**UNIVERSIDAD POLITÉCNICA DE MADRID**

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN


MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

**MEDIA DATA SYSTEMS DESIGN**

# P4: Sentiment Analysis and Fake News Detection

MARÍA VILLA MONEDERO

May 2024

# 1 Introduction

In the following report, we will discuss and analyze the proposed code along with the various results obtained in two areas. First, we will examine the code related to Sentiment Analysis on Amazon Reviews, followed by an analysis of Fake News Detection.

# 2 Sentiment Analysis

In the following section, we will analyze the different codes proposed for conducting sentiment analysis on Amazon reviews. To do this, we will structure this section according to the various scripts.

1. Data cleaning and feature extraction
2. Amazon review data visualization
3. Sentiment Analysis in Amazon reviews
4. Topic Modeling in Amazon reviews
5. Creation of the prediction model:

## 2.1 Data Cleaning and Feature Extraction

The provided code for this section performs data cleaning and basic feature extraction on a dataset containing text reviews. The objective is to prepare the data for further analysis or modeling tasks by removing noise and extracting relevant features.

#### Data Loading and Exploration

This code begins with loading the dataset using pandas *read_csv* function. By displaying the five first roads of the dataset we observe that we have a dataset of 10 columns, like the ProductId, the UserId, the Summary and the Text.

The next step is to identify and handle the missing values in the dataset. The function *isna()* is used to check for missing values, followed by the *sum()* function to calculate the total number of missing values for each columns. By doing this we obtain that for the ProfileName column we have 26 missing values and for the Summary column we have 27 missing values. At last, the missing values are dropped, using the *dropna()* function.

#### Features Extraction Before Data Cleaning

Thanks to the *nltk* library we start by calculating the number of **stopwords**, **punctuation marks**, **hashtags**, **numeric characters** and **uppercase characters**. For example, for the first review in the dataset we obtain there are 21 stopwords, 3 punctuation characters, 0 hashtags, 0 numeric characters and 1 uppercase word.

Once the feature extraction is done we proceed to delete this type of characters, and also emojis and urls from the text reviews. And we also proceed to check the spelling and correct it from each review.

In order to achieve this we use the *textblob* library. We first apply a first round of cleaning techniques, which include making the text lowercase, removing the text in square brackets, the punctuation and words containing numbers. At last, a second round of cleaning techniques is applied, getting rid of some additional punctuation and non-sensical text.

This is very useful as it will help us to reduce multiple copies of words. For example the most repeated word is 'like' appearing 251.302 times.

#### Features Extraction After Data Cleaning

Once we have extracted some features before the data cleaning, we proceed to extract some features after the cleaning process.

We first start by calculating the **word count** for each text review using the *split()* function to separate words based on spaces, and we store the result in a new column named *word_count*. Following this, we calculate the **character count**, including spaces using the *str.len()* function and then store the result in a new column.

Subsequently we calculate the **average word length** for each text review by dividing the total character count by the number of words. For example, for the first review the average word length is of 6.1.

Having done this, we proceed to the data cleaning and manipulation. We first start by converting the **Time column** into datetime format using the *pd.to_datetime()* function. Secondly, we drop the **ProfileName** columns because it is not relevant as we already have the UserId column.

Similarly to the previous section we proceed to do two rounds of text cleaning and additional stopwords are added to the stopwords list and removed from the text reviews. Additionally, we perform string manipulation to remove specific endings from text reviews, in order to eliminate noise or artifacts from the reviews.

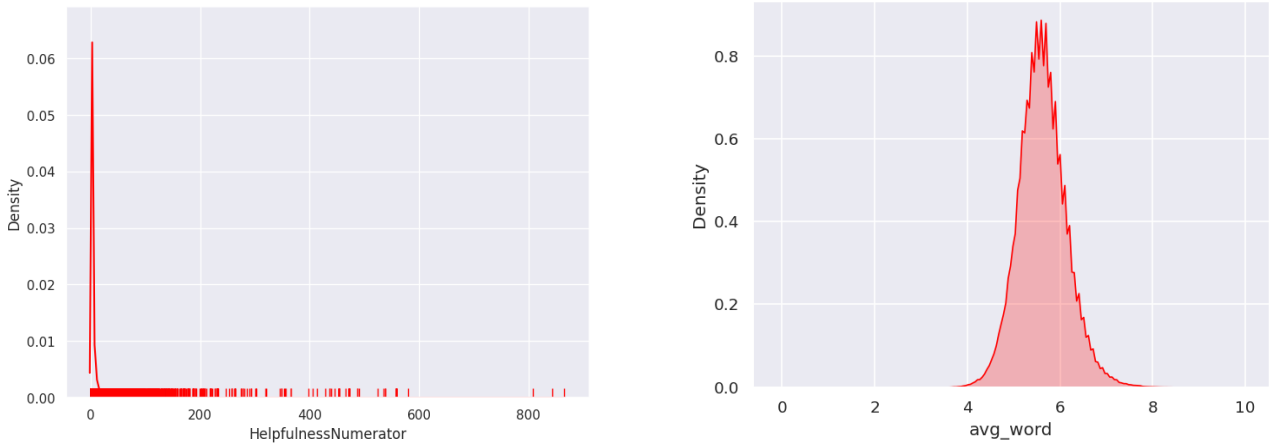
At last we identify text reviews with a high punctuation and the most frequent words in the text reviews. With this we can analyse common themes or topics mentioned in the reviews. For example one of the most common topics is *coffee*. We finish this section by saving the processed dataset.

