in different situations. This problem can be due to multiple causes:

- Ambiguity: words having multiple meanings that can only be understood by their context.

- Common Knowledge: as we grow up we acquire certain knowledge that is obvious to us and common in our environment or culture.

- New words: as it was said before, language adapts to human needs, meaning it is dynamic and keeps evolving. So something to take into consideration is that new words keep appearing.

2.1.2.5 NLP Approaches

As it was mentioned in the introduction, there are three main approaches [26] in which NLP tackles problems, these are:

- Rule-based or Heuristics: it consists of building a set of rules to do a task, requiring previous knowledge from the developer in the field of work. In most cases, they require a dictionary of words and the use of Regular Expressions or context-free grammar(CFG).

- Machine Learning: the distributional approach using ML is focused on comparing the similarity of words based on the statistical frequency of occurrence in other texts. For instance, the word "calculus" is more probable to appear in a text about mathematics than in one about politics. After having the statistical representation of a text, traditional machine learning algorithms can be applied. Although this approach does not rely on the meaning of words, it can be useful for text classification or checking if words are being used similarly. The Machine Learning approach, either supervised or unsupervised has three shared steps: feature selection, learning the model from the features and evaluating the model. This model is very scalable and flexible, as it treats text as arrays of data.

- Neural Networks: these networks are composed of layers of nodes, in which the first layer is the input, the last one is the output and all the layers in between remaining hidden. Nodes are connected to other nodes and they have an associated weight and threshold, that determine the importance of a variable in the model. This method has the advantage that, unlike ML, does not need the feature selection phase as the neural network does it. Some of the more common neural networks used in NLP are recurrent neural networks, convolutional, transformers and auto-encoders. Two of the

main problems of using this approach are that it is easy to do overfitting when the data sets are not big enough and the high cost involved in training the networks.

2.1.2.6 Pipeline

Pipelines [26, 9, 16] allow addressing problems in an organised and structured way. The most common NLP pipeline is structured as seen in Fig. 2.2: data acquisition, preprocessing, training and evaluating the results. However, some of the preprocessing steps can be done in a different order if necessary.

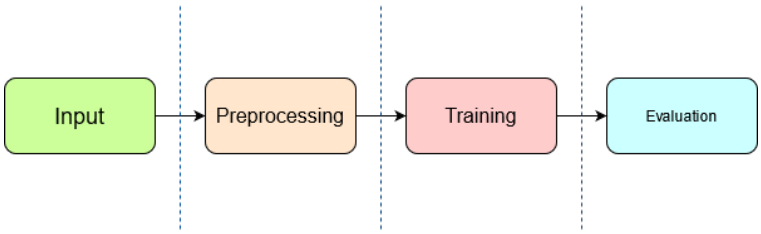

Figure 2.2: NLP Pipeline

1. Data Acquisition: to build the model we need data that in the case of NLP is text. Data can be acquired in multiple ways but a common method in NLP is using open source datasets available or web scraping to retrieve information from websites and social networks. Another method worth to consider in this project's context, is creating an email account spam and malicious email gathering. The next step is extracting the raw text from the input and cleaning it up by removing empty or invalid fields. In some cases spelling correction can be added due to the fact the scraped data will probably have spelling errors and it is necessary to use as much clean data as possible to train the models more effectively.

2. Preprocessing: once we have the raw text and have cleaned the dataset we can advance into the preprocessing stage, which will allow advancing into the next stage.

   (a) Normalization: in this step, all the text is converted to lower case and punctuation is removed.
   [*'They won the trophy easily'*] would be [*'they won the trophy easily'*]

   (b) Tokenization: the text is divided into an array of smaller units, normally words, called tokens.
   [*'they won the trophy easily'*] would be [*'they', 'won', 'the', 'trophy', 'easily'*]

(c) Stemming/Lemmatizing: Stemming consists of removing the suffixes of words leaving the core or stem, even though many times these stems are not linguistically correct.
Lemmatizing instead of reducing words to their stem, reduces them to its lemma, their basic form.
Stemming: [*'they', 'won', 'the', 'trophy', 'easily'*] would be [*'they', 'won', 'the', 'trophi', 'easili'*]
Lemmatization: [*'they', 'won', 'the', 'trophy', 'easily'*] would be [*'they', 'win', 'the', 'trophy', 'easily'*]

(d) Stopwords: removing irrelevant words can be important before applying further algorithms, that is why stopwords are removed. It requires a dictionary of words considered stopwords for the language in which we are working. It is also possible to add more stopwords if necessary.
[*'they', 'won', 'the', 'trophy', 'easily'*] would be [*'they', 'trophy', 'easily'*]

(e) Feature Engineering: After the first part of the preprocessing has been done, we have to do feature engineering before we apply the machine learning algorithms. Machine learning works with numbers, not with text, so in this phase, we will have to map the text into a set of numerical features. The definition of feature is: "A prominent or distinctive part, quality, or characteristic", which is what we will have to do with our preprocessed text. We will have to find features that are representative enough to characterise the text when we process it with the ML algorithms. Some of the most frequent methods will be discussed below.

Once the feature engineering is done, it is important to do feature scaling, because machine learning algorithms work better when the attributes are in the same range. Two common ways of achieving this [6, 11] are 1) normalization: shifting and rescaling values so that they range from 0 to 1, or 2) standarization: subtracting the mean value so that the total mean is zero. In this project we will be using the second method via the Standard Scaler provided by Sklearn.

2.1.2.7 Feature Engineering Algorithms

In this part we will be describing some of the many feature engineering algorithms [24, 25] A vector space model is a way of representing text units as vectors to use them for machine learning. This model is based on the concept of similarity [14, 26]. A way of calculating

the similarity between texts is cosine similarity, defined as:

$$\cos\varphi = \frac{A \cdot B}{||A|| \cdot ||B||}$$

The closer the two vectors are, the smaller the angle, meaning the cosine is closer to 1 and the words are similar. The more separated they are, the closer they will be to 180, meaning the cosine will approximate -1 and the words are opposite. When the cosine equals zero, it means the vectors are orthogonal and the words are not correlated.

The following algorithms fall under the category of vector space models and work similarly.

- One Hot Encoding: [8] given a vocabulary of k words, we assign to each word an integer between 1 and k, that we will call x to encode it. Now, each word will have assigned a vector of k dimensions, in which every value is 0 except for the value in the position x of the vector, which will be 1. While it is a simple way of converting words into vectors, it is not the most efficient since the dimensions of the vectors will depend on the vocabulary of the text analysed. An example of one-hot encoding:

Corpus	Encoding	One-Hot Encoding
they	1	[1,0,0,0,0]
won	2	[0,1,0,0,0]
the	3	[0,0,1,0,0]
trophy	4	[0,0,0,1,0]
easily	5	[0,0,0,0,1]

- Bag of Words: the bag of words model or BoW [24] is a way of representing text as vectors similar to one-hot encoding. In this case a text is viewed as a bag in which context and word order are ignored. The most relevant thing is the set of words used and the times they occur in said text. This way we can represent a text with a vector of k dimensions, being k the number of unique words in our vocabulary. In this vector, the positions corresponding to words in the text equal the number of occurrences of those words and zeros in the rest of positions. It is more clear in the following example:

With the corpus : " *They won the trophy easily*" and " *The car won the race*"

	They	Won	The	Car	Trophy	Easily	Race	Vector
Text 1	1	1	1	0	1	1	0	[1,1,1,0,1,1,0]
Text 2	0	1	2	1	0	0	1	[0,1,2,1,0,0,1]

The main issue BoW has is that the vector dimensions increase with the size of the vocabulary and it completely ignores context. Another drawback is the sparsity of the vectors as most values will be zeros.

