NOVA IMS Information Management School

TEXT MINING

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Airbnb Properties Classification

Group 42

Adriana Costinha, number: 20230567

Ana Filipa Silva, number: 20230577

Beatriz Vasconcelos, number: 20230755

June, 2024

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Index

1. Introduction

In today's society, text holds significant importance due to the communication and meaning it transmits, making it crucial in the business industry, especially when it comes to user reviews. Airbnb, as a platform, thrives on the richness of textual data provided in host descriptions, property details, and guest reviews. These elements not only enhance user engagement but also serve as a vital source of information for understanding the quality and appeal of listings.

The problem presented to us was to predict whether an Airbnb property is unlisted or listed based on the following features: host description, property description, and property reviews. To address this, we developed NLP (Natural Language Processing) predictive models that can analyse and interpret the nuances of language used in these texts.

2. Data Exploration

2.1. Overview of the Data

The provided datasets included the following: two training files (*train.xlsx* and *train_reviews.xlsx*) and two testing files (*test.xlsx* and *test_reviews.xlsx*). The *train.xlsx* and *test.xlsx* files contain information about the listings, specifically the host's description and property descriptions, while the *train_reviews.xlsx* and *test_reviews.xlsx* files contain user reviews of the properties.

The initially provided training dataset was split into *X_train* and *X_test* to avoid bias during data exploration, in order to prevent examining unseen data for patterns that need to be addressed, as doing so could lead to data leakage.

In a more superficial analysis, we focused on duplicated values, null values, and the target class. A very small portion of the reviews dataset had duplicated data and null values, which were deemed not relevant and therefore dropped.

2.2. Statistic analysis

2.2.1. Description and Host About Columns

For both the properties *description* and *host_about* columns the word count had a minimum of a very small value of three, which raised some concerns (Figure 1, 2). This led to a further exploration of this situation, therefore, to finding special characters, dots, single letters and/or emojis (Figure 3, 4). Other than that, there were also explored standard patterns.
Furthermore, outliers were found in the *host_about* column, where entries were going above 300/400 words, which was very far from the mean (74 words).

Further analysis led to the discovery that this was due to some entries that contained multiple languages.

