# Text Mining Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

## Predict Airbnb Unlisting

Group 20

Carlota Carneiro, number: 20210684

Cláudia Rocha, number: r20191249

Susana Dias, number: 20220198

June, 2023

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

# INDEX

Table of Figures

Table of Tables

## 1. Introduction

This report focuses on creating a predictive model as the final project in the Text Mining course. For the development of the project, a real situation was given, where we had to analyze distinct host and Airbnb property descriptions, as well as reviews from previous guests, and implement an NLP[1] classification model able to predict, for each property, if it was listed or unlisted. In other words, our main concern is answering the question "What properties will be unlisted in next quarter?". In the following pages, we present and discuss the entire process of the project development, including data exploration, preprocessing, feature engineering, models created and major findings.

## 2. Data Exploration

We began by conducting an exploration of our datasets, which consisted of four distinct datasets: *train*, *train_reviews*, *test*, and *test_reviews*. The two datasets regarding reviews contained all the guests' comments made to each Airbnb property, while the other two datasets provided information on the Airbnb descriptions, host details, and the target feature: *Unlisted*, which had a binary representation, with 1 indicating that the property would be unlisted in the next quarter, and 0 indicating that the property would remain listed. It's important to note that the *test* dataset did not include the *Unlisted* feature, as it was the dataset on which we needed to make predictions. In table 1 we can see the distinct features and their data type presented on the train datasets.

| Dataset | Features | Non-Null Count | Data Type |
|---|---|---|---|
| *train* | *description* | 12.496 | Object |
| | *host_about* | | |
| | *unlisted* | | Integer |
| *train_reviews* | *comments* | 721.402 | Object |

Table 1 - Information about the train datasets

Regarding the *train* dataset, we initially observed that the descriptions and host information contained observations in different languages, reflecting the diverse global properties in the dataset. We acknowledged that this could pose challenges, even with translation to English, due to cultural and contextual differences that could impact the text's interpretation and its relevance to the prediction task. We also examined the unique values and descriptive statistics of the *Unlisted* feature. With a mean of approximately 0.28 and a 50% quantile of 0, we observed that there were more properties expected to remain listed (0) than those anticipated to be unlisted (1) in the next quarter. Dealing with imbalanced datasets can pose challenges for machine learning models, as they tend to favour the majority class and provide suboptimal results for the minority class. Therefore, we will further explore various techniques to address this class imbalance issue and improve the performance of our models. For the categorical features, we found that among the 12,496 observations, there were only 4,396 unique host information, suggesting that the same host might possess multiple properties. The most common host information was repeated 304 times, indicating that this particular host potentially owned 304 properties. As for the property *description*, the most frequently occurring description

[1] Natural Language Processing

appeared 50 times, while there were 11,788 unique property descriptions present in the dataset. Apparently, there are no missing values. We identified 996 duplicated rows when checking for duplicates based on the *description* and *host_about* columns. These duplicates indicate instances where the combination of property *description* and *host_about* is identical to other observations in the dataset, revealing data redundancy. Additionally, we employed a language detection algorithm to identify the language in the respective *description*, by adding a column called *description_lang.* English was the most common language, accounting for 81.6% of the descriptions, followed by Portuguese at 14.8% and French at 1.2%.

We proceeded with the exploration of the *train_reviews* dataset, which consists of *comments* as the sole feature, as shown in Table 1, and it contained no missing values. Each property can have multiple comments, while some properties may have no comments at all. Specifically, out of the 12,496 distinct properties in the *train* dataset, only 8,467 had reviews, which accounts for nearly 68% of the properties. Upon examining the descriptive statistics of this dataset, we discovered that the most common comment was "." and appeared in 908 comments. It became apparent that although there were numerous comments, many of them were not valuable for our analysis as they consisted of punctuation, blank spaces, simple letters, and lacked useful information. Consequently, we decided to investigate comments with less than 6 characters to understand their nature and usefulness, as there were a substantial number of such comments. Furthermore, we identified approximately 1,260 comments that solely comprised blank spaces, emojis, punctuation, or non-relevant letters, which were not beneficial for solving the specific problem at hand. We also checked for 580 duplicates, considering that this may be normal, particularly for common comments like "Excellent!" or "Great location," which can be predefined on some devices. Similarly to what we have done before, we applied a language detector to the *comments,* and found that English is the most common language, accounting for 64% of the comments. French follows with 14.9%, and Portuguese accounts for 6.33%.