## 2.2 Amazon Review Data Visualization

The provided code for this section focuses on visualising various aspects of the cleaned Amazon review data to observe the distribution and characteristics of the reviews.

### Distribution Analysis and Filtering Outliers

A density plot using *seaborn* is used to visualize the distribution of the **HelpfulnessNumerator** feature. The density plot shows that there are outliers values, indicating potential issues with repetitive or spam-like reviews.




(a) Density Plot for HelpfulnessNumerator Feature

(b) Density Plot for Average Word Length

Figure 1: Density Plots

There are serious number of repetitions in reviews, this is why we drop the reviews with HelpfulnessNumerator values exceeding 100 to remove the outliers and to maintain data integrity.

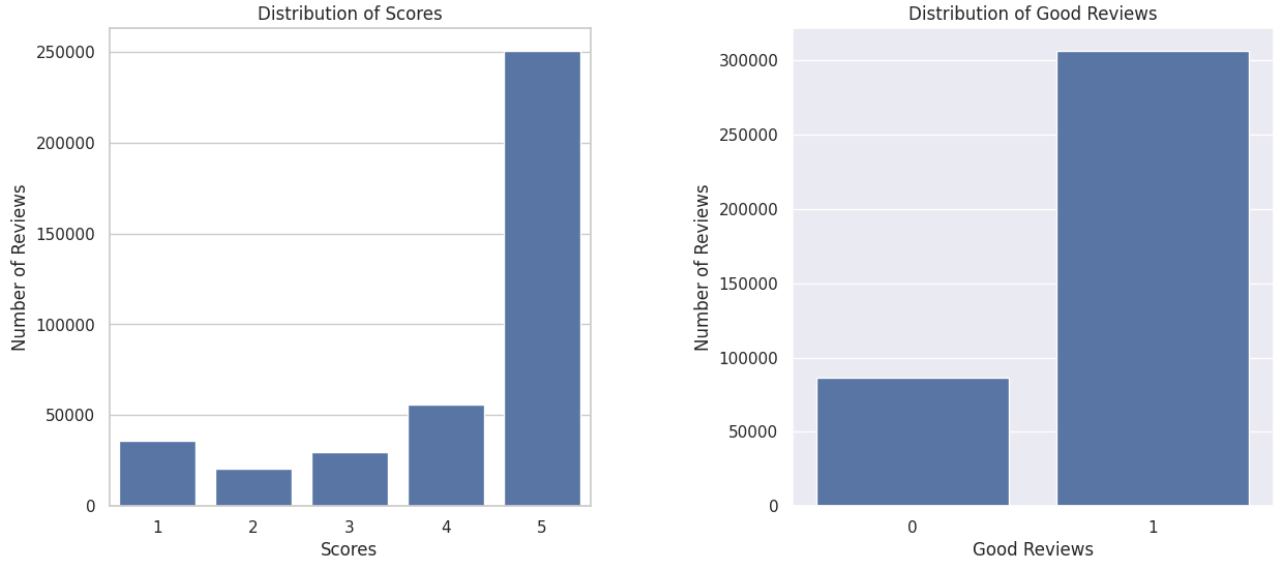
Similar distribution analyses are performed for other features such as **HelpfulnessDenominator**, **stopwords count**, **punctuation count**, **word count**, **character count**, **average word length**, **distribution of upper case characters**. Outliers are then identified and filtered out where is necessary, to ensure data quality. For example Figure 1b shows the distribution of average word length.

During this analysis we have to keep in mind that short reviews can be same coincidentally. So we check the number of reviews that have 3 and lower number of words and check if there are identical reviews, which is not the case.