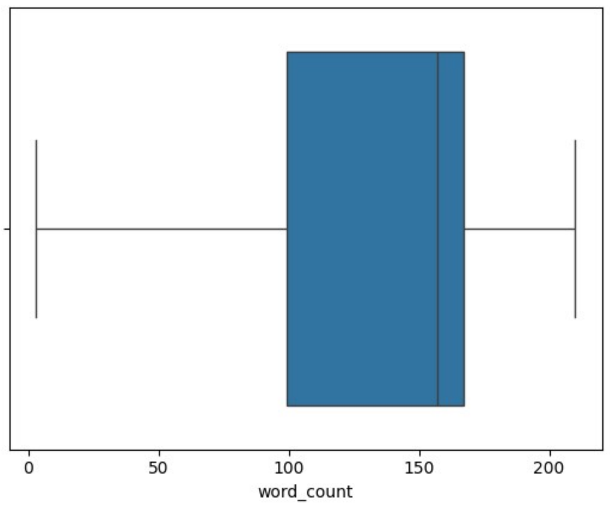

Figure 1 - Box Plot of the Word Count of the Description Column

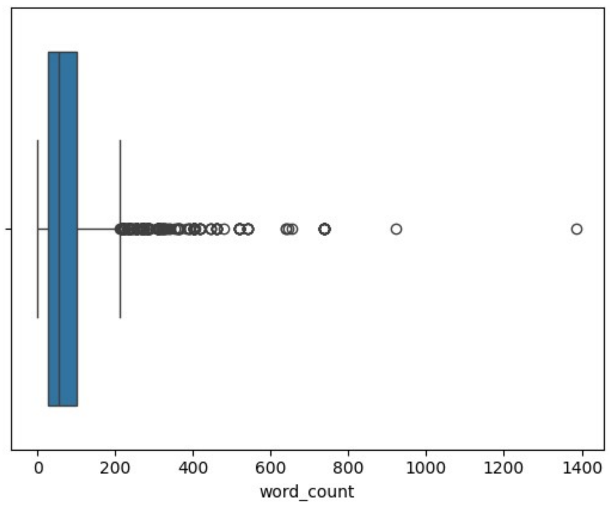

Figure 2 - Box Plot of the Word Count of the Host_About Column

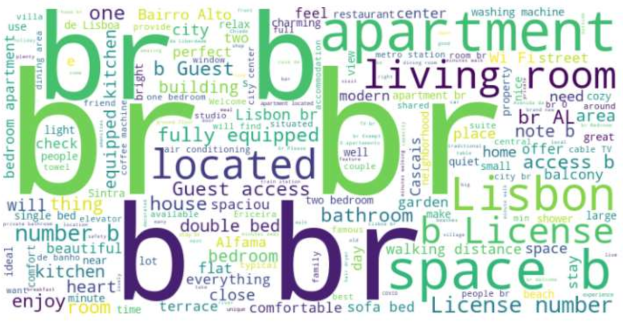

Figure 3 – Word Cloud of the Word Count of the Description Column

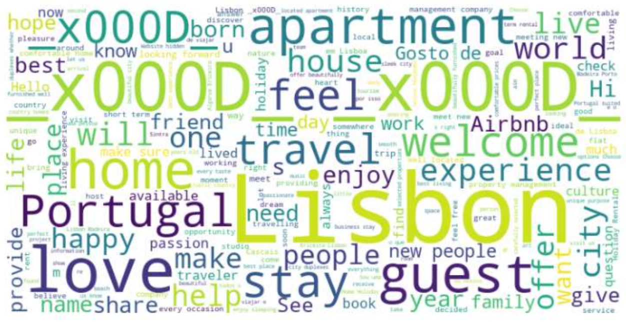

Figure 4 – Word Cloud of the Word Count of the Host_About Column

2.2.2. Comments Column

When analysing the distribution of reviews for each property, notable discrepancies were revealed, with certain houses having a significantly higher number of reviews compared to others (Figure 5). Some houses had up to 891 reviews, while others had none. Most properties had a low number of comments; a few properties had a very high number of comments; and properties with more than 200 comments were exceedingly rare.

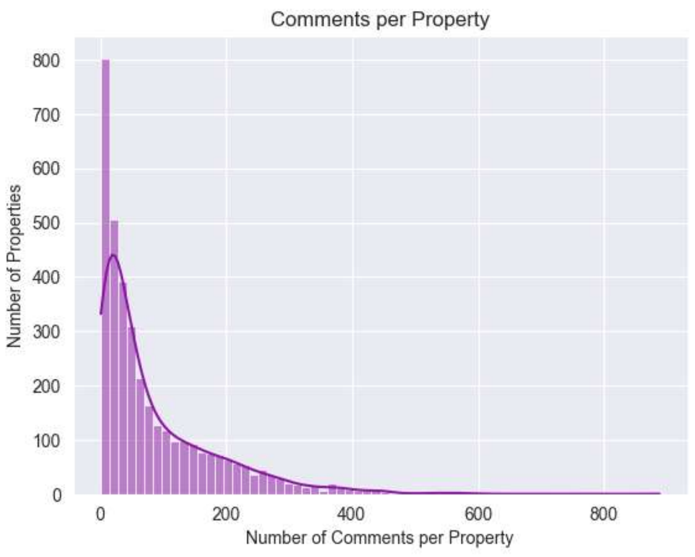

Figure 5 – Histogram of comments per property

This was suggesting of variations in popularity or visitation rates. Such variations hinted at disparities in popularity or satisfaction levels among different listings.

A word count analysis provided insights into the length distribution of reviews (Figure 6). The average word count per review was approximately 48 words, with a wide range from 1 to 1,019 words. Reviews with very few words (five or fewer) often lacked substantive information, similar to what was observed in the descriptions and host columns and therefore were considered noise in the dataset (Figure 7).
By analysing this statistics, we found significant heterogeneity in the number of comments across different properties, with notable outliers receiving a substantially higher number of comments.

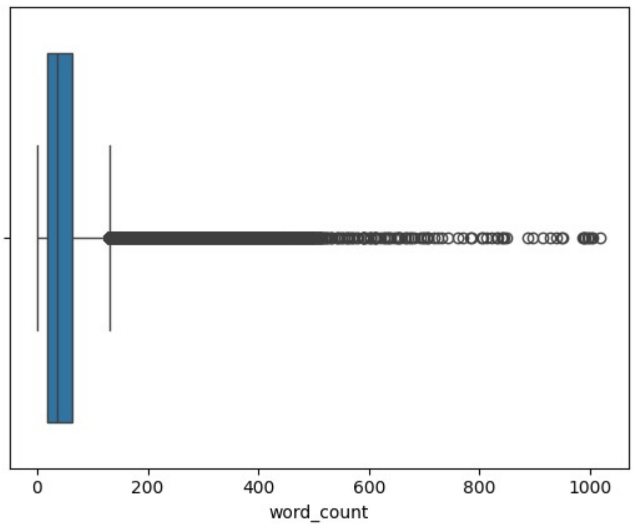

Figure 6 – Boxplot of the comments columns

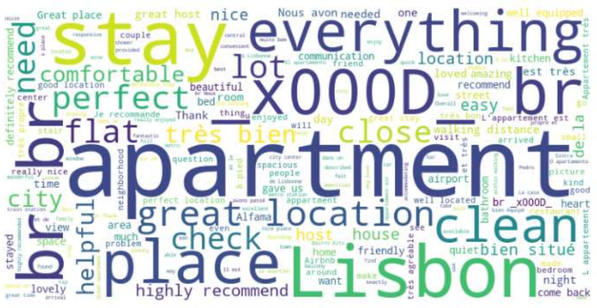

Figure 7 – Word Cloud of the comments columns

2.2.3 Target Variable Analysis

The label class is binary, indicating whether a listing is currently unlisted (1) or not (0). When analysing the frequency of both, it was evident that we were dealing with an unbalanced classification problem (Figure 8). This will need to be taken into careful consideration when evaluating the developed models. Therefore, we have decided to assess the model's performance based on their F1 macro and weighted scores as well as precision and recall, for the overall predictions, as well as these metrics for each of the classes.