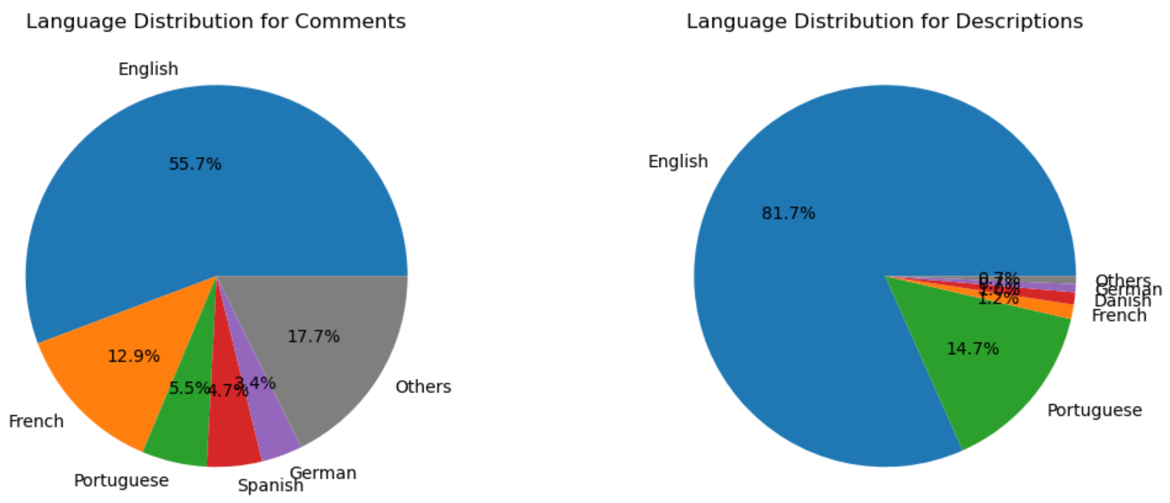


Figure 1 - Pie charts of the Most Common Languages in the Comments and Description Features

In our analysis, we recognize the significance of comments in predicting the *Unlisted* label. However, since not all properties have comments, we acknowledged that relying solely on comments may not be the optimal approach. To address this, we decided to incorporate the *description* feature into the task. Combining these two features, which were initially in separate datasets, into a single dataset would facilitate our exploration. Therefore, we merged the *train* and *train_reviews* datasets to create a *combined_train* dataset, with a size of 721,401 observations and 6 features. In this initial analysis, this dataset contains multiple rows where the *description* and the *host_about* features are repeated, but the comment is different.

Following this, we decided to create some visualizations. Visualizations in text mining are important for exploring and understanding the data, as they help uncover distribution patterns, reveal hidden structures, and identify outliers or anomalies (Hofmann & Chisholm, 2016). These facilitate EDA[2], highlight significant words or phrases, and communicate findings effectively, supporting informed decision-making based on the text data. We began the analysis by examining the distribution of the *Unlisted* feature in the *train* dataset using a bar plot and a pie chart. This revealed that 72.3% (9,033) properties were listed, while 27.7% (3,463) were unlisted. Next, we created word count histograms for the *description* and *host_about* features, which showed that most observations had a range of 125-170 words in the *description* and 0-50 words in the *host_about* feature. We also generated word clouds and histograms for the top 10 words and bigrams in both features. However, these initial results were not very informative - “br”, “the”, “and”, “a”, “b”, “apartment”, and the most commons bigrams are “in the”, “the space” and “of the” - due to the lack of data pre-processing. Lastly, concerning the *train_reviews* dataset, we created a word cloud and frequency histograms for both individual words and bigrams. However, similarly to the *train* dataset, the results were not conclusive, with the most common words being "br," "apartment," and "de" – see Figure 1 - and the prominent bigrams being "in the," "of the," and "a great." Consequently, we intend to repeat these visualizations once pre-processing has been conducted to gain more meaningful insights.