After all the cleaning we obtain a dataset of 16 columns and 392.714 entries.

**Feature Analysis. Score Distribution**

Following this we continue with the analysis of **scores distribution**. We generate bar plots to illustrate the distribution, where we observe that most of the reviews are 5 starts, which means that we have an unbalances distribution. In an attempt to solve this we convert the *Score* feature into a binary feature, where score values 1, 2 and 3 will be coded as 0, and values 4 and 5 will be coded as 1.




(a) Bar Plot for Score Feature

(b) Bar Plot for Binary Score Feature

Figure 2: Bar Plots for Score Feature

**Sentiment Analysis**

Having done that we are going to check if there is some kind of relationship between all the features and the score feature. In order to do this we plot the number of words per reviews taking into account the scores, the number of upper case characters for reviews, the HelpfulnessNumerator and the average world length vs Good Reviews.

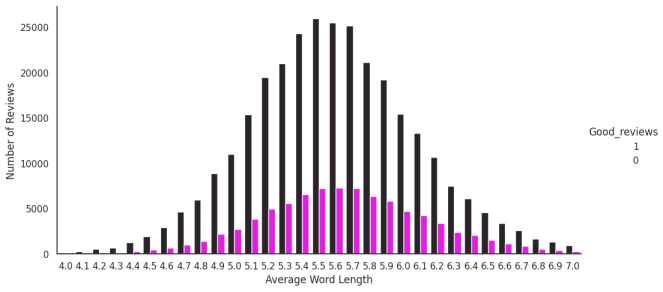



Figure 3: Bar Plot for Average Word Length of the Good Reviews

At last we export the cleaned and visualized dataset.

## 2.3 Sentiment Analysis in Amazon Reviews

The provided code for this section performs sentiment analysis on the Amazon reviews, calculating polarity and subjectivity scores using **textblob**.

**Creating *Subjectivity* and *Polarity* Scores**

Each review's polarity and subjectivity were computed and rounded. Then the averages of these scores were calculated based on different review ratings (1 to 5 stars) and whether the reviews were considered *Good* or *Bad*.

We then use scatter plots to visualize the relationship between polarity and subjectivity across different review scores.

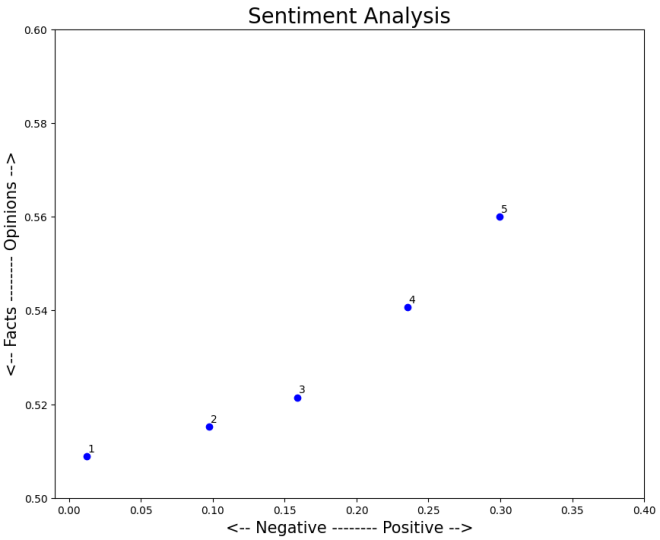



Figure 4: Scatter Plot for relationship between Polarity and Subjectivity

We also programmed a histogram plot to show the distribution of polarity scores, which show variability within good and bad reviews.

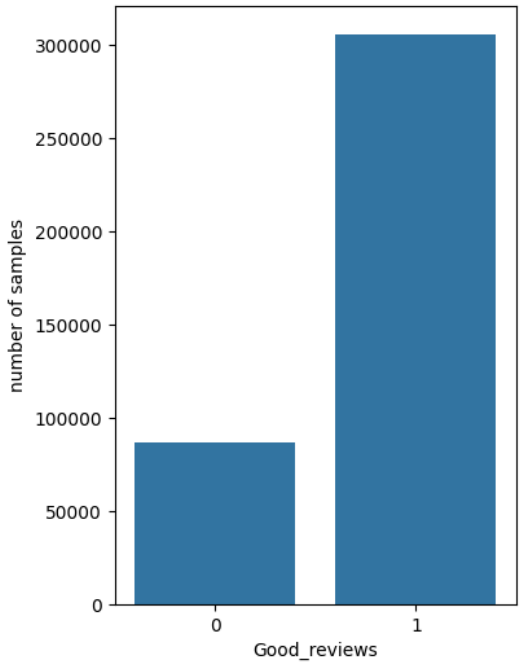
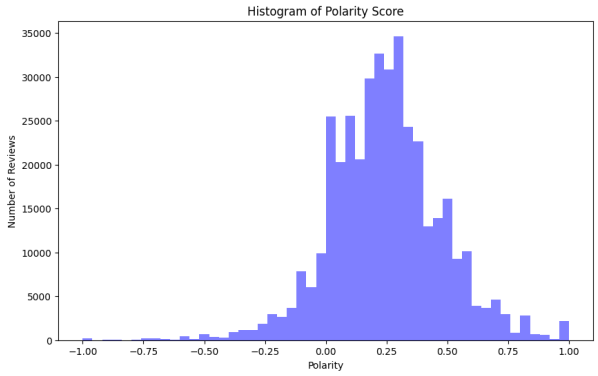