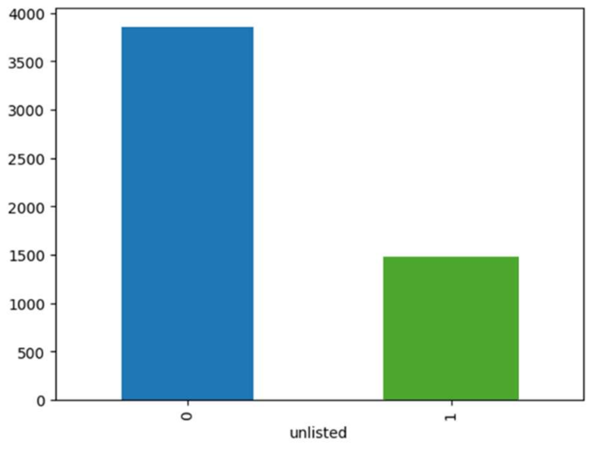

Figure 8 – Target Label (unlisted) distribution

Furthermore, we explored the difference in comments between unlisted and listed properties (Figure 9) to understand the impact that the reviews have on whether a house is listed or not. The visualization of the distribution revealed several key differences. Both unlisted and listed properties frequently receive a small number of comments, but this is more pronounced for unlisted properties, with a right-skewed distribution and a maximum around 600-700 comments. In contrast, listed properties have a broader and slightly less skewed distribution, with maximum comment counts extending up to around 800.

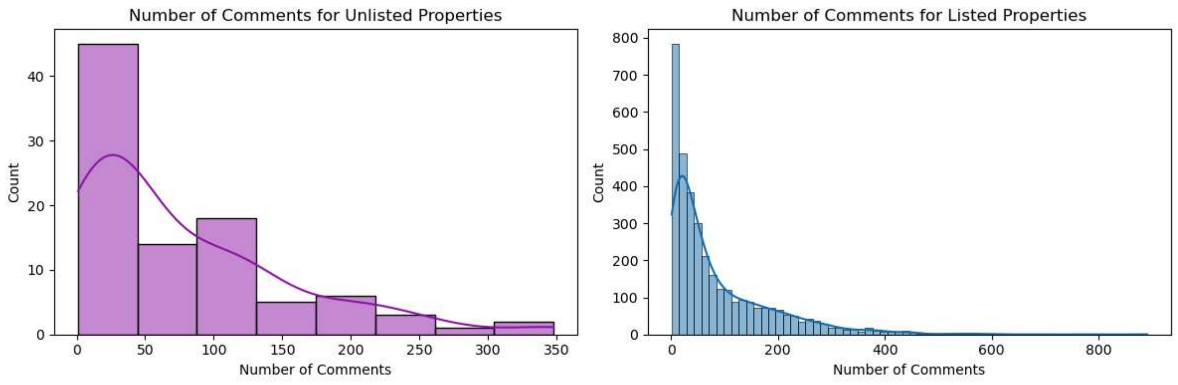

Figure 9 – Number of Comments for Unlisted and Listed Properties

2.2.4. Language distribution

After plotting the distribution for each language within each column present within our data (*description*, *host_about* and *comments*) it was obvious that it contained a large linguistic diversity.

<u>Description Column (Figure 10):</u>
English (en) dominates the description column, constituting approximately 81.8% of the entries.
Portuguese (pt) follows, comprising around 14.4% of the descriptions.
Other languages such as French (fr), Danish (da), and German (de) contribute smaller proportions, indicating a diverse linguistic landscape within the descriptions.

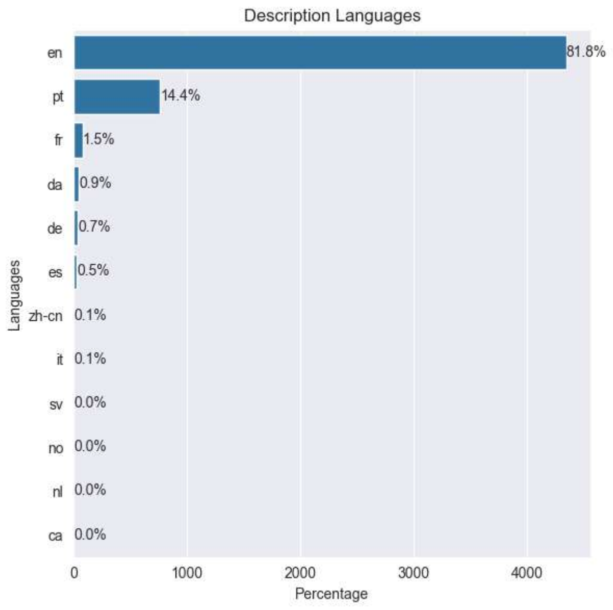

Figure 10 – Language distribution of the description column

<u>Host About Column (Figure 11):</u>
English (en) is prevalent in this column, accounting for approximately 70.08% of the entries.
Portuguese (pt) is the second most common language, representing around 20.6% of the host descriptions.
Similarly to the description column, other languages like French (fr), Spanish (es), and German (de) also have notable but smaller representations.