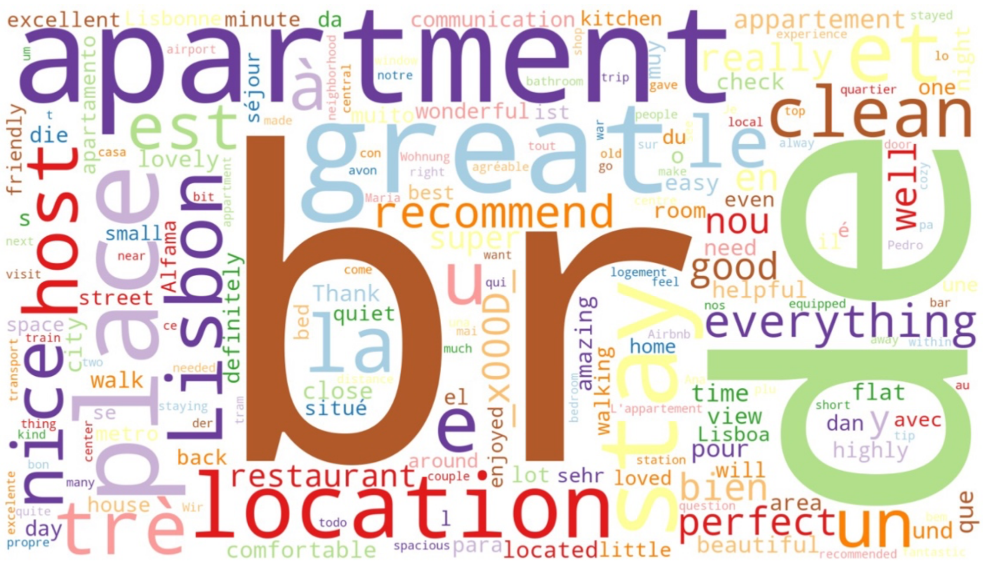


Figure 2 - Comments Feature Word Cloud

[2] Exploratory Data Analysis

## 3. Data Preprocessing

As mentioned in the previous Data Exploration, our team decided to proceed with the *combined* dataset, and build a model using both the *comments* and *description* features. Therefore, in the preprocessing phase, we first removed the *host_about* column, as it is no longer necessary and keeping it would make our dataset heavier.

During our exploration, we encountered comments that contained emojis, as well as comments that were solely emojis. Emojis have become increasingly popular as a means of expressing emotions, enhancing clarity, and achieving brevity, therefore, as an additional task for this project, our group decided to convert these emojis into text.  We believe that translating emojis into text can provide benefits to the training of our models. Before proceeding with the conversion of emojis into text, we first counted the number of emojis present in the *comments* feature of the *combined_train* dataset, which revealed a total of 18,758 recognized emojis. Using the *emoji* library and the *emoji.demojize()* function, we performed the translation. Afterwards, we rechecked the number of emojis in the comments feature and found that there were only 5 remaining. Interestingly, these 5 emojis appeared as empty rectangles, probably indicating that they were unrecognized by the library. Consequently, we made the decision to remove these 5 rows containing unrecognized emojis from the dataset. It is important to note that we only applied this procedure to the *comments* feature.

To ensure proper evaluation of our model's performance on unseen data, we recognized the need to split the *combined_train* dataset into training and validation sets. For this purpose, we employed the *train_test_split* method, with a validation set of 30% of the whole *combined_train* dataset, getting, as a result, four distinct datasets: *X_train, X_val, y_train,* and *y_val.* In the X datasets, we included the *description, description_lang, comments, comments_lang,* and *house_nr* features. We have chosen to retain the *comments_lang* and *description_lang* features for cleaning purposes*.* On the other hand, the y datasets solely comprise the label and the *house_nr* feature. Including the *house_nr* feature in this step helps us retain the house number information throughout the process. Moreover, we set it as an index for both the X and y datasets, ensuring accurate predictions based only on the *comments* and *description* features. It is important to highlight that splitting the *combined_train* dataset at this stage is effectively equivalent to splitting it based on the *comments*. Since each row represents a different comment, the description remains the same for a given property. For a matter of curiosity, we decided to investigate the number of unique *house_nr* values in both *X_train* and *X_val* datasets. Surprisingly, we found a similar count of approximately 8.300 properties in the *X_train* dataset, and around 8000 properties in the *X_val* dataset, despite the *X_train* dataset having 500,000 observations (*comments*) and the *X_val* dataset having around 215.000 observations. This indicates that while the number of properties is similar, the *X_train* dataset contains significantly more comments than the *X_val* dataset. Consequently, the model will be trained on a much larger set of comments compared to the validation dataset. After splitting the data, we proceeded to perform data cleaning operations. Observations with less than three characters were deemed uninformative and subsequently removed, as we saw in the Data Exploration part. Additionally, we decided to keep the NaN comments and replace it with the word “commentless” to account for missing values, given that the absence of comments may have meaning for our predictive modeling.