- N-grams: the n-gram [1] approach centres on analysing the text by dividing it into groups of n words, allowing to see which words are more probable to go together. When using n-grams, we are considering a Markovian model, in other words, we are considering that each word depends on the previous one, making it possible to predict the next most probable word. Language can be modelled as a probability distribution of words, so the chain rule for the n-gram model is:

$$p(w_1, w_2, w_3, ..., w_n) \approx p(w_1)p(w_2|w_1)p(w_3|w_2, w_1)...p(w_n|w_{n-1}, w_{n-2}, ..., w_{n-N}))$$

To estimate the maximum probability, we can use the maximum likelihood estimation or MLE, that for a bigram model would be:

$$p(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{\sum_w c(w_{i-1}, w)}$$

An example of this, with <s> and </s> indicating the start and end of a sentence:

<s>I ate good </s>, <s>I ate a tasty apple</s>, <s>My friend went fishing</s>

$$p(I| < s >) = \frac{c(I, \ < s >)}{c(< s >)} = \frac{2}{3}$$

$$p(good|ate) = \frac{c(ate, \ good)}{c(ate)} = \frac{1}{2}$$

$$p(apple|tasty) = \frac{c(tasty, \ apple)}{c(apple)} = 1$$

It is a way of solving the problem that BoW has with context. However, both models are often combined in what is called Bag of N-grams, taking sets of words instead of

Term	Count
I	1
Ate	1
Well	1

Term	Count
I	1
Ate	1
A	1
Tasty	1
Apple	1

individual ones. Although this model captures context better than the ones mentioned before, and the co-occurrence of n-grams in different texts will result in a closer similarity, the resulting matrix will still be sparse.

- TF-IDF: it stands for term frequency-inverse [26] document frequency and it is used to measure the frequency of a word or n-gram in a document. Unlike the previous models. TF-IDF does not treat all words with the same importance. If a word appears a lot in a document but does not appear in the other documents, it means it provides more information.

  Term frequency measures how many times a word appears in a document.

  $$TF = \frac{Number\ of\ occurrences\ in\ a\ document}{Number\ of\ words\ in\ the\ document}$$

  Inverse Document Frequency measures how much information a term provides to the whole corpus by analysing how rare it is.

  $$IDF = log\ \frac{Number\ of\ documents\ in\ the\ corpus}{Number\ of\ documents\ containing\ the\ term}$$

  Term frequency-inverse document frequency:

  $$TFIDF = TF \times IDF$$

  For instance:

  Document 1: "*I ate well*" ; Document 2: "*I ate a tasty apple*"

  For every word in document 1: $TF_{doc1} = \frac{1}{3}$

  For every word in document 2: $TF_{doc2} = \frac{1}{5}$

  The idf for "ate": $IDF("ate") = log_2\ \frac{2}{1} = 1$

$$TFIDF(\text{"}ate\text{"},\ Doc\ 1) = 0.33 \times 1 = 0.33$$

$$TFIDF(\text{"}ate\text{"},\ Doc\ 2) = 0.2 \times 1 = 0.2$$

- Word Embedding: many words are related even though they are not similar. Word embedding [4] is a much more complex model that can capture semantical relationships between words. A word embedding is a vector that represents a set of words. A technique frequently used is to create embeddings using Wor2vec. This architecture has as its core a 3 layer neural network used to learn how words are associated in a corpus. The neural network can use 1) the continuous bag of words model, a variation of BoW, that captures word occurrence with respect to a word. This models allowing prediction given a context. Or it also can use 2) skip-grams, that opposed to CBoW, try to predict context when given a word. Training word embeddings can be a costly task, so a common solution is using pre-trained ones, such as Gensim Word2vec. The Doc2Vec architecture is another possibility, being is based on the Word2Vec model, but instead of vectorizing words, it creates that represent documents.

2.2 Related Work

Spam detection is a well-researched problem with lots of examples, in this section we will be talking about two of them.

The first project to discuss is titled "Mail Spam Detection: A Method of Metaclassifier Stacking" as found in [28]. It is focused on email spam detection using hybrid methods and evaluating the results. In this project, first, the author explains different types of classification as well as some models. After explaining the methodology to achieve classification, three models are chosen. Finally, the classification is done, testing the different combinations of those three models. In our project we are going to do something similar with two of the models, evaluating if there is an improvement when combining them.

In this case, a similar project is "Adversarial Attacks on SMS Spam Detectors" by Lowri Williams and Irene Anthi. This article [27], is a part of the whole project, centered in classifying SMS messages. Although it uses different procedures to the ones implemented in this project, it was useful to see how some of the problems were dealt with. In their project, they use the a model based on word embeddings using Gensim Word2vec library for feature engineering. From the two options mentioned when describing Word2Vec, they use skipgrams. This enables predicting context out of the SMS messages they are processing.

CHAPTER 3

Architecture

In this project, the data used as an input comes from two open source datasets from the Kaggle website. The project will be divided in 2 parts: one for detecting if an email is spam or ham, and another for detecting if an email is fraudulent. To achieve the best results, we will study different models to analyse which one works better for each dataset.

There will be four different models varying in the feature engineering phase, and each model will be tested with various machine learning algorithms. Each model has its own pipeline that can vary, but the basic one is the one seen in Fig. 3.1.

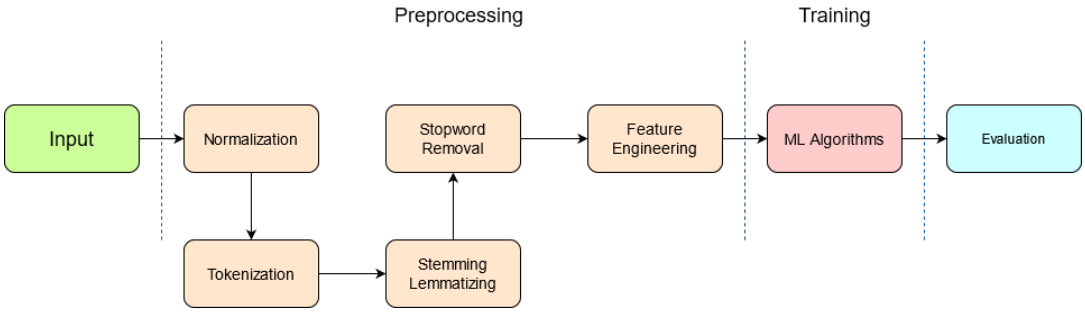

Figure 3.1: Architecture

3.1 Model 1: Word Count and Sentiment Analysis

In this model, the text will be characterized by the number of words in each text and its sentiment. The preprocessing will consist of obtaining both parameters. Obtaining the word count is straighforward, however, for the sentiment we will need to use the Sentiment Intensity Analyzer from NLTK. This library returns a value ranging from -1 to 1, representing sentiment. After these two values are obtained, they have to be scaled with the Sklearn Standard Scaler to prepare them for the training phase. The pipeline will end up with the structure seen in Fig. 3.2.

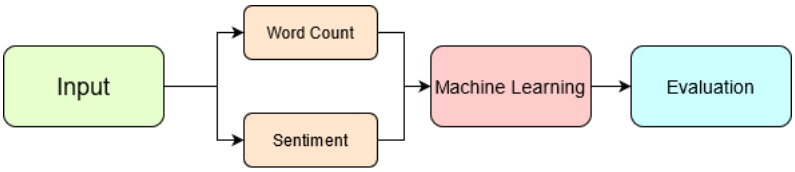

Figure 3.2: Model 1

3.2 Model 2: TF-IDF/Count Vectorizer

In this model, we will be using the Sklearn libraries Tfidf Vectorizer and Count Vectorizer, testing which one gives better results. The normalization, tokenization, lemmatization and stopword removal is done automatically by these libraries. So the preprocessing phase is reduced to applying the TFIDF and Count Vectorizer algorithms. After obtaining the features, we use the Standard Scaler to prepare the data for training. The pipeline will end up looking as seen in Fig. 3.3.

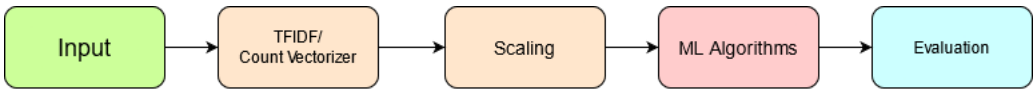

Figure 3.3: Model 2

3.3 Model 3: Model 1 + Model 2

Model 3 is a combination of the two previous models, including the preprocessing from model 1 and from the Sklearn libraries. In the Feature Engineering phase, once the vectorizer is trained, the word count, sentiment and vectors are concatenated into a single matrix that is passed to the Standard Scaler for further training, as it can be seen in Fig. 3.4.