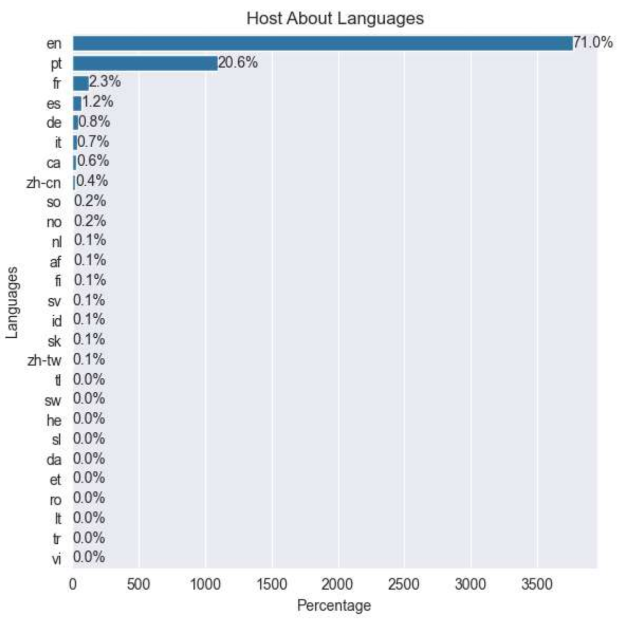

Figure 11 – Language distribution of the host_about column

<u>Comments Column (Figure 12):</u>
English (en) continues to lead in the comment's column, constituting approximately 64.3% of the comments.
French (fr) and Portuguese (pt) follow, contributing 14.8% and 6.2% of the comments respectively.
The presence of other languages such as Spanish (es), German (de), and Italian (it) among others, showcases the diversity of languages used in the comments section.

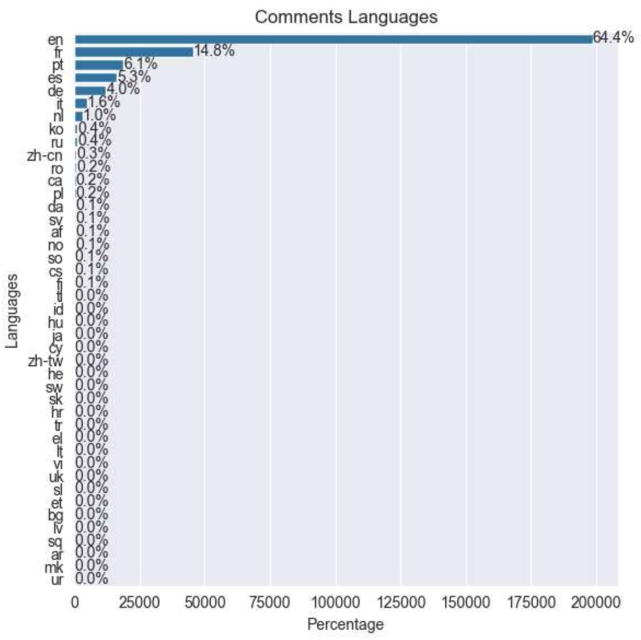

Figure 12 – Language distribution of the comments column

This led us to conclude that most of the language present was English, followed by languages such as Portuguese, Spanish, French, and German. Although these last four did not represent the majority, they could still hold valuable information within our data. Armed with this information, we had into consideration the high complexity of working with all the languages present in the data, specifically the necessity of loading different embedding models and the computational expense that comes with it. After considering the impact this could have on our future work, we decided to work only with the majority languages presented in the dataset, i.e., English, French, Spanish, Portuguese, and German. This approach allows us to focus not only on the dominant language but also on other minor languages that can impact future predictions.

Besides, it is important to note that for different columns of the same entry, languages were different. This would have to be later dealt with.

Finally, during exploration of the three columns we noticed diverse rows had nan values, as well as only a space, or multiple.

2.3. Pattern Analysis

It is important to mention that besides the standard elements to consider in a text mining problem, such as currencies and emojis, a review of the data revealed a high prevalence of specific patterns within the dataset. The phrases “alojamento local registro” and “license number” were found very frequently and were subsequently addressed.

3. Data Preprocessing

3.1. Text Standardization

For data preprocessing, we needed to standardize the text. To achieve this, we developed a pipeline consisting of numerous steps to condense the text as much as possible. Before removing or applying any patterns, we converted the data to lowercase to ensure consistency throughout. Then, we removed the patterns identified during data exploration as well as standard patterns and preprocessing steps that could introduce noise. This included replacing hyphens within words with spaces to maintain word separation, removing excel patterns, license numbers, emails, URLs, dates, numerical expressions (such as m2), special and Unicode characters, Roman numerals, replacing punctuation with spaces, and removing stopwords. Sometimes, due to the complexity of the text, this processing resulted in multiple consecutive spaces and null values. To address this, we collapsed the multiple spaces.

Since the English language represented most of the text, we treated one of its most characteristic aspects: contractions. This method could give us the possibility of retaining important information.

The curse of dimensionality is a problem that surfaces in the field of text mining, and therefore methods like stemming and lemmatization are usually applied. When to apply one or the other, depends on the problem at hand. Since we wanted to retain the majority of the sentiment the text could provide, we decided to use lemmatization.

3.1.1. Treating Outliers:

While doing data exploration, we came across various outliers on the *host_about* column. This happened because there were some entries written with multiple languages. It was one of the reasons our language detector had so much difficulty detecting. To resolve this issue, we separated each language section, divided them and selected one with the condition of it being one of the languages we chose. If this condition wasn't verified, it would update the text to 'not available'.