As an additional aspect of our project, we contemplated translating the comments to a single language, English, the universal language, to facilitate subsequent analysis by machine learning models. For this purpose, we selected a subset of the *X_train* dataset, comprising the first 30 observations. We attempted translation using the *GoogleTranslator* class from the *googletrans* library. However, it was computationally expensive due to the large number of rows our dataset has, prompting us to explore alternative approaches. Another extra task we undertook was implementing a spelling correction technique using the *TextBlob* library. Similarly to before, we applied this technique to a subset of comments. Upon careful examination, we observed that while the original comments were comprehensible, the versions with spelling corrections were not accurate. Consequently, we decided not to proceed with these two additional methods, as they could potentially undermine the performance of our models.

We employed a function from our text mining laboratory classes to perform feature cleaning in our data frames, while incorporating additional tasks. This comprehensive function incorporates language detection to determine the appropriate cleaning steps based on the language of the target feature to be cleaned. The cleaning process involves tokenization, removal of stop words, lowercase conversion, regular expressions, lemmatization, stemming, and language-specific processing. In addition, we conducted character replacement to handle spelling accents. For instance, characters like "à," "á," and "ã" were replaced with their root character, "a." To achieve this, we created a dictionary encompassing all possible combinations of spelling accents across all the languages present in our dataset. This function was applied to both the *comments* and *description* features in the train and validation datasets. Specifically, we only employed lemmatization while omitting stemming. Subsequently, we consolidated all comments associated with the same house number into a single row. Then, we concatenated the *cleaned_comments* and *cleaned_description* columns, creating a new column named *comments_and_desc*. To streamline the data further, we removed all other features, resulting in a single feature, *comments_and_desc*, for the X datasets. The house number was designed as the index for all four datasets.

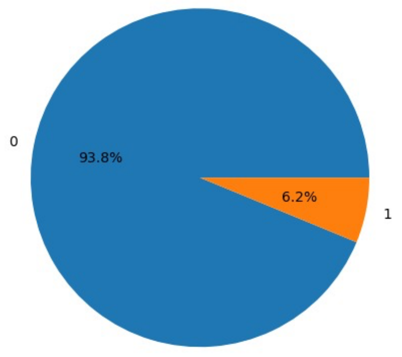
## 4. Feature Engineering

Feature engineering plays a critical role in transforming pre-processed text data into meaningful features that can be effectively utilized by machine learning algorithms. By engineering features, we extract relevant information, enhance the representation of the text, and improve the predictive power of our models.

In this project, our group explored four different feature engineering methods, beginning with sentiment analysis. To perform this type of analysis, we employed the *TextBlob* sentiment analysis method. Specifically, we utilized the *sentiment.polarity* attribute to assign a sentiment score to each text in the *comments_and_desc* feature. This score indicates whether the text conveys a positive sentiment, if the score is higher than zero, a negative sentiment, if the score is lower than zero, or neutral sentiment, if the score is equal to 0. However, upon conducting an in-depth analysis of the sentiment labels distribution, it became evident that this approach yielded suboptimal results. Notably, an overwhelming majority of the dataset, approximately 99.37%, was assigned positive sentiment analysis label. However, after careful examination by our team of the rows labelled as positive revealed the presence of comments with a negative sentiment. Regrettably, the sentiment analysis method did not accurately capture the negative and neutral sentiment in the comments, so we decided not to proceed with this approach.