Figure 5: Bar Plot of Polarity Scores

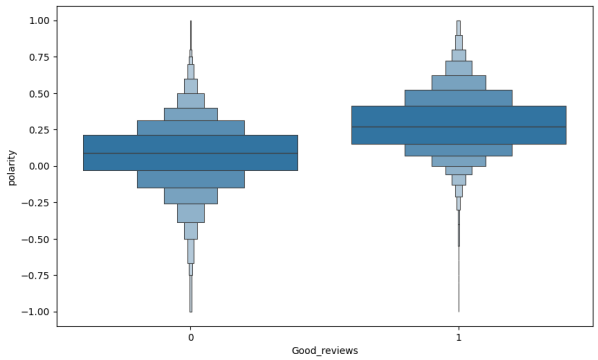
As you can see the data is unbalanced.

### Polarity Analysis

First we present the histogram and the box plot where we can see that some good reviews have very low polarity and some bad reviews have high polarity, which is something we should check.




(a) Histogram of Polarity Score




(b) Box Plot of Polarity Score

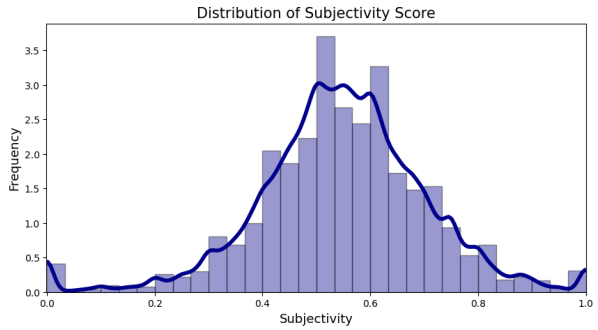
Figure 6: Plots for Polarity Score

The reason why there are good reviews but highly negative is because they are sarcastic reviews, which cause a scoring problem for probability.

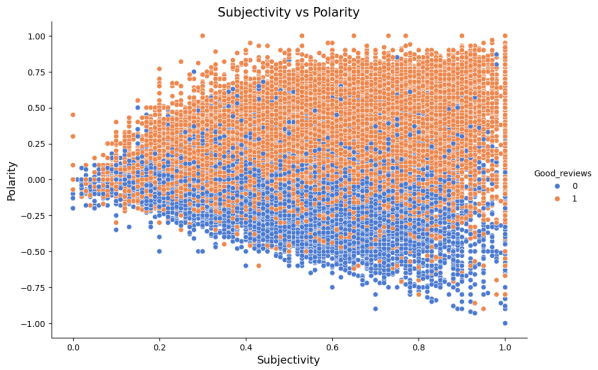
In order to conduct the polarity analysis we also examined the relationship between the polarity scores and factors such as the number of words in a review and the helpfluness of the review.

### Subjectivity Analysis

Similarly to the previous section we are going to plot the distribution of the subjectivity score and a relational plot.




(a) Histogram of Subjectivity Score




(b) Relational Plot of Subjectivity Score

Figure 7: Plots for Subjectivity Score

As it is shown, most reviews are moderately subjective. We also analyse the number of words per review based on the subjective score and conclude that good reviews tend to have a higher number of words and higher subjectivity scores compared to bad reviews.

## 2.4 Topic Modeling in Amazon Reviews

The next section is about topic modeling techniques, where the objective is to find various topics that are present in the reviews. We sill be using the Latent Dirichlet Allocation approach, which it will help us to identify and interpret the main topics present.

In order to achieve this we will create a Document-Term Matrix using the *CountVectorizer* and we will tokenize the text to focus on nous and adjectives for more meaningful topic extraction. At last we will apply the LDA model.