3.1.2. Dealing with missing values:

As stated before, we found multiple rows with nan values, as well as multiple spaces in diverse columns. After applying preprocessing to the three columns, multiple rows lost their data. This happened because they might have been made of only stopwords, special characters and/or patterns. To resolve these issues, we replaced it with the sentence 'not available'.

For data exploration we analysed all the languages present in our dataset. Since we decided to work with some algorithms that were not multilingual, we decided to work with only the top five most used languages. This raised an issue - how would we deal with the data we now had of so many languages we decided to 'discard'. We simply updated each row to the sentence 'not available', similarly to what we did to resolve the nan values. This meant we had to detect the language for all columns individually.

3.2 Extra: Preprocessing

3.2.1. Emojis

We used the library *demoji* which finds emojis within a string and replaces them with their description. By doing so, we were able to transform emojis into valuable information that could be straight away analysed, since their sentiment is very short and clear.

3.2.2. Summarize

Addressing the outliers found in the *comments* columns, and taking into consideration that the following methods were greatly computationally expensive we used a pre-trained machine learning model, a multilingual seq2seq model that can be used for generating summaries from text in various languages, therefore relaxing the problem of having longer text. As a result, entries that had more than 300 words of text were turned into a 20 words summary.

3.2.3. Sentiment Analysis

In our efforts to extract more valuable insights from our data, we decided to incorporate sentiment analysis to enrich the information fed into our predictive models. By leveraging the *distilled_student_sentiment_classifier* from the *transformers* library, we automatically assigned a sentiment score to each listing. This score captures the emotional tone carried in the text associated with each listing, providing an additional layer of data. Incorporating these sentiment scores could significantly enhance the performance of our models by giving us a deeper understanding of the context and nuances within the textual data.

3.3. Post Processing Data Exploration

For each of the columns, the figures representing the text before and after preprocessing show that the patterns identified were successfully removed, as evidenced by the word clouds plotted.
Additionally, discrepancies in word count across columns were also effectively addressed. In the host_about column the outliers were prosperously removed through the separation of the various languages in one instance, as explained in the section above.
In the reviews we can see a significant difference in the outliers due to the summarize function mentioned above.