The following method tried by our team was the Term Frequency – Inverse Document Frequency (TF-IDF) technique, which assigns weights to terms based on their frequency and rarity to capture their significance in document representation. Additionally, as two extra methods we have incorporated the Bag of Words (BoW) approach and the GloVe Embeddings. The first represents documents as a collection of words without considering their order, allowing the user to capture the presence or absence of specific terms in the text. Furthermore, we leveraged GloVe Embeddings, which utilizes pre-trained word vectors to represent words in a continuous vector space. This method captures semantic relationships between words and enables our models to benefit from contextual information. As said before, since the sentiment analysis does not provide accurate sentiment labels, we will not proceed with that feature engineering technique, but with the other three methods tested.

In light of the imbalanced distribution observed in our train dataset and the inability of our initial classification models using TF-IDF features to predict the minority class (class 1), we recognized the need to address the challenge posed by imbalanced datasets. Our team decided to employ the Synthetic Minority Over-sampling Technique (SMOTE). By applying the SMOTE technique to the three feature engineering methods we will utilize in our models, namely TF-IDF, BoW, and GloVe Embeddings, we aim to address the data imbalance challenge by generating synthetic samples for the minority class. Through the integration of SMOTE, we strive to create a more balanced and representative dataset, which in turn aids in achieving better classification results by improving the model's ability to learn from the minority class and reducing the bias towards the majority class.



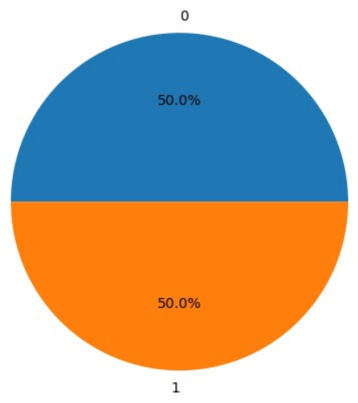


Figure 3 - Label Distribution Before SMOTE (on the left-handside)

Figure 4 - Label Distribution After SMOTE (on the right-handside)

In addition to implementing the SMOTE method, we have also explored the concept of assigning weights to the labels to address the imbalanced nature of the dataset. Within our train datasets, we have identified 7.862 instances belonging to class 0, which represents the majority class, and 521 instances belonging to class 1, which represents the minority class. To calculate the weight for the minority class, class 1, we decided to employ a widely used formula known as the class weight calculation formula in the fi eld of machine learning. This formula is derived based on the total number of samples in the dataset, the number of samples in the minority class and the total number of classes:

$$Weight_{minority\ class} = \frac{Total\ Samples}{\left(Nr.samples_{minority\ class}\right) \times \ (Nr.classes - 1)}$$

Therefore, we proceeded to the calculation of the weight for class 1, the minority class:

$$Weight_{class\ 1} = \frac{8.383}{(521) \times \ (2 - 1)} \simeq 16{,}09$$

Consequently, the class weights assigned to our imbalanced dataset are 1 for the majority label (0), and 16,09 for the minority label (1). These weights will be integrated into our machine learning models to prioritize the minority class (1) during training. By doing so, we aim to address the challenges posed by data imbalance and enhancing the model's ability to learn from the minority class. However, the decision to utilize either the SMOTE technique or the class weights will depend on the specific characteristics and requirements of each model, as we will assess which approach is more appropriate in each case. It is important to highlight that SMOTE was only applied on the training dataset and not validation, to evaluate the model's performance on unseen data as accurately as possible.

## 5. Classification Models

Our classification models were partitioned into five sections, each of which corresponded to a specific model. Within each section, our idea was to apply the three feature engineering methods that we defined before, that is, the TF-IDF, BoW, and GloVe Embeddings, however, due to high computational demand, some combinations of models were not able to run. The models used in our analysis encompassed Logistic Regression, Naïve Bayes and Random Forest as requested work, and KNN and Gradient Boosting as extra work.

Logistic Regression is a widely utilized linear classification model that finds applications across various domains. It estimates the probability of an instance belonging to a specific class by applying a logistic function to a linear combination of input features. The model is known for its simplicity and interpretability, making it well-suited for binary classification tasks. Logistic Regression assumes a linear relationship between the input features and the output variable. In our analysis, we initially applied logistic regression to the TF-IDF, BoW, and GloVe Embedding features without any imbalanced correction. Subsequently, we repeated the logistic regression using the three feature engineering methods after balancing the dataset using the SMOTE. This approach aimed to enhance the model's performance by addressing the data imbalance issue.