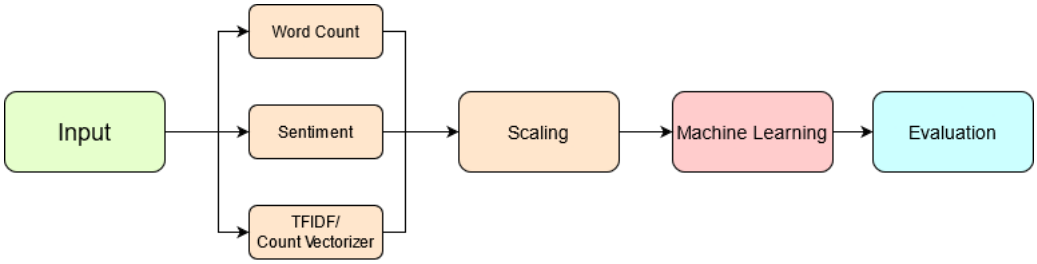

Figure 3.4: Model 3

3.4 Model 4: Doc2Vec

In this model, we will need to do the first part of the preprocessing manually as seen in Fig. 3.4, because it is not included in the Doc2Vec library:

1. Normalization: removing punctuation, removing digits and making every character lowercase.

2. Stopword Removal: using the nltk corpus of stopwords, we proceed to remove them from our text, leaving only the relevant words. It allows to add custom words to the stopwords corpus if necessary.

3. Lemmatizing: the lemmatizing is done with the nltk Word Net Lemmatizer, applied to the text without stopwords.

4. Tokenization: with the nltk Word Tokenizer, the text is converted into tokens, ready for the feature engineering.

The feature engineering will be done with the Doc2Vec library from Gensim. It works similarly to the Word2Vec algorithm, but instead of providing word vectors, it provides a vector for each text. However, before generating these vectors, it is necessary to build a vocabulary and train the algorithm.

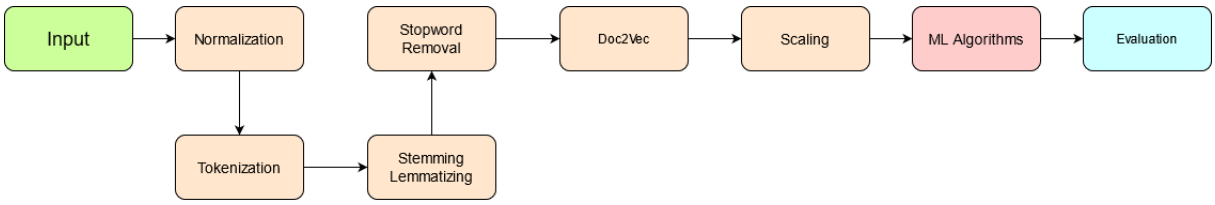

Figure 3.5: Model 4

3.5 Training: Machine Learning Algorithms

Each model is going to be tested with various ML algorithms to evaluate the different results, as seen in Fig. 3.6. The 4 algorithms used for every model will be:

- Decision Tree Classifier
- Stochastic Gradient Descent Classifier
- Linear Regression
- Random Forest

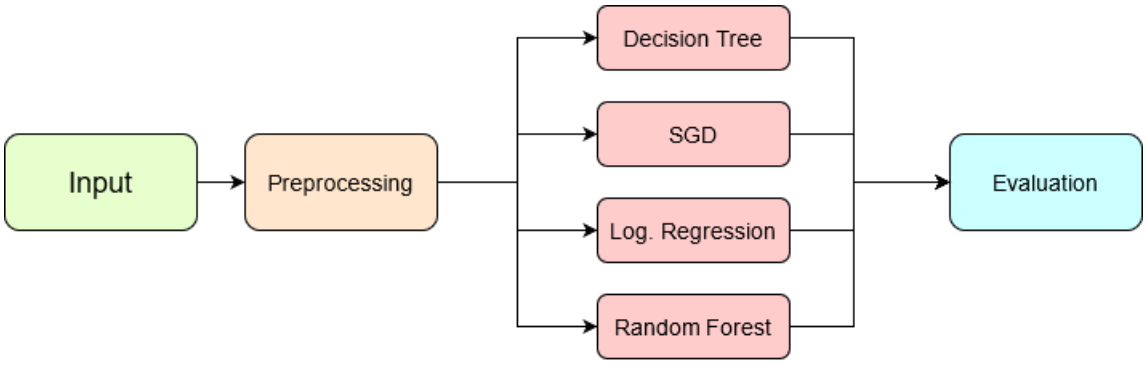

Figure 3.6: ML Algorithms

CHAPTER 4

Evaluation

4.1 Introduction

In this chapter, we will be covering the evaluation of the different models used to detect spam and fraudulent emails, as mentioned in the architecture. First, we will be examining the datasets used in the project and what they are used for. After, we will be analyzing the results of the different models and machine learning algorithms for each dataset, using as a measurement of success the accuracy of the classifiers.

4.2 Performance Measurements

There are other ways to evaluate the performance of a classifier rather than using only the percentage of emails classified correctly [11]. One of these is the confusion matrix, which represents positive values classified as positive (true positive), negative as negative (true negative), positive as negative (false positive) and vice versa (false negative). This can be seen more clearly in the Fig. 4.1.

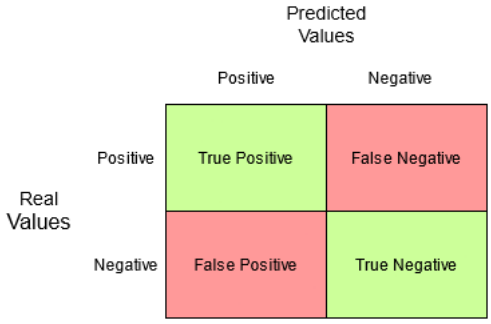

Figure 4.1: Confusion Matrix

From this matrix we can extract some more measurements, such as precision, recall and F1-score. Precision measures the true positives (TP) out of all positives:

$$precision = \frac{TP}{TP + FP}$$

Recall measures the true positives (TP) out of all the real positives:

$$recall = \frac{TP}{TP + FN}$$

The F1-score is a useful value when comparing classifiers and it is obtained by combining precision and recall as follows:

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

4.3 Datasets Used

In this project, we have used two datasets, one for ham/spam classification and another for fraudulent email detection.

The ham/spam dataset [10] was obtained from the Kaggle website and was already labelled. It is a '.csv' file in which each sample consists of: id number, text, label (ham or spam) and a numeric label. This file format is simply converted into a Pandas dataframe, enabling handling each field easily. Once it is converted into a dataframe, the text is divided into two fields, subject and email body, due to the fact that they were merged.

The dataframe's data looks as seen in the table below. In addition, in Fig. 4.2 it can be seen the distribution of ham and spam in the dataset. Having much more ham emails than spam ones may condition the spam detection accuracy.

subject	text	label
Re: Indian Springs	this deal is to...	0
Looking for medication?...	it is difficult to make ...	1
re: first delivery ...	vance, r ndeal number...	0

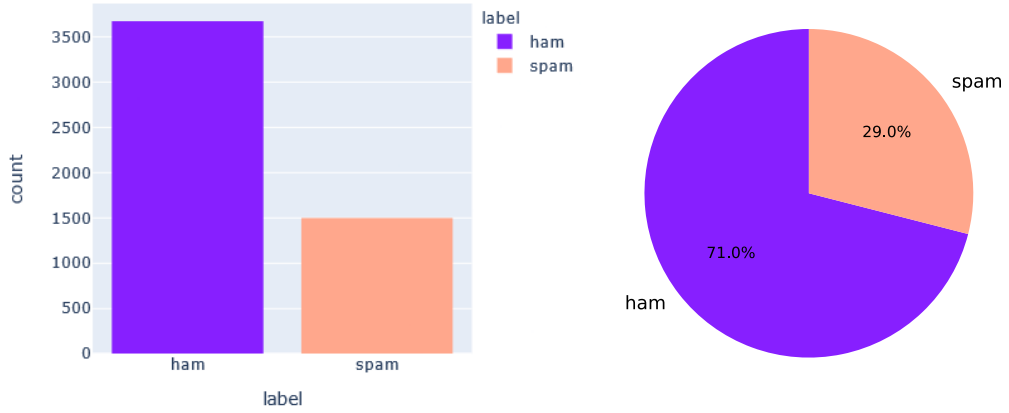

Figure 4.2: Distribution for Dataset 1

The second dataset was also obtained from the Kaggle website and contains only fraudulent emails. The file is in '.txt' format and contains the raw emails without separation, meaning that it was necessary to clean and label the dataset. Each email is composed of around 10 fields such as: "From", "To", "Return path", "Content type", etc. These fields were ignored in this project and we only kept the subject and the body. The label had to be added manually by creating a new field in the dataframe, which equals 1 (fraudulent) for every email. This leaves a dataset only composed of fraudulent emails (all labels equal to 1). So after this, the dataset was merged with the emails labelled as ham from the first dataset, labelled as 0. Fig. 4.2 shows the distribution of ham and spam for the second dataset. In this case, the dataset is much more balanced, having 56% ham and 44% fraudulent emails. The dataframe's data looks as seen in the table below.