#### Creating Document-Term Matrix for Good Reviews

We filter the dataset for good reviews, and then create a document-term matrix. This matrix captures the frequency of each word in the reviews. Subsequently we converted the DTM into a format suitable for LDA using *gensim* and *scipy*. Then the LDA model was then applied to extract topics, initially with 2 topics and 10 passes. As a result we have keywords like: *coffee*, *like*, *flavour*, etc. and their respective weights within the given topic.

#### Refining Topic Modeling with Nouns Only for Good Reviews

We refined the topic modeling by focusing on nouns only, which often carry more meaningful content. A new DTM was created using only nouns, and the LDA model was applied again. If we compare the output results with the previous section we find that both techniques include discussions about food and beverages. But it is true that with the previous techniques the topic are less specific, like for example we extract a topic that only focus on general products including words like: *product*, *amazon*, *great*.

#### Refining Topic Modeling with Nouns and Adjectives for Good Reviews

In our third attempt we refined the topic modeling by focusing on nouns and adjectives. And when comparing the output results it is similar to the first section were we didn't focus on nouns. For example, in one topic the keywords included are: *great*, *product*, *good*, which means is too broad.

In conclusion when Refining Topic Modeling for Good Reviews we can conclude that it is more useful to focus on nouns, as we identify more specific topics such as *cookies and snacks*, *drinks*, *pet items*, etc.

#### Refining Topic Modeling for Bad Reviews

We apply the same strategy, explained in the previous sections for the Bad Reviews, and we conclude that focusing on nouns or adjectives clarified topics like *coffee* and *tea*. We also observed that the primary topics were about *service complaints* and *pet items*.

### 2.5 Creation of the Prediction Model

This section involves preparing the data, splitting it into training and testing sets, and then fitting machine learning models to predict sentiment classification based on Amazon reviews.

#### Creating a Document Term Matrix with TF-IDF

We first start by splitting the data into training and testing. Where the training set consist of 263.118 samples and the testing set of 129.596 samples. Subsequently we convert the reviews to a matrix of TF-IDF features using the function *TfidVectorizer* from sklearn.

#### Model Fitting and Evaluation

Firstly, the **Gaussian Naive Model** is fitted to the training set and evaluated using metrics like accuracy, precision, recall and F1-Score. Obtaining a F1-Score of 0.76%.

Similarly, we also evaluate and plot the fitted **Multinomial Naive Bayes Model**, the **Bernoulli Naive Bayes** and the **Logistic Regression** models.

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Gaussian Naive Model | 0.76 | 0.93 | 0.75 | 0.76 |
| Multinomial Bayes Model | 0.81 | 0.81 | 0.99 | 0.81 |
| Bernoulli Naive Bayes | 0.82 | 0.88 | 0.89 | 0.82 |
| Logistic Regression | 0.86 | 0.89 | 0.95 | 0.86 |

In conclusion the highest F1-Score, is obtained with the Logistic Regression Model.

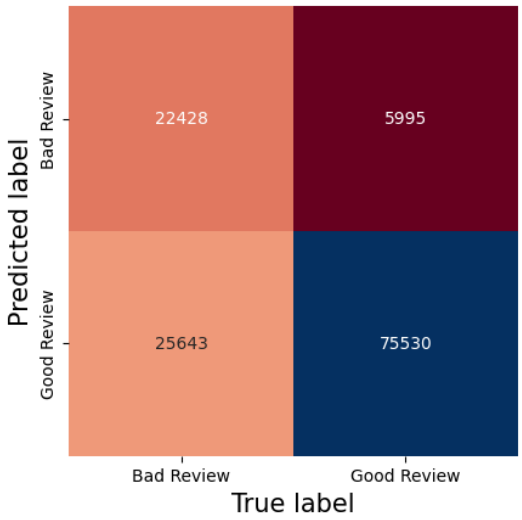
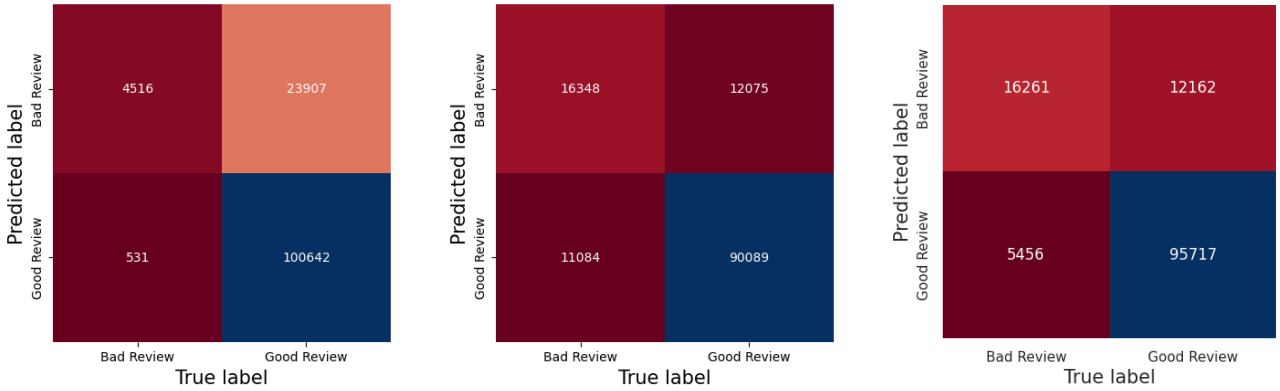



