NOVA IMS Information Management School

Text Mining Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Predicting Airbnb's property listings

Group 16

Gonçalo Cardoso, 20230588

Diogo Silva, r20201643

Marisa Sequeira, 20230989

June, 2024

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

Table of Contents

1. Introduction

The sharing economy's explosive expansion has completely changed how individuals locate places to stay. Travelers and hosts can interact through websites like Airbnb, which offers a variety of accommodations ranging from opulent villas to quaint apartments. Efficient management of these listings is crucial for the platform and hosts alike.

In this project our objective is to predict, using several classifiers and data manipulation methods, if a property will be unlisted in the next quarter, or remain listed.

The data available is a set of 4 excel files ('.xlsx'), that constitute our train and test data, each of these composed of two files in which the first has general descriptive information about each property, and the second all the reviews associated with the respective properties.

2. Problem Definition

Our initiative attempts to solve a significant problem that hosts of Airbnb encounter: speculating as to whether a home will be unlisted (that is, removed from the marketplace) in the upcoming quarter. We can make educated guesses by utilizing Natural Language Processing (NLP) tools to examine property descriptions, host profiles, and visitor remarks.

Our job is to create an NLP classification model given a dataset that includes information on the property, the host, and the reviews left by previous visitors. To be more precise, we want to forecast if a property will stay listed (0) or go off the market (1) for the next quarter. This forecast can assist hosts in improving visitor experiences, fine-tuning pricing policies, and listing optimization.

Python and well-known libraries like NLTK, Scikit Learn, Keras, or PyTorch were used. Our initiative encourages innovative ideas by allowing for methodological flexibility.

We will examine data exploration, preprocessing, feature engineering, model selection, and evaluation in the sections that follow.

3. Data Exploration

In this section we examine our dataset's preliminary analysis of our dependent and independent features, concentrating on word counts, language distribution, and common words. Our aim is to understand how the train dataset is structured, in order to manipulate what we deem necessary.

This analysis was conducted on two datasets, train and train_reviews.

We started Data Exploration by looking at the dependent feature 'Unlisted', in order to understand how balanced it was, concluding it's overwhelmingly unbalanced, as option 'Listed' represents about 70% of the values, depicted in Figure 2.

As for the independent features (‘description’, ‘host_about’, and ‘comments’), we started by analyzing language distribution among the text, and found that we could narrow most of the data into 6 languages, that represent the vast majority, these were: Portuguese, English, French, Spanish, German, and Danish. The distributions for each of the features can be seen in Figures 3, 4 and 5.

Later, we started understanding how the text was structured, by creating word counts for each row, and eventually calculating the mean for each feature to get mean number of words per row (seen in table 1). This statistic would give us important knowledge on which feature had most information per row, and possibly contribute most towards our end goal.

After this, we applied the method of word distribution seen in class to see which words were appearing most, and concluded that for the most part, the most frequent words were just mostly determinants or html tags we would have to get rid of later.

Finally, after checking for missing values, it’s important to note that we assumed that the duplicate rows where the host’s description and the property’s description were the same, but had several different comments, were just properties built the same and sold by the same owner, for example hotel rooms, and did not remove them.

4. Data Preprocessing

In this section, our priorities were assuring consistency, managing linguistic diversity, and improving data quality. For this we started by trying to translate all the text in languages not included in the 6 most common languages mentioned previously. This experiment ended up not working very well during implementation, and since these languages were representative of a small amount of data, we set the goal of coming back to this point later on if we had time, which ended up not being the case.

Furthermore, we applied basic techniques also shown in class, mainly lowercasing the data, stopwords (for words in the common languages list), and finally RegEx to remove all the extra symbols and html tags that would ultimately end up annoying further implemented algorithms.

To conclude pre-processing, we merged the main data set, which halberded the descriptive part of the properties (‘description’ and ‘host_about’), with the respective reviews. Important to note that we ended up stacking all the reviews for each Airbnb, meaning the current structure of each data frame (train and test) is composed of the three features, where the reviews for one specific property (represented by one row), are all in the same row separated by a space. This made sense to us since our end goal was to have words that would accurately describe each property, meaning having all the comments stacked in one big sentence was just a way of having all the positive or negative information in one place, more on this later.

Before moving on to Feature Engineering, we saved our pre-processed data in one folder, for organization and performance purposes.

5. Feature Engineering

Our objective here was to convert unstructured textual data into characteristics that are useful and collect pertinent information to apply our models on.

Initially, we applied a regular split on the train data, in order to generate our validation set.