subject	text	label
URGENT BUSINESS ASSIST...	FROM:MR. JAMES NGOLA...	1
URGENT ASSISTANCE...	Dear Friend, I am Mr. Ben...	1
GOOD DAY TO YOU	FROM HIS ROYAL MAJESTY ...	1

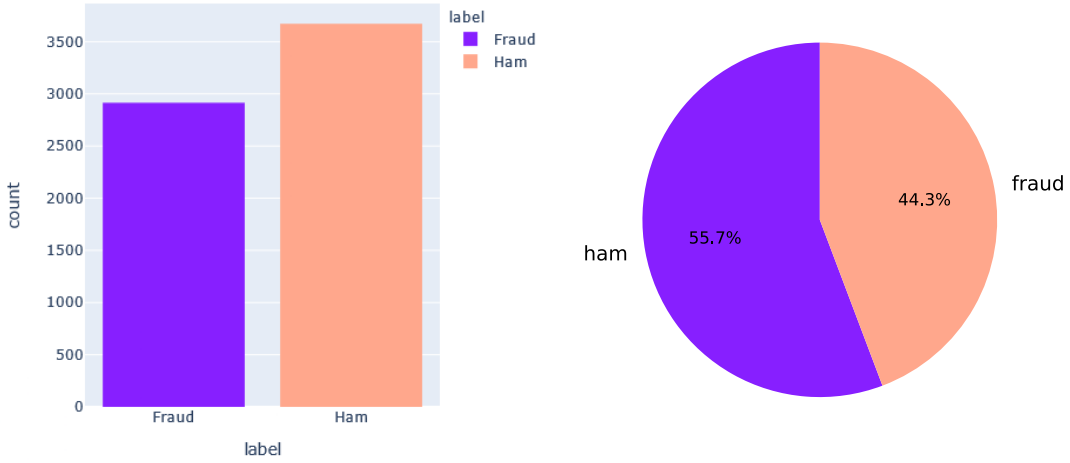

Figure 4.3: Distribution for Dataset 2

4.4 Model Evaluation

This part of the project will test the four models and machine learning algorithms with the ham/spam (Dataset 1) and with the ham/fraudulent dataset (Dataset 2). In addition, each model will be evaluated selecting the ML algorithm with the best accuracy.

4.4.1 Model 1: Word Count and Sentiment Analysis

The first model has as main features the word count and sentiment: 1) word count is just the number of words in each text and 2) sentiment is represented by a number between -1

and 1. These two variable are combined to form the features and then are scaled to prepare them for training. Even though it is a rough approach, it can be useful to see the accuracy that can be achieved with this model in order to combine it with other features.

4.4.1.1 Dataset 1

With the ham/spam dataset, we can appreciate in the word count histogram (Fig. 4.4) that the subject is less than 50 for most emails in both the ham and spam categories. The word count for the main text generally remains below 2000, but ham texts seem to have most values closer to zero.

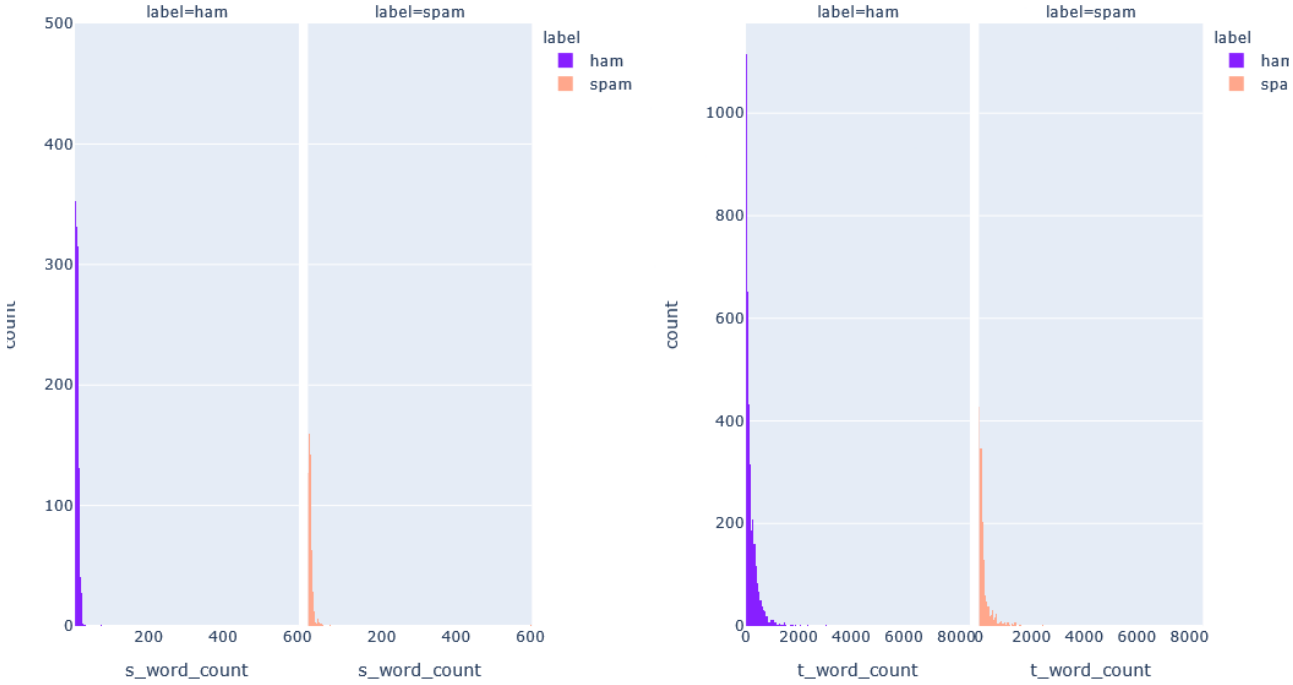

Figure 4.4: Word Count Histogram for Dataset 1

In the sentiment histogram seen in Fig. 4.5, there is no obvious pattern, except that ham emails tend to be more neutral.