3.3.1. Description Column

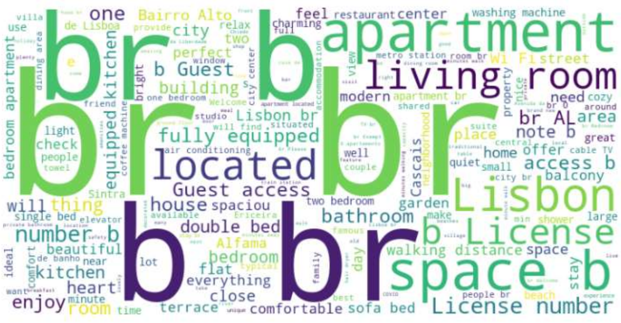

Figure 13 – Word Cloud of the Description Column before the preprocessing

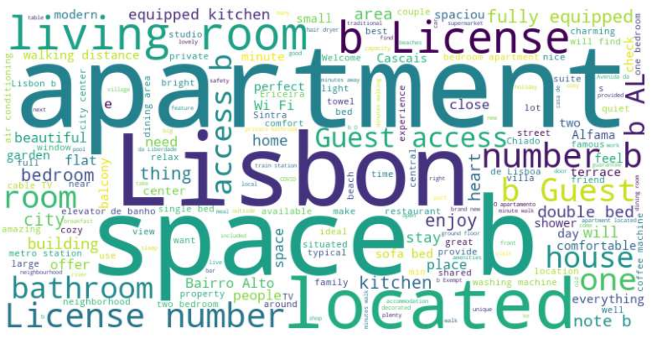

Figure 14 – Word Cloud of the Description Column after the preprocessing

3.3.2. Host About Column

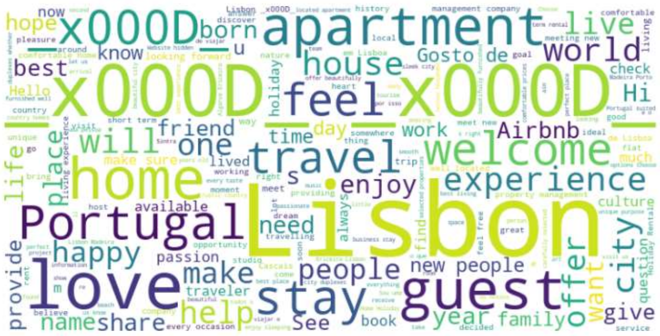

Figure 15 – Word Cloud of the host_about Column before the preprocessing

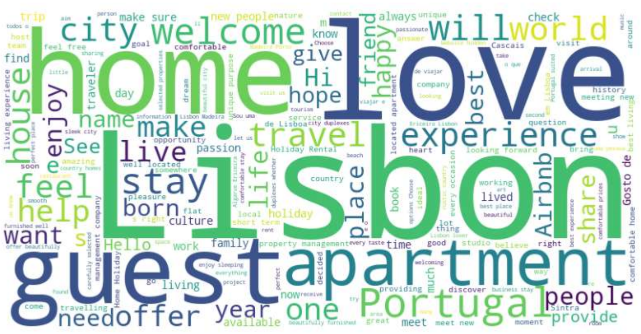

Figure 16 – Word Cloud of the host_about Column after the preprocessing

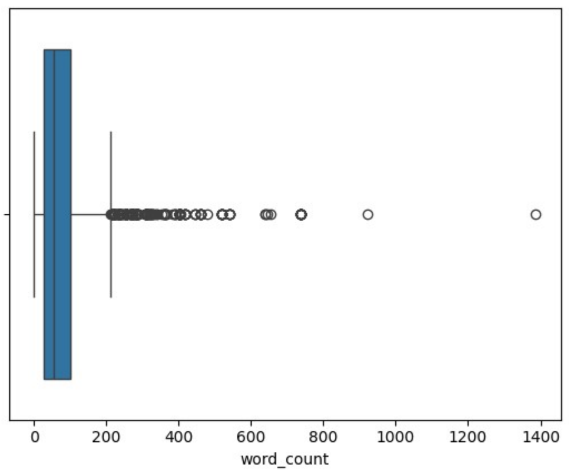

Figure 17 – Box plot of the word count of the host_about Column before the preprocessing

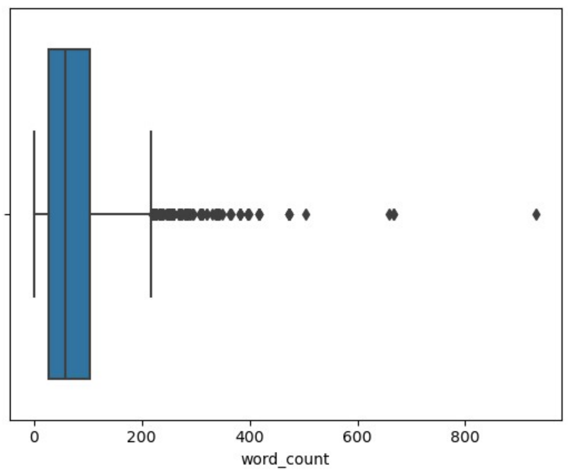

Figure 18 – Box plot of the word count of the host_about Column after the preprocessing

3.3.3. Comments Column

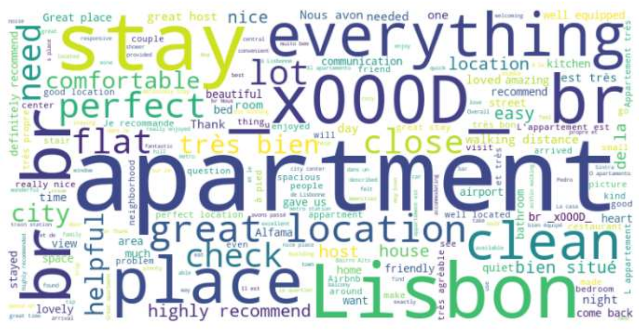

Figure 19 – Word Cloud of the comments columns before preprocessing

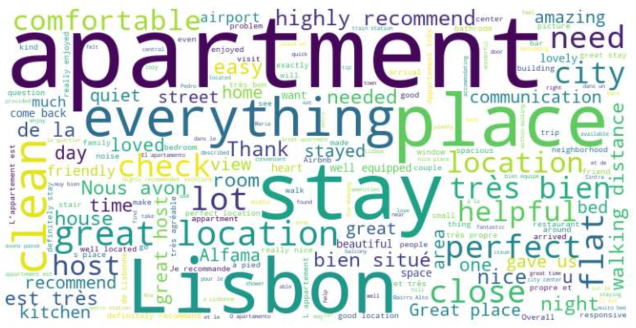

Figure 19 – Word Cloud of the comments columns after preprocessing

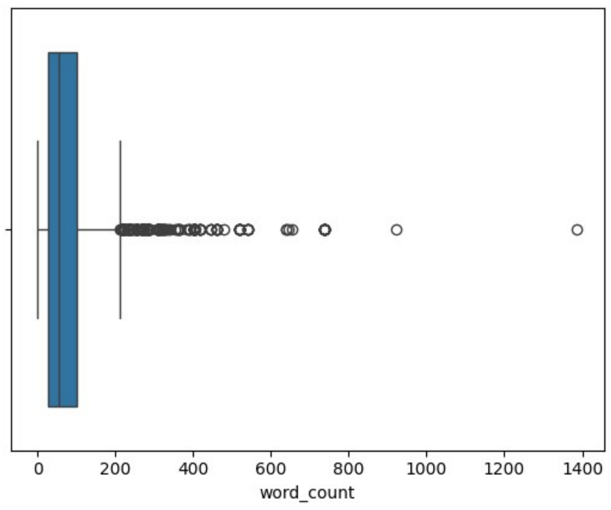

Figure 20 – Box plot of the word count of the comments Column before the preprocessing

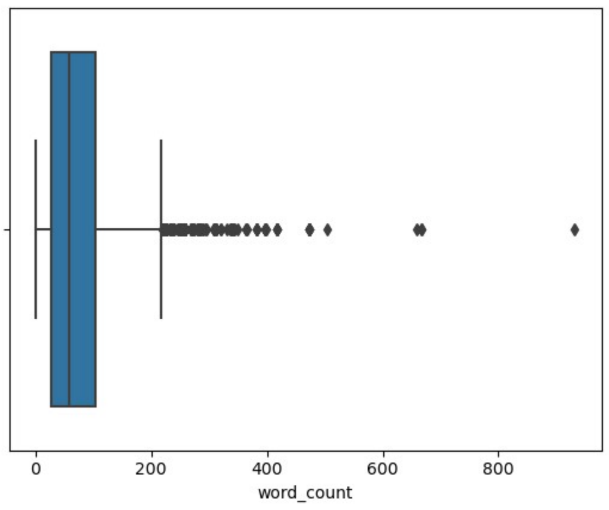

Figure 21 – Box plot of the word count of the comments Column before the preprocessing

4. Feature Engineering
When reaching Feature Engineering we took on two approaches: using word embeddings and Transformer-based Embeddings.