Figure 8: Confussion Matrix of de Gaussian Naive Model




(a) CM of Multinomial Naive Bayes (b) CM of Bernoulli Naive Bayes (c) CM for the Logistic Regression

Figure 9: Confusion Matrix of the different Models

### Receiver Operating Characteristic (ROC) Curve

ROC Curves are plotted for each of the previous models to visualize their performance, obtaining the following a score for the area under the curve.

| **Model** | **AUC** |
|---|---|
| Gaussian Naive Model | 0.83 |
| Multinomial Bayes Model | 0.88 |
| Bernoulli Naive Bayes | 0.86 |
| Logistic Regression | 0.90 |

The ROC Curves reveal different performance levels depending of the model under evaluation. The Logistic Regression model demonstrates the highest performance. Multinomial also performs well, closely following Logistic Regression, and are good alternatives when handling imbalanced datasets. Bernoulli Naive Bayes performs reasonably but slightly worse than the previously mentioned models. At last, the Gaussian Naive Bayes shows the poorest performance, this might be because the data do not follow a Gaussian distribution.

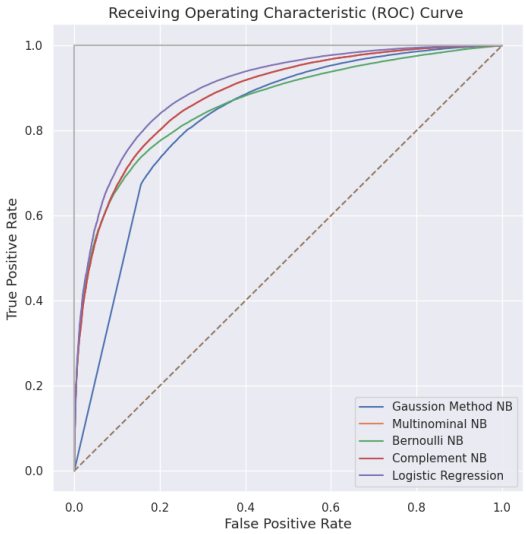



Figure 10: ROC Curves for the models

### Sanity Check

At last, some examples are provided to check the model's predictions against the actual reviews. As you can see in most cases the model performs correctly.

| | actual_label | prediction | review |
|---|---|---|---|
| **121792** | 1 | 1 | tea great shipping fastexcelent service |
| **123456** | 0 | 0 | purchased gamble product information listed id... |
| **64884** | 1 | 1 | husband likes gf mix uses white bread setting ... |
| **3933** | 1 | 1 | walkers shortbread cookies greatvery fast deli... |
| **12793** | 0 | 0 | excited product read reviews little disappoint... |
| **79901** | 1 | 1 | youre fan russian mustard get make scratch |
| **88280** | 1 | 1 | miniature schnauzers assortment pills take yea... |
| **110033** | 1 | 1 | drink tea daily fifty years first tried barrys... |
| **125642** | 1 | 1 | wanted note fact msg product maltodextrin msg ... |
| **21667** | 0 | 1 | maker diet claims following nrg maxim grainles... |

## 2.6 Conclusions

In conclusion, this report includes the analysis of the Amazon reviews sentiment using machine learning techniques. Where after cleaning, analysing and applying sentiment scores we trained and evaluated different classification models to complete these task.