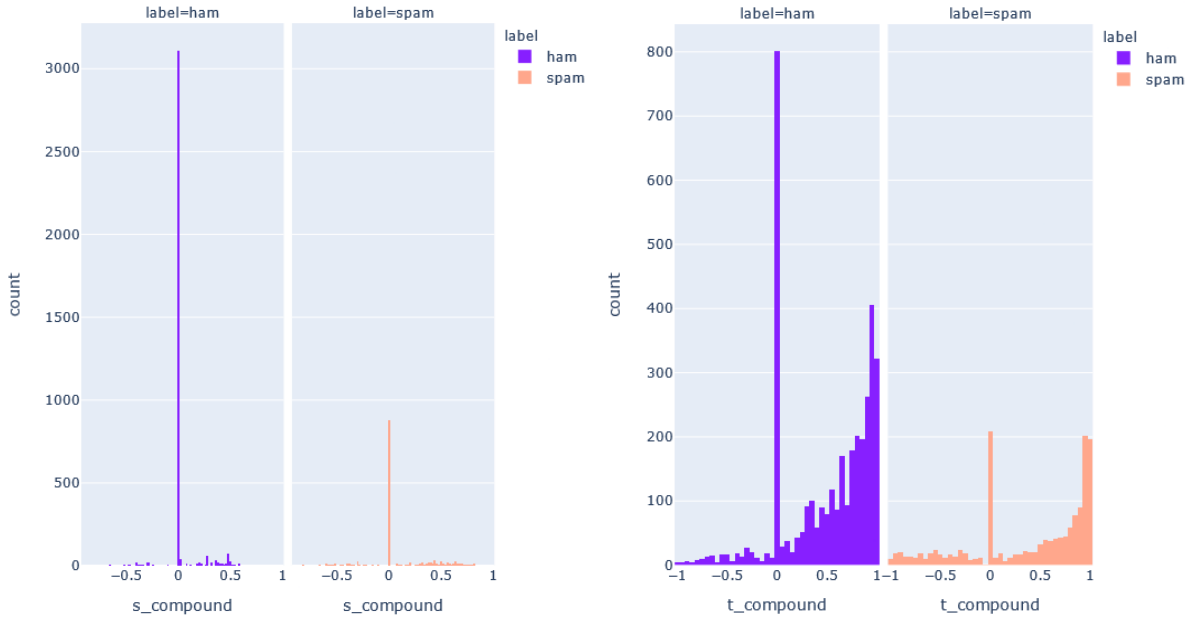

Figure 4.5: Sentiment Histogram for Dataset 1

Studying the correlation between this variables and the label, we can see that there is practically no linear relation between them, which can make some machine learning algorithms perform badly.

	Word Count	Sentiment
Label	0.066	0.132

Table 4.1: Correlation matrix for the subject

	Word Count	Sentiment
Label	0.006	-0.0729

Table 4.2: Correlation matrix for the text

Looking at the percentage of mails correctly classified, that can be seen below, we can see that the Random Forest is the one getting more right.

ML Algorithm	Success Rate
Decision Tree	66.28 %
SDG	75.36 %
Logistic Regression	73.82 %
Random Forest	77.87 %

In the Fig. 4.6, we can see from the f1 value that, indeed, the Random Forest algorithm performs the best for both classifying ham and classifying spam.

Decision Tree

	Precision	Recall	f1
0	0.77	0.73	0.75
1	0.45	0.50	0.47

SGDC

	Precision	Recall	f1
0	0.75	0.97	0.85
1	0.79	0.25	0.38

Logistic Regression

	Precision	Recall	f1
0	0.74	0.96	0.84
1	0.72	0.21	0.33

Random Forest

	Precision	Recall	f1
0	0.79	0.93	0.85
1	0.72	0.43	0.54

Figure 4.6: Precision and Recall Matrices for Model 1.1

4.4.1.2 Dataset 2

With this dataset, it is easier to see certain patterns in the histograms (Fig 4.7 and Fig. 4.8), such as: the word count for the fraud text has a higher mean than the ham text, and they clearly tend to be positive when looking at the sentiment.

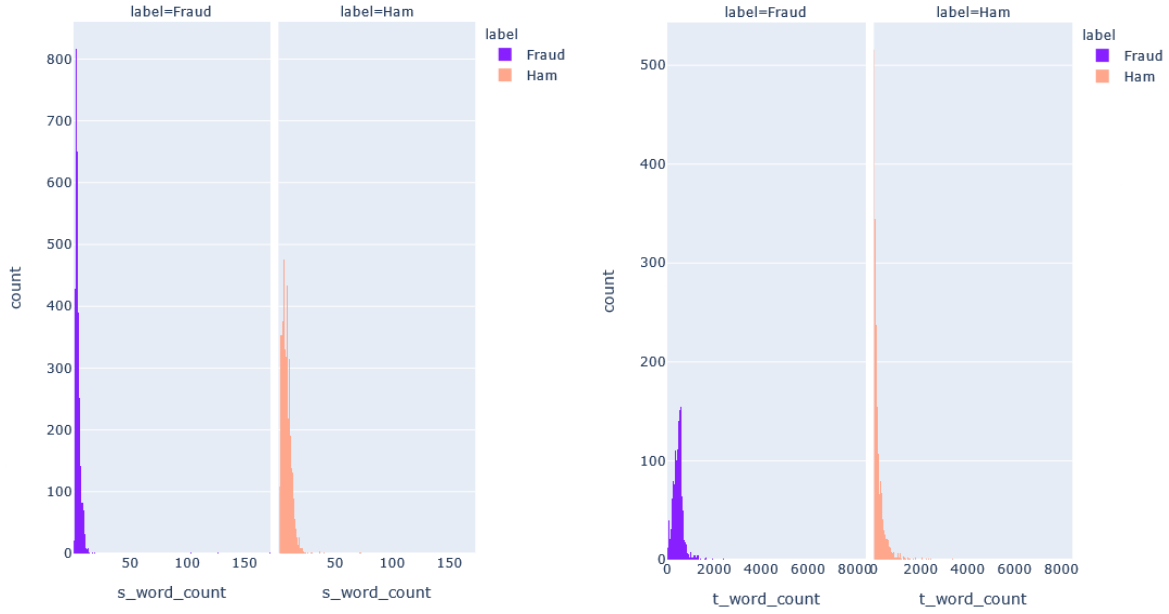

Figure 4.7: Word Count Histogram for Dataset 2

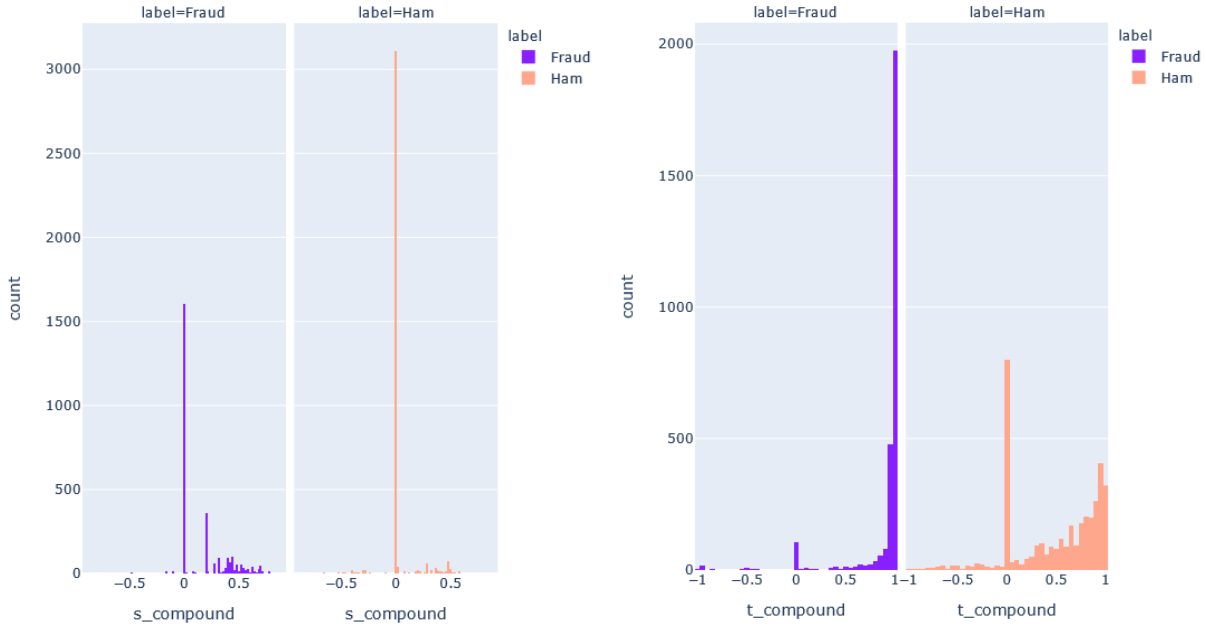

Figure 4.8: Sentiment Histogram for Dataset 2

In this case, there is some correlation between the variables and the label, which means there is a weak linear relation between them. It can be seen in these tables:

	Word Count	Sentiment
Label	-0.336	0.132

Table 4.3: Correlation matrix for the subject

	Word Count	Sentiment
Label	0.307	0.4547

Table 4.4: Correlation matrix for the text

Looking at the success rate and precision and recall matrices, we can observe that the Random Forest is by far the algorithm that performs best in this model, achieving a 90% of emails correctly classified.