Word Embeddings are simple to implement, fast and memory efficient, although they do not capture multiple meanings of a word in different contexts and are limited to capturing semantic information.
On the other hand, Transformer-based Embeddings provide context-sensitive word representations, allowing for a more in depth understanding of language. But it does require significant computational resources, as well as time to train.

Since both have their pros and cons, we have decided to use various embeddings with different models, to try and achieve the best results.

4.1. Word Embeddings

To extract word embeddings from our text data, we began by tokenizing the text, which involves splitting strings based solely on whitespace. This tokenization leverages previous preprocessing steps. Each token is then converted into its corresponding embedding using the model in question, provided the token is recognized within the model's vocabulary. For tokens not found in the model, no embedding is retrieved, and these are omitted from further processing. After this, instead of applying padding at this stage, we have decided to handle the embeddings differently based on the context: for individual fields like host descriptions and property descriptions, we compute the mean of the embeddings to create a single representative vector for each text entry. For comments, we aggregate and then average embeddings to handle variable lengths and ensure consistent data handling without the need for padding.

Since these models do not preserve sequential order, we are only going to use them for certain machine learning models like Logistic Regression (LR), Naive Bayes (NB), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Random Forest(RF) and XGBoost (XGB). Padding introduces sparse areas (zeros) in the data that do not carry meaningful information, which can lead to poorer performance in these models. More importantly, the architectures of these machine learning models are not designed to handle sequential data, assuming that the input features are independent and contribute equally. By averaging the embeddings, we align with this assumption by providing a summary vector that represents the entire text equally, rather than emphasizing the order or placement of words, which would be meaningless to these models.

4.1.1. GloVe

GloVe embeddings were applied to the data because they ensure that the differences between words capture meaningful semantic relationships. Unlike other word embedding techniques that primarily focus on local text, GloVe combines both global statistics and context window methods. It offers flexibility as these models are available for different vocabulary sizes and vector dimensions. Technically, the concept of GloVe can be applied to any language, provided there is a sufficiently large corpus to train on.

However, for languages with limited available text data or where the text corpus is not diverse enough, GloVe may struggle to develop an accurate set of embeddings. This situation becomes even more prominent as the volume and diversity of languages other than English are quite small, especially after preprocessing. Furthermore, it was not possible to download the files for other languages due to them not being available.

With all of this in mind, we have decided to continue working with alternative embedding methods that can generalize better across different forms of a word, such as those that follow.

The pretrained model from GloVe chosen is not multingual[1], containing only embeddings for words found in English. The model has been trained on a corpus that include 6 billion tokens, where each word is represented by 300-dimensional vectors.

4.1.2. FastText

By being an extension of the Word2Vec model, this model is particularly powerful when it comes to understanding languages with rich morphology or handling words which were not seen during training.
The selected FastText model, trained on a multilingual corpus of 157 languages[2], trained on *Common Crawl* and *Wikipedia*. These models were trained using Continuous Bag of Words (CBOW) in dimension 300.

4.1.3. Wiki2Vec

The chosen model[3] represents a set of pre-trained word embeddings generated using the Word2Vec algorithm, trained on a corpus from *Wikipedia*, trained on a multilingual corpus of 12 languages, therefore processing a multitude of topics and information, capturing human text behaviour.
Each word is represented as a 100-dimenstional vector, a trade-off between computational efficiency and the depth of semantic detail.
Choosing a corpus from Wikipedia ensures diversity in expressions, from formal to colloquial, which is a very relevant point in our problem.

4.2. Extra: Transformer-based Embeddings

When it comes to transformer-based models, extensive text cleaning is not always necessary and can sometimes be counterproductive, since these models are trained on large, diverse datasets that typically include a wide range of linguistic features as well as noise. Therefore, we decided to use the raw data, as recommended by the official source of the chosen transformers.

Although the number of comments was large, they were also lengthy. To address this, we used the summarization function mentioned earlier, applying it before cutting or filtering comments to preserve the initial overall sentiment. However, this approach was still too computationally expensive. Consequently, we reduced the dataset size so that the total

[1] GloVe: Global Vectors for Word Representation (stanford.edu)
[2] Word vectors for 157 languages · fastText
[3] https://wikipedia2vec.github.io/wikipedia2vec/intro/

number of tokens per comment did not exceed a certain threshold, while still trying to include as many tokens as possible without dropping below a second, lower threshold.

With this approach, we aim to preserve valuable information in another type of representation, maintaining the total sentiment from before the initial comment reduction.

Lastly is important to noticed that in order to capture the greatest amount of value from the data, and because the transformers we used were pretrained on a large corpus of multilingual raw data, we decided to feed our raw data directly to the model[4].