Naïve Bayes is a probabilistic classification model based on Bayes' theorem. It assumes that the features are conditionally independent given the class variable, simplifying the computation process. Despite its assumption of feature independence, Naïve Bayes exhibits simplicity, efficiency, and effectiveness in handling high-dimensional data. It is commonly employed in text classification tasks. In our analysis, we followed a similar approach for the Naïve Bayes model as the previous models. Initially, we applied the model to our train and validation datasets without any balancing treatment. Subsequently, we utilized the calculated weights, as mentioned earlier in this report, to balance the dataset and improve the performance of the Naïve Bayes model.

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It constructs an ensemble of decision trees and aggregates their predictions through voting or averaging. Random Forest models are robust against overfitting, capable of handling a large number of input features, and provide insights into feature importance. The model's versatility allows it to capture complex interactions within the data. In our analysis, we followed a similar approach as the approach followed in the Naïve Bayes. Initially, we applied the Random Forest model to the train and validation datasets without any balancing treatment. Subsequently, we incorporated the calculated weights to balance the dataset and enhance the Random Forest model's performance.

As additional work, our team undertook the implementation of other two distinct models: K-Nearest Neighbours (KNN) and Gradient Boosting. KNN is a non-parametric classification algorithm that assigns a class label to an instance based on the class labels of its k nearest neighbours in the feature space. Not relying on explicit assumptions about the underlying data distribution, KNN can handle nonlinear decision boundaries. While a simple yet effective model, its performance can be sensitive to the choice of the number of neighbours (k) and the distance metric employed. When attempting to execute KNN with TF-IDF features, the computational requirements exceeded feasible time constraints, resulting in an inability to assess its performance on this feature set. However, we did explore KNN with BoW features by conducting a grid search to identify the optimal number of neighbours without any imbalancing correction. Similarly, we investigated KNN with GloVe Embeddings by performing a grid search for the ideal number of neighbours, again without any unbalancing correction. Furthermore, we conducted an additional run with the GloVe Embeddings feature, this time applying the SMOTE to address imbalanced data.

Gradient Boosting, an ensemble learning technique, constructs a robust predictive model by combining multiple weak prediction models, typically decision trees. Employing a stage-wise approach and optimizing a loss function through gradient descent, Gradient Boosting demonstrates proficiency in handling complex data relationships and various data types, often achieving high predictive accuracy. Nonetheless, due to its computational intensity, executing Gradient Boosting with TF-IDF features proved infeasible within acceptable time frames. However, we successfully employed Gradient Boosting with BoW features both without and with SMOTE unbalancing correction. Additionally, we applied Gradient Boosting to GloVe Embeddings without unbalancing correction and subsequently with SMOTE correction.

It is worth noting that the selection of either SMOTE correction or weights correction depended on the models' performance and their respective compatibility with the correction methods. In total, we implemented 24 models, and comprehensive results for all these models can be found in Annex 1 of this report.

## 6. Evaluation and Results

As mentioned previously, the model approach to the imbalanced dataset depended on its compatibility. Using SMOTE contained a trade-off between our models not being able to learn the label with less instances and creating noise and overfitting. Therefore, we decided to apply it in Logistic Regression (LR), Gradient Boosting (GB) and K-NearestNeighbors. Given that the models Naïve Bayes and Random Forest contain a parameter to handle class weights, we chose this approach instead. We decided to focus our evaluation on the validation results, to provide an estimate of how well the model is expected to perform on new data. The most relevant metric to evaluate results and choose a final model was the F1-score, given that it combines precision and recall, therefore evaluating the model's ability to accurately identify false positives and false negatives.

The performance and metrics of the 24 variations of the models performed in the validation dataset can be seen in more detail in Appendix 1.

From the results, it is observable that, in most cases, balancing the dataset using SMOTE does not significantly improve the performance of the models. The accuracy, F1-score, precision, and recall values are similar in or even decrease significantly after applying SMOTE, such as the cases of BoW combined with LR and KNN using GloVe. Regarding feature engineering, the three approaches - TF-IDF, BoW and GloVe embeddings, have slightly similar performance when it does not include SMOTE, varying from values of 0.91 to 0.96 in F1-Score. It is also important to highlight that Random Forest was the model that provided more robustness and, consequently, less sensitiveness to the changes in feature engineering.