ML Algorithm	Success Rate
Decision Tree	84.07 %
SDG	44.08 %
Logistic Regression	73.75 %
Random Forest	90.06 %

In the Fig. 4.9, we can see from the f1 value that, the Random Forest algorithm performs the best again for both ham and spam.

Decision Tree

	Precision	Recall	f1
0	0.80	0.94	0.86
1	0.91	0.73	0.81

SGDC

	Precision	Recall	f1
0	0.48	0.45	0.47
1	0.40	0.43	0.41

Logistic Regression

	Precision	Recall	f1
0	0.74	0.80	0.77
1	0.74	0.66	0.70

Random Forest

	Precision	Recall	f1
0	0.87	0.96	0.91
1	0.72	0.83	0.88

Figure 4.9: Precision and Recall Matrices for Model 1.2

4.4.2 Model 2: TFIDF/Count Vectorizer

In this model, we will test the results for TFIDF and Count Vectorizer separately for both datasets. These Scikit libraries do not require any preprocessing as they already include it and saves a few steps. The min-df parameter represents the minimum number of times a term must appear to be considered by the function, in this case is set to three. Another important thing to consider is what range of n-grams works best for this model. By defining a test function that plots the probability of success for each model and considering different n-gram ranges, we obtain the graph seen in Fig.4.10:

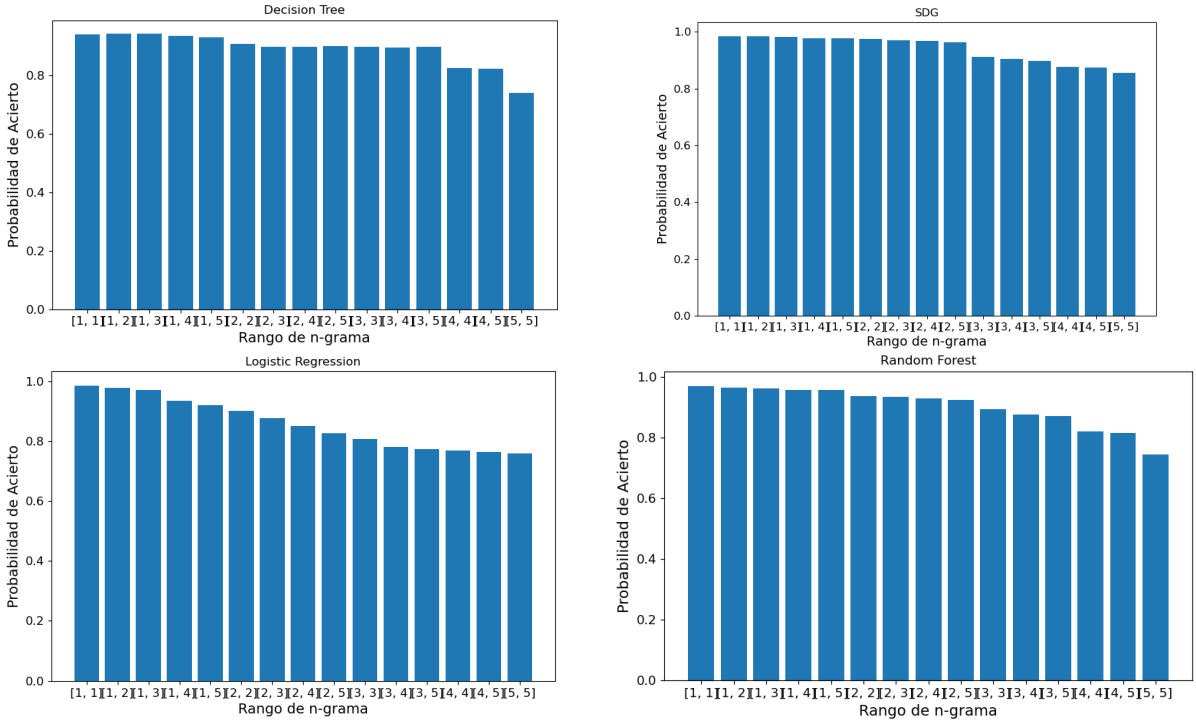

Figure 4.10: Accuracy depending on N-Gram Range

It is clear that the bigger the n-gram range used, the worse it performs. This can be due to the fact that the datasets used are fairly small for NLP and the bigger the n-grams, the lower probability of appearance in another text. This means that for this model we will use unigrams, achieving better results.

4.4.2.1 Dataset 1

Using the TFIDF Vectorizer on the first dataset, the four algorithms perform quite well, having as the best candidate the logistic regression one, with a 98.16 % success rate, while

having 0.99 and 0.97 for the f1 values.

ML Algorithm	Success Rate
Decision Tree	93.43 %
SDG	95.27 %
Logistic Regression	98.16 %
Random Forest	97.39 %

In the Fig. 4.11, we can see from the f1 value that Logistic Regression performs the best for ham and classifying spam.

Decision Tree

	Precision	Recall	f1
0	0.95	0.95	0.95
1	0.89	0.89	0.89

SGDC

	Precision	Recall	f1
0	0.94	1.00	0.97
1	0.99	0.85	0.92

Logistic Regression

	Precision	Recall	f1
0	0.98	0.99	0.99
1	0.97	0.96	0.97

Random Forest

	Precision	Recall	f1
0	0.98	0.98	0.98
1	0.96	0.95	0.96

Figure 4.11: Precision and Recall Matrices for Model 2.1 with TFIDF

With the Count Vectorizer, the performance decreases, having the Random Forest as the best performing.

ML Algorithm	Success Rate
Decision Tree	92.85 %
SDG	94.30 %
Logistic Regression	95.27 %
Random Forest	97.5 %

In the Fig. 4.12, we can see from the f1 value that Random Forest is the most suitable.

Decision Tree

	Precision	Recall	f1
0	0.95	0.95	0.95
1	0.88	0.88	0.88

SGDC

	Precision	Recall	f1
0	0.93	0.99	0.96
1	0.98	0.83	0.90

Logistic Regression

	Precision	Recall	f1
0	0.99	0.94	0.97
1	0.88	0.98	0.93

Random Forest

	Precision	Recall	f1
0	0.98	0.98	0.98
1	0.96	0.95	0.96

Figure 4.12: Precision and Recall Matrices for Model 2.1 with Count Vectorizer

4.4.2.2 Dataset 2

For the TFIDF Vectorizer, the algorithm that works the best in this case is the Logistic Regression (LR), reaching 99.17 % of correctly classified emails. Even though LR has the higher success rate, the other algorithms also perform very well.

ML Algorithm	Success Rate
Decision Tree	96.89 %
SDG	98.56 %
Logistic Regression	99.17 %
Random Forest	98.94 %

In the Fig. 4.13, we can see from the f1 value that Logistic Regression and Random Forest have the same performance.

Decision Tree

	Precision	Recall	f1
0	0.98	0.96	0.97
1	0.96	0.98	0.97

SGDC

	Precision	Recall	f1
0	1.00	0.97	0.99
1	0.97	1.00	0.98

Logistic Regression

	Precision	Recall	f1
0	0.99	0.99	0.99
1	0.99	0.99	0.99

Random Forest

	Precision	Recall	f1
0	0.99	0.99	0.99
1	0.99	0.99	0.99

Figure 4.13: Precision and Recall Matrices for Model 2.2 with TFIDF

The Count Vectorizer has a decrease in performance, reducing the success rate of the SDG, LR and Random Forest by 1%, and the Decision Tree by around 6%.

ML Algorithm	Success Rate
Decision Tree	92.79 %
SDG	98.79 %
Logistic Regression	98.71 %
Random Forest	98.79 %

Seeing the Fig. 4.14, we can observe that Random Forest, LR and SGDC have the same f1 values.