4.2.1 mBert

Multilingual BERT, a version of the Bidirectional Encoder Representations from Transformers (BERT), processes text differently than sequential models that handle text word by word. This model considers all words in a sentence simultaneously, allowing it to capture context from both the left and right sides of a word. This capability addresses a common challenge in the field of text mining. The embeddings produced are context-dependent, capturing different meanings of the same word depending on its context.

4.2.2. XLM-RoBERTa

Although this transformer is an adaptation of roBERTa[5], its architecture has more leverage once it has been trained for longer on larger datasets, therefore understanding the context of words in a sentence more effectively, capturing detailed meanings and cross-lingual similarities.
Similarly to mBert, xlm-roBERTa is also multilingual and sequential.

5. Classification models

In order to ensure a diverse range of models, we have chosen to work with the following: Logistic Regression (LR), Naive Bayes (NB), K-Nearest Neighbors (KNN), Multi-Layer Perceptron, Random Forest (RF), and XGBoost (XGB).

For each of these models, we decided to feed them with the different embeddings mentioned earlier. In an effort to maximize the use of our data for training and validation, and to prevent overfitting by ensuring the model performs well across multiple subsets of the data, we performed cross-validation for each model's training. Additionally, we performed a grid search to hyperparameter-tune RF and XGB, aiming to enhance their performance, prevent overfitting and underfitting, and adapt the model to specific dataset characteristics, thereby improving generalization. The inputs fed to each model were as follows: property reviews, host descriptions, property descriptions, and an extra feature we calculated—the sentiment based on the reviews.

For each of the models and the fed embeddings, we obtained the following results:

[4] https://huggingface.co/google-bert/bert-base-multilingual-cased
[5] FacebookAI/xlm-roberta-base · Hugging Face

5.1. Logistic Regression (LR)

Embedding	F1-weighted	F1-macro	Precision	Recall	Accuracy
FastText	0.88	0.85	0.83	0.87	0.87
Wiki2Vec	0.88	0.85	0.83	0.87	0.87
MBert	0.87	0.84	0.83	0.85	0.87
XLMroBERTa	0.89	0.86	0.85	0.87	0.88

5.2. Naive Bayes (NB)

Embedding	F1-weighted	F1-macro	Precision	Recall	Accuracy
FastText	0.87	0.85	0.83	0.87	0.87
Wiki2Vec	0.87	0.85	0.83	0.87	0.87
MBert	0.87	0.85	0.83	0.87	0.87
XLMroBERTa	0.87	0.84	0.83	0.87	0.87

5.3. K-Nearest Neighbors (KNN)

Embedding	F1-weighted	F1-macro	Precision	Recall	Accuracy
FastText	0.87	0.83	0.84	0.83	0.87
Wiki2Vec	0.88	0.85	0.85	0.86	0.88
MBert	0.88	0.85	0.84	0.86	0.88
XLMroBERTa	0.87	0.84	0.83	0.84	0.87

5.4. Multi-Layer Perceptron (MLP)

Embedding	F1-weighted	F1-macro	Precision	Recall	Accuracy
FastText	0.88	0.85	0.83	0.87	0.87
Wiki2Vec	0.88	0.85	0.83	0.88	0.87
MBert	0.88	0.86	0.84	0.88	0.88
XLMroBERTa	0.88	0.85	0.83	0.87	0.87

5.5. Extra: Random Forest

Embedding	F1-weighted	F1-macro	Precision	Recall	Accuracy
FastText	**0.90**	**0.87**	**0.86**	**0.89**	**0.90**
Wiki2Vec	0.89	0.86	0.85	0.87	0.89
MBert	0.89	0.87	0.85	0.88	0.89
XLMroBERTa	0.90	0.87	0.86	0.88	0.90

5.6. Extra: XGBoost

Embedding	F1-weighted	F1-macro	Precision	Recall	Accuracy
FastText	0.88	0.85	0.85	0.86	0.88
Wiki2Vec	0.89	0.86	0.86	0.86	0.89
MBert	0.90	0.87	0.86	0.88	0.90
XLMroBERTa	0.90	0.87	0.87	0.88	0.90

Taking into consideration the metrics above, and keeping the problem of Airbnb's listings in mind, we have decided to prioritize recall over precision. High recall for listed properties is important to avoid missing any active listings, which could have directly negative business implications.

Therefore the best model with the optimal embedding appears to be FastText using Random Forest, which has an F1-score of 0.87, a precision of 0.86, and a recall of 0.89. We have decided to use this embedding and predictive model to address our problem.

6. Conclusion

Text mining has had an enormous impact on business, especially for those that rely heavily on online reviews. These reviews come in a great variety, including diverse aspects and forms, such as slang, which can be challenging to handle.

Furthermore, one of the major challenges in this field is label imbalance, which complicates the problem even further.

In addressing our specific issue, the multilingual aspect was one of the main challenges, along with the computational expense it entails. Consequently, we had to make some trade-offs, such as focusing on the most representative languages.

Even though these challenges arose, we managed to achieve good results by performing FastText embedding and feeding those embeddings into a Random Forest, achieving an F1 macro score of 0.87.

Further improvements could address the multilingual aspect, finding ways to manage the large quantity of data across all languages. This might involve producing synthetic data or adjusting class weights to cope with label imbalance. However, synthetic data must be carefully explored, as this approach can sometimes lead to worse outcomes.