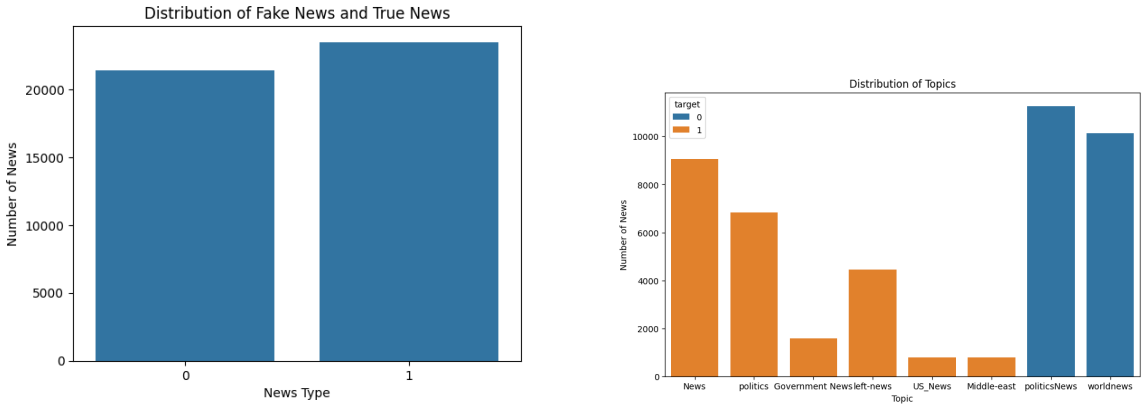
# 3 Fake News Detection

This section of the report aims to explain and comment on the provided code for fake news detection. We will explain the different steps taken in data preparation, exploratory data analysis, data preprocessing, model training, and evaluation.

## 3.1 Exploratory Data Analysis (EDA)

We first start by downloading the necessary libraries such as *spacy* and *textacy*, necessary for text processing. Subsequently we set the *nltk* library in order to download the different datasets for Natural Language Processing tasks, like *punkt*, *wordnet*, etc.

In this report we are going to use two datasets, one containing fake news with 23.481 samples and the other containing true news with 21.417 samples.




(a) Distribution of Fake News and True News

(b) Distribution of Topics

Figure 11: Bar Plots for Fake and True News

As you can see the dataset is relatively balanced since the number of fake news and true news is similar. We also plot the code in order to see the distribution of topics taking into account if they are true or fake news

Once we have stated that the data is well balanced we check if there are null and duplicates values. We identified that although there is no missing value in all columns, 631 rows have empty text in text columns, and that there are 209 duplicates.

### 3.1.1 Word and Text Length Analysis

Common words, excluding stopwords, are identified and visualized using a word cloud for both fake and true news.

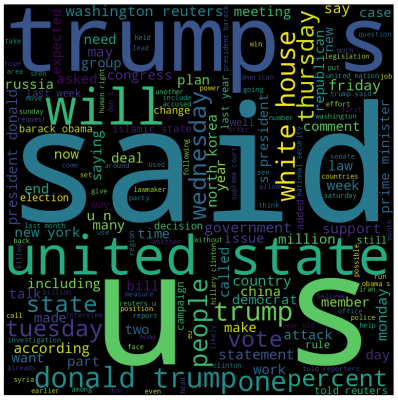
As you can see, Figure 12 in both true and fake news the most common words are almost similar, like *said* and *trump*. It is important to mention that for example the word said appears repeatedly and in different conjugations, in the next section we will solve this problem.

Subsequently we continue with the analysis of the average word length and the number of words in each article as shown in Figure 13

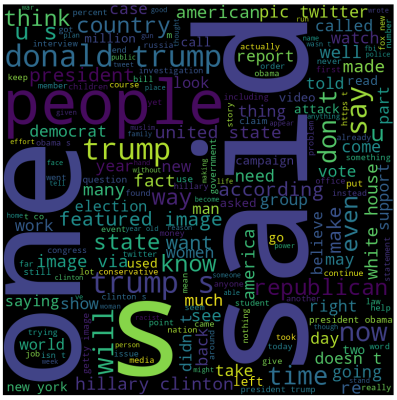
## Data Preprocessing

In this section we are going to apply cleaning strategies to all the news. We first start by removing the duplicated news.