Decision Tree

	Precision	Recall	f1
0	0.94	0.93	0.93
1	0.92	0.93	0.92

SGDC

	Precision	Recall	f1
0	0.99	0.99	0.99
1	0.99	0.99	0.99

Logistic Regression

	Precision	Recall	f1
0	0.99	0.99	0.99
1	0.99	0.99	0.99

Random Forest

	Precision	Recall	f1
0	0.99	0.98	0.99
1	0.98	0.99	0.99

Figure 4.14: Precision and Recall Matrices for Model 2.2 with Count Vectorizer

4.4.3 Model 3: Model 1 + Model 2

In this subsection, we will be studying what happens when we combine Models 1 and 2. For Model 2 we will be using TFIDF with unigrams, as it outperformed the Count Vectorizer. Afterwards, we will evaluate if it made the results improve and why.

4.4.3.1 Dataset 1

With the ham/spam dataset, combining the two models produces a decrease in the performance compared to the TFIDF model alone. The Decision Tree, SDG and Random Forest decrease, and the Logistic Regression remains the same.

ML Algorithm	Success Rate
Decision Tree	92.56 %
SDG	94.49 %
Logistic Regression	98.16 %
Random Forest	97.29 %

As it can be seen in Fig. 4.16, Logistic Regression has the highest f1 values.

Decision Tree

	Precision	Recall	f1
0	0.95	0.94	0.95
1	0.87	0.88	0.88

SGDC

	Precision	Recall	f1
0	0.93	1.00	0.96
1	0.99	0.83	0.90

Logistic Regression

	Precision	Recall	f1
0	0.98	0.99	0.99
1	0.97	0.96	0.97

Random Forest

	Precision	Recall	f1
0	0.98	0.98	0.98
1	0.95	0.95	0.95

Figure 4.15: Precision and Recall Matrices for Model 3.1

4.4.3.2 Dataset 2

In this case, the combination of both models does improve the success rate of all algorithms, except for the Random Forest, that decreased by 2%.

ML Algorithm	Success Rate
Decision Tree	94.76 %
SDG	98.94 %
Logistic Regression	99.32 %
Random Forest	99.24 %

In this case, in Fig. **??**, Random Forest, LR and SGDC have the highest f1 values, with 0.99.

Decision Tree

	Precision	Recall	f1
0	0.97	0.93	0.95
1	0.93	0.96	0.94

SGDC

	Precision	Recall	f1
0	1.00	0.98	0.99
1	0.98	1.00	0.99

Logistic Regression

	Precision	Recall	f1
0	0.99	1.00	0.99
1	1.00	0.99	0.99

Random Forest

	Precision	Recall	f1
0	0.99	1.00	0.99
1	1.00	0.99	0.99

Figure 4.16: Precision and Recall Matrices for Model 3.1

4.4.4 Model 4: Doc2Vec

The last model studied is the Doc2Vec, which is based on word embeddings to create vectors out of each document, in this case, from each email. This model is the one that takes the longest to train, due to the fact that it is necessary to build a vocabulary first for it to work.

4.4.4.1 Dataset 1

The results of this model are notably worse than in models 2 and 3, but are still much better than using word count and sentiment. The best performing algorithm is Random Forest with a 91.5% of success rate.

ML Algorithm	Success Rate
Decision Tree	77.34 %
SDG	87.24%
Logistic Regression	88.48 %
Random Forest	91.5 %

Fig. 4.17 confirms that Random Forest is the ML algorithm that works best in this case.

Decision Tree

	Precision	Recall	f1
0	0.84	0.85	0.85
1	0.63	0.62	0.63

SGDC

	Precision	Recall	f1
0	0.90	0.94	0.92
1	0.84	0.76	0.79

Logistic Regression

	Precision	Recall	f1
0	0.91	0.93	0.92
1	0.82	0.79	0.81

Random Forest

	Precision	Recall	f1
0	0.92	0.97	0.94
1	0.92	0.79	0.85

Figure 4.17: Precision and Recall Matrices for Model 4.1

4.4.4.2 Dataset 2

This model has the Random Forest algorithm as the best performing one with 98.42 %. Even though it performs better than Model 1, Models 2 and 3 have a higher probability of classifying fraudulent emails correctly.

ML Algorithm	Success Rate
Decision Tree	91.38 %
SDG	94.41 %
Logistic Regression	96.05 %
Random Forest	98.42 %

In this case, Random Forest is the more suitable algorithm, as seen in Fig. 4.18

Decision Tree

	Precision	Recall	f1
0	0.91	0.88	0.90
1	0.87	0.89	0.88

SGDC

	Precision	Recall	f1
0	0.95	0.95	0.95
1	0.94	0.94	0.94

Logistic Regression

	Precision	Recall	f1
0	0.94	0.98	0.96
1	0.97	0.93	0.95

Random Forest

	Precision	Recall	f1
0	0.97	0.98	0.97
1	0.98	0.96	0.97

Figure 4.18: Precision and Recall Matrices for Model 4.2

4.5 Final Conclusions

As it was established in the introduction of this section, the characteristic used as a measurement of success in this project is the model accuracy , but also considering the f1 values. For that reason we can say that the models that worked the best were:

- For Ham/Spam Classification: TFIDF and TFIDF combined with word count and sentiment with 98.16% success rate.
- For Ham/Fraud Classification: The TFIDF model combined with word count and sentiment had the best results with 99.32% success rate.

This does not mean that we should instantly discard the other models, it could be interesting to test them in a real environment to analyse how they adapt.

Now we will try to see why some of the models might have failed:

- Model 1: It is a very basic approach to the problem and the results where better than expected initially. In spite of that, it was not enough to characterize the problem as well as the other models.
- Model 3: For Dataset 1, even though it has the same accuracy than Model 2 with TFIDF alone, the word count and sentiment are only adding noise, and in the best case the probability remains the same than in Model 2.
- Model 4: As it was mentioned before, the Doc2Vec needs to have a vocabulary built before training the algorithm. This means that if the dataset is not big enough, it will

perform poorly. The datasets used are relatively small, especially Dataset 1, which explains why the success rate was lower than in models 2 and 3.

CHAPTER 5

Conclusions

5.1 Conclusions

Human interactions and language are complex processes that with current technology are extremely hard to model effectively, and even harder to predict. This project has focused on examining the performance of a set of widely used methods in the field of Natural Language Processing, applying them for spam and fraudulent detection. This chapter is dedicated to discuss the conclusions that can be drawn from the results.

An important takeaway has been how challenging it is to tackle an open ended problem. First, analysing the problem as a whole, doing the necessary reading and background research. Then, laying out the key components that form the problem in order to consider the optimal solutions to implement. Finally, analysing critically a set of results drawing reasonable conclusions.

The spam detection problem is a well studied topic, with a great amount of papers on it. Doing a deep literature search has helped me in order to understand how problems are dealt with in this field. However, it is an ongowing issue that is still trying to be solved in more efficient ways.

It was crucial learning about how important a good dataset is and the different data acquisition methods, such as web scraping or email account creation for spam gathering. Although, for this specific project, the datasets used were open source, there was still cleaning and adaptation to be done.

Another thing that can be extracted from this work, is learning the basics of Machine Learning and NLP, for instance:

- What supervised and unsupervised ML models are.
- How different ML algorithms work.
- Different NLP approaches.
- The basic NLP pipeline.
- The different NLP models, as well as their advantages and disadvantages.
- Representing and evaluating different models

One of the most important things to highlight from this project, is the challenge of building a workflow in order to develop a functional program from start to end. In this process, it is where most work has been done and knowledge was gained. Constant reading, and trial end error was required for successfully completing the task.

5.2 Achieved Objectives

This section will review if the objectives stated in the Introduction have been achieved:

- Feature extraction was successfully achieved, performing it with 4 different models, understanding each part necessary to make them work.
- By creating the application, I had to acquire the knowledge necessary to understand the fundamentals of NLP and how to use many libraries to carry out NLP algorithms.
- The analysis and comparison of results was done in the evaluation section, specifying which of them worked best.
- The final program combines NLP and ML algorithms achieving email classification, detecting if they are ham, spam or of malicious intent.

5.3 Problems Faced

In this project there were many problems along the way, being key doing the research and investigation necessary to solve them. The main problems were:

- Finding a suitable dataset: at the start of the project, the dataset chosen was not suitable for this task. The fields were varying from email to email, and many of the texts were merged which made it impossible to get enough samples. The dataset used in the end was very well separated and labelled, however, it did not have many fields that could have been relevant such as sender, receiver, etc.