Among the implemented models, the Naïve Bayes model with TF-IDF features and weights correction stood out as the top performer and most generalizable. It achieved an impressive validation F1-Score of 0.96, along with high accuracy, precision, and recall values of 0.96. This model exhibited exceptional predictive capabilities, accurately classifying instances in the validation dataset. Therefore, this was our final combination chosen for the prediction and we aim to have a similar performance on the test data.

Throughout this project, we collectively gained invaluable experience, further enhancing our skills in the field of text mining. We encountered crucial stages in any text mining project, including pre-processing and feature engineering, which are vital for achieving accurate and meaningful results, so we tried our best to treat the data without removing significant portion or without distorting the dataset. Undoubtedly, we tackled various challenges during the project, particularly in dealing with computational expenses that hindered the execution of certain computationally intensive tasks, such as cleaning a vast amount of data, implementing intricate processes like translating and spelling correction, translating emojis into text, and expensive models, such as the combination of the Gradient Boosting with the TF-IDF feature engineering.

Despite the limitations imposed by computational constraints, it is crucial to acknowledge the overall usefulness of this project for us as future data scientists. The project showcased the practical application of text mining in diverse industries, emphasizing its significance and potential impact. In summary, this project not only solidified our understanding of essential text mining processes, but also made us recognize the value and utility of this subject in various domains.

## 7. References

Hofmann, M., & Chisholm, A. (2016). *Text Mining and Visualization: Case Studies Using Open-Source Tools.* Minneapolis: CRC Press.

## 8. Appendix

**Appendix 1 -** Models and Their Performance in the Train / Validation datasets

**Models and Their Performance in the Train / Validation datasets**

| **Model** | **Feature Engineering** | **Balanced** | **Accuracy** | **F1-Score** | **Precision** | **Recall** |
|---|---|---|---|---|---|---|
| **Logistic Regression** | TF-IDF | No | 0.94 | 0.91 | 0.94 | 0.94 |
| | | Yes - SMOTE | 0.95 | 0.94 | 0.94 | 0.95 |
| | BoW | No | 0.94 | 0.93 | 0.92 | 0.94 |
| | | Yes - SMOTE | 0.89 | 0.90 | 0.91 | 0.89 |
| | GloVe | No | 0.94 | 0.91 | 0.89 | 0.94 |
| | | Yes - SMOTE | 0.65 | 0.74 | 0.91 | 0.65 |
| **Naïve Bayes** | TF-IDF | No | 0.94 | 0.91 | 0.89 | 0.94 |
| | | Yes - Weights | 0.96 | 0.96 | 0.96 | 0.96 |
| | BoW | No | 0.94 | 0.91 | 0.92 | 0.94 |
| | | Yes - Weights | 0.94 | 0.92 | 0.91 | 0.94 |
| | GloVe* | No | 0.65 | 0.74 | 0.90 | 0.65 |
| **Random Forest** | TF-IDF | No | 0.95 | 0.93 | 0.94 | 0.95 |
| | | Yes - Weights | 0.94 | 0.92 | 0.94 | 0.94 |
| | BoW | No | 0.94 | 0.92 | 0.95 | 0.94 |
| | | Yes - Weights | 0.94 | 0.92 | 0.95 | 0.94 |
| | GloVe | No | 0.94 | 0.91 | 0.94 | 0.94 |
| | | Yes - Weights | 0.94 | 0.91 | 0.89 | 0.94 |
| **KNN** | BoW | No | 0.94 | 0.91 | 0.94 | 0.94 |
| | GloVe | No | 0.94 | 0.91 | 0.92 | 0.94 |
| | | Yes - SMOTE | 0.65 | 0.74 | 0.92 | 0.65 |
| **Gradient Boosting** | BoW | No | 0.94 | 0.92 | 0.93 | 0.94 |
| | | Yes - SMOTE | 0.86 | 0.88 | 0.91 | 0.86 |
| | GloVe | No | 0.94 | 0.91 | 0.90 | 0.94 |
| | | Yes - SMOTE | 0.79 | 0.84 | 0.90 | 0.79 |