Depending on the source of the news, different cleaning techniques will be applied. For example, for *Reuters*, all news articles without dates will be removed. In this section, we will eliminate URLs, hashtags, photo citations, date mentions, and other symbols from the news using the *textacy* library. We will also remove stopwords and single characters to reduce noise

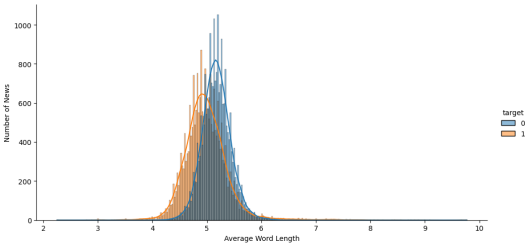


(a) Word Cloud for True News

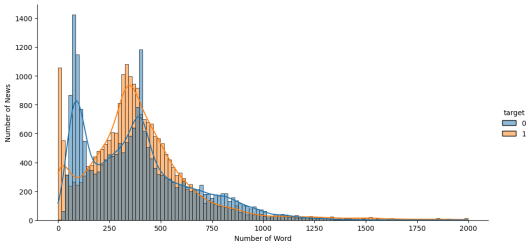


(b) Word Cloud for Fake News

Figure 12: Word Clouds




(a) Distribution of the Average Word Length

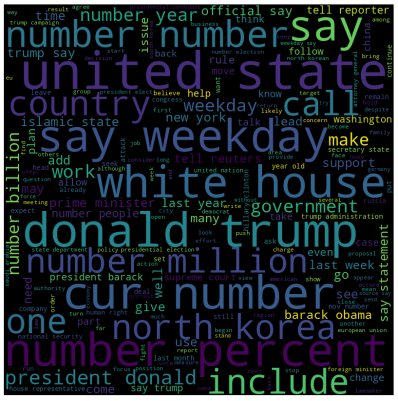



(b) Distribution of the Number of Words

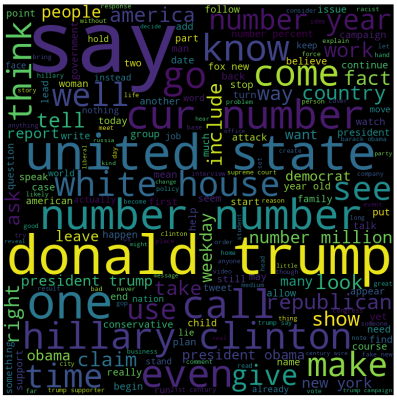
Figure 13: Distributions for Word Count and Length

Once most of the cleaning is done we concatenate the title and the cleaned text in a new column called *summary*. At last, this cleaned text is also **lemmatized** to convert words to their base forms, which helps in reducing the dimensionality of the text data and improving the model's performance.

When plotting the words cloud, Figure 14 ,we notice that is less noise, we have repeated words but we do not have the same verb, for example, with different conjugations.



(a) Word Cloud for True News after Cleaning



(b) Word Cloud for Fake News after Cleaning

Figure 14: Word Clouds after Cleaning

Finally, the cleaned data is save to a CSV file for further use in modeling and analysis.

## 3.2 Modeling

We first start by downloading various vector representations of text data, including unigram and bigram count vectors, TF-IDF vectors, and Word2Vec vectors.

Subsequently multiple models are trained using different vector representations. The models include **Logistic Regression**, **Support Vector Machine**, **Naive Bayes** and **Random Forest Classifier**. This part could not be executed due to the limitations of Google Colab, we also couldn't execute the part of the downloaded models because of the same reason.

## 3.3 Feature Importance with Logistic Regression

At last we use a Logistic Regression Model to find the most contributing tokens to fake and true news. Firstly, we begin by loading the preprocessed unigram token data and splitting it into training and test sets. Subsequently, a Logistic Regression Model is trained on the training set to learn the relationship between the unigram tokens and the target labels. We use a maximum iteration parameter of 500.

Once the training is done we observe the coefficients from the trained model, as they are going to indicate the importance of each token. Positive coefficients suggest a higher contribution towards predicting fake news, while negative coefficients indicate a higher contribution towards predicting true news.

For fake news, several of the tokens are: *breaking*, *gop*, *according*, etc. On the other hand, for true news, the tokens include: *reuters*, *factbox*, *weekday*, etc. These lists of tokens will help us understand the different characteristics of fake and true news in articles.

We also analyse the patterns in order to see how specific phrases are used differently in fake vs true news. This analysis indicates that true news articles tend to refer to presidents by their full names (*President Donald Trump*, *President Barack Obama*), whereas fake news articles are more likely to use just the las names (*President Trump*, *President Obama*).

## 3.4 Conclusions

The provided code effectively preprocess text data, explores various aspects of the data, and applies multiple machine learning models to detect fake news. The EDA and preprocessing steps ensure the data is clean and ready for modeling. The modeling section evaluates different models and vectorizations, providing insight into the most effective approaches for fake news detection. For the fake news varios de los tokens son: *breaking*, *gop*, *according*, etc. On the other hand for the true news, they are: *reuters*, *factbox*, *weekday*, etc. These list of tokens will help us to understand the different characteristics of fake and true news in articles.