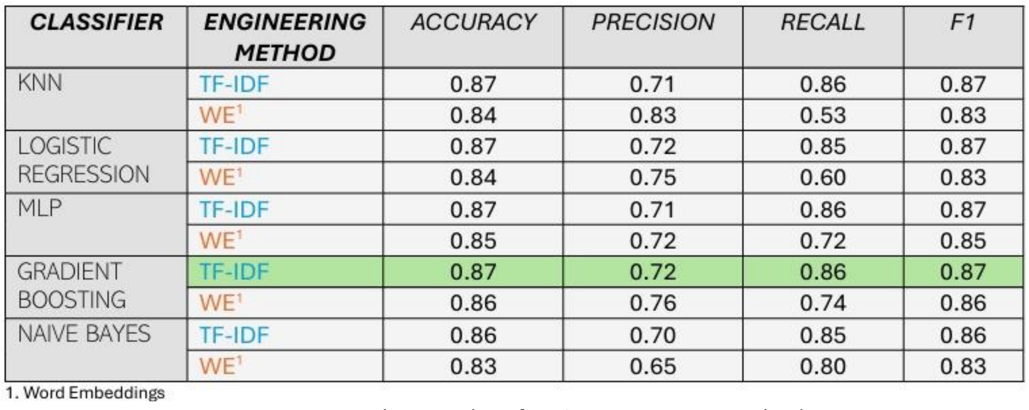
The plan was to implement sentiment analysis on the reviews, since it made sense to understand the emotional aspect of the comments made on each property, but after careful consideration, and trying different ways of implementation, we opted to leave this algorithm behind, and moved on to vectorization and embeddings. The techniques used in this phase were TF-IDF and Word Embeddings, inspired on the implementation displayed in the classes. Bag of Words was thought about, but since it's a 'worse' version of TF-IDF, which doesn't highlight words that are more important to the context, we also opted to not apply it. Important to say that in contrary to TF-IDF, Word Embeddings was extremely computationally expensive, and took around 30 hours to conclude.

In the notebook, a section for Pad Sequencing also appears, which we started to implement when we realised we might want to use LSTM as a classifier, but since the later model's implementation was coming off as unpredictable and uncertain, we opted to go for MLP, not needing pad sequencing any longer.

6. Modeling & Evaluation

The models we investigated, their performance measures, and our conclusions drawn from the evaluation's findings are all included in this part. Our goal was to try both word embeddings and TF-IDF to forecast property unlisting. The main models we chose to use as classifiers, also implemented in class, were K-Nearest Neighbors, Logistic Regression, and Multi-Layer Perceptron (MLP). As extra models, we applied Naïve Bayes, and Gradient Boost, which ended up being our best model.

Since our target was observed to be unbalanced, we thought f1 score was the best metric to keep in mind when analyzing the performance of each model. After each implementation of the models, a confusion matrix can be seen (also shown in the annex), describing the models performance. All the scores can be seen in the following figure.



CLASSIFIER	***ENGINEERING METHOD***	*ACCURACY*	*PRECISION*	*RECALL*	*F1*
KNN	TF-IDF	0.87	0.71	0.86	0.87
	WE[1]	0.84	0.83	0.53	0.83
LOGISTIC REGRESSION	TF-IDF	0.87	0.72	0.85	0.87
	WE[1]	0.84	0.75	0.60	0.83
MLP	TF-IDF	0.87	0.71	0.86	0.87
	WE[1]	0.85	0.72	0.72	0.85
GRADIENT BOOSTING	TF-IDF	0.87	0.72	0.86	0.87
	WE[1]	0.86	0.76	0.74	0.86
NAIVE BAYES	TF-IDF	0.86	0.70	0.85	0.86
	WE[1]	0.83	0.65	0.80	0.83

1. Word Embeddings

Figure 1 - Results per Classifier & Engineering Method

7. Conclusion

TF-IDF seemed to consistently produce higher F1-scores and recall for all models. Although helpful, word embeddings fell short of TF-IDF's effectiveness. This might be explained by TF-IDF's capacity to draw attention to terms that are significant in context. In terms of models, Gradient Boosting seems the most trustworthy option for forecasting property unlistings due to its performance and consistency on both engineering techniques.

As a group, we think that with more time other techniques could had possibly been implemented, and maybe deepened the dimension of the project scope, but we believe the objective was fulfilled, and ultimately we agree to have enhanced our understanding of text mining and it's underlying ropes, which in the end, produced good results on the validation set (and hopefully the test set).

8. Annex

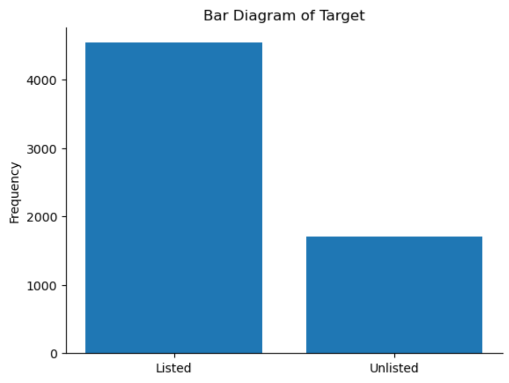
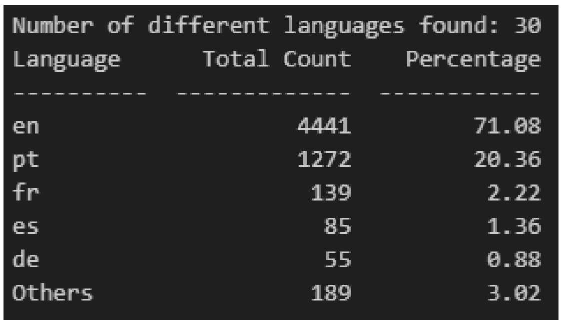