- Learning Machine Learning, Natural Language Processing and Python from scratch: when the project was planned, I had no knowledge of either of those topics, so it was a challenge to balance learning all of them at the same time.

- Organising all the models in the program, for efficient evaluation and graphic representation was quite tedious and caused many errors. However, it payed off in the end by simplifying the process of comparing every model making it faster.

5.4 Future work

The project achieved the initial goal of detecting spam and malicious emails. For this, we considered different models, some of them reaching 99% of accuracy. Even though this task was accomplished successfully, there is still much work that can be done to improve.

A crucial aspect to take into consideration for further work is the resources needed to train some models, achieving good results. They not only do they require a lot of time to train, but also, they need large datasets to perform effectively.

Something considered at the start of the planning, was using the email sender address and receiver, as well as analysing if there were files attached, but it was not possible due to the bad quality of most datasets found. Another thing that could be of great interest, is making the program detect more types of malicious emails such as phishing or spoofing.

Another point to improve, is the number of models used. A big part of projects in this field has to be dedicated to testing different models and Machine Learning algorithms. That is why the models used in this particular project can be fine tuned, finding the best hyperparameters with tools like GridSearch, to optimize the results.

Furthermore, is necessary to test the program in a real environment to see how well it performs outside of the testing dataset, seeing if there was oferfitting done when testing it in the training environment. This can be achieved with a new dataset or for instance, making it an add-on for an email service application.

The code for the application can be found at: "https://github.com/gCanete/Email-Classification"

APPENDIX A

Impact of this project

In the appendix we will analyse the different impacts in relation to this project, including: social, economic, environmental and ethical.

A.1 Social Impact

Emails are a communication medium very extended in the Internet era, having around 4 billion users worldwide. This project concerns everyone that has an email account due to the fact that almost everyone receives spam sooner or later. There is a wide range of email users, from individual people to big companies. This is why having a good spam detection system can prevent privacy and money loss in many cases, providing security online. Before there were systems for spam detection, the user had to view every email manually to see if it was an important email or just spam.

The main objective of this project is to provide commodity and security when using email accounts, preventing some types of cyberattacks such as fraudulent emails by correctly detecting them.

A.2 Economic Impact

When discussing the economic impact, we have to take into account that cyberattacks can cause companies or individual users to lose money directly, if for example someone leaks confidential information. However, with spam the money loss comes more indirectly, due to the fact that it is not the company the one suffering it, its the employees. Having to find emails in spam piles can seem insignificant, but in the long run the time used for this, is time not used to work, causing money loss.

The use of this kind of systems reduces the risk of opening a malicious email by mistake and can help improving efficiency when using email accounts, having to spend less time to find things.

A.3 Environmental Impact

To study the environmental impact we have to take a look to the equipment required to have an email account and to provide the service.

To possess an email account pretty much any device with Internet access will be able to handle it. This devices require electricity, which needs to be generated, in many cases in very polluting ways. Another drawback is the process and materials needed to build these devices, requiring mining heavy metals, as well as the incorrect disposal of the devices when they end their life cycle. We have mentioned the effect of making electronic devices, but for email services to work, it is necessary to have data centers were information is stored. These have a great environmental impact, as they are buildings full of computers and servers, running uninterrupted.

A.4 Ethical Impact

On the ethical impact, we can see that email filters have the sole purpose of helping people and companies manage their email accounts without receiving intrusive advertisements and malicious emails. Another benefit that can be obtained is the risk reduction of privacy leaking and protection against malware through the email. It can help people who are not accustomed to the use of technology, preventing them from falling into scams or other things.

APPENDIX B

Economic Budget

In the appendix we will analyse the economic budget in relation to this project, including: Project structure, physical resources, human resources and taxes.

B.1 Physical resources

This section is focused on the cost of the resources that made possible this project.

The only paid software used is the OS, which in this case is Windows 10 Home, with a price of 139.99 $.

The hardware is the following:

- CPU: Intel(R) Core(TM) i9-10900KF CPU
- GPU: NVIDIA GeForce RTX 2080 Ti
- RAM: 64 GB
- Storage: 500 GB

The estimated cost of the hardware is around 2,500 $.

B.2 Project structure

The project was planned with a number of tasks in mind, having the days structured as it is described in the table below:

Activity	Days
Researching about the topic	22
Learning Technologies Required (Python, NLP, etc.)	25
Looking for Possible Solutions	15
Planning the Structure	9
Developing the Application	26
Writing the Document	22
Total	119

B.3 Human Resources

The budget required to cover the cost of human resources will be for one person, considering the project has been made individually.

Estimating a salary of 1,500 € monthly, that would be 50 € per day. Knowing that the project took 119 days it would have a cost of 5,950 € (without considering taxes).

B.4 Taxes

If the product was selled it would be subjected to the Value-Added Tax in Spain, that is 21 % of the product value.

Bibliography

[1] N-gram Language models, April 2020. https://web.stanford.edu/~jurafsky/slp3/3.pdf.

[2] Number of Email Users Worldwide 2021/2022: Demographics & Predictions, February 2020.

[3] Number of Email Users Worldwide 2021/2022: Demographics & Predictions, February 2020.

[4] Vector Semantic and Embeddings, April 2020. https://web.stanford.edu/~jurafsky/slp3/6.pdf.

[5] Simon Baker, Anna Korhonen, and Sampo Pyysalo. Cancer Hallmark Text Classification Using Convolutional Neural Networks. page 9.

[6] Aniruddha Bhandari. Feature Scaling | Standardization Vs Normalization, April 2020.

[7] Fabio Ciravegna. Adaptive Information Extraction from Text by Rule Induction and Generalisation. page 6.

[8] Michael DelSole. What is One Hot Encoding and How to Do It, April 2018.

[9] Andrea Ferrario and Mara Naegelin. The Art of Natural Language Processing: Classical, Modern and Contemporary Approaches to Text Document Classification. *SSRN Electronic Journal*, 2020.

[10] Venkatesh Garnepudi. Spam Mails Dataset, 2019. CLAIR collection of fraud email, ACL Data and Code Repository, ADCR2008T001, https://www.kaggle.com/venky73/spam-mails-dataset.

[11] Aurelien Geron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition [Book]. ISBN: 9781492032649.

[12] Simon Heron. Technologies for spam detection. *Network Security*, 2009(1):11–15, January 2009.

[13] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[14] Kithsiri Jayakodi, Madhushi Bandara, Indika Perera, and Dulani Meedeniya. WordNet and Cosine Similarity based Classifier of Exam Questions using Bloom's Taxonomy. *International Journal of Emerging Technologies in Learning (iJET)*, 11(04):142, April 2016.

[15] Joseph Johnson. Spam statistics: spam e-mail traffic share 2019.

[16] Chaitanya Krishna Kasaraneni. Understanding NLP Pipeline, June 2020.

[17] Muhammad Taimoor Khan, Mehr Durrani, Armughan Ali, Irum Inayat, Shehzad Khalid, and Kamran Habib Khan. Sentiment analysis and the complex natural language. *Complex Adaptive Systems Modeling*, 4(1):2, February 2016.

[18] Marcus Kracht. Introduction to Linguistics. page 211.

[19] Alan Mathison Turing. Computing Machinery and Intelligence. page 22.

[20] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

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

[22] Daniel Bastos Pereira and Ivandré Paraboni. A Language Modelling Tool for Statistical NLP. page 10.

[23] D. Radev. Fraudulent E-mail Corpus. `https://www.kaggle.com/rtatman/fraudulent-email-corpus`.

[24] Sam Scott and Stan Matwin. Feature Engineering for Text Classification. page 10.

[25] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature Selection for Classification: A Review. page 33.

[26] Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, and Harshit Surana. Practical Natural Language Processing [Book]. ISBN: 9781492054054.

[27] Lowri Williams. Adversarial Attacks on SMS Spam Detectors, January 2021.

[28] Mi ZhiWei, Manmeet Mahinderjit Singh, and Zarul Fitri Zaaba. EMAIL SPAM DETECTION: A METHOD OF METACLASSIFIERS STACKING. (200):8, 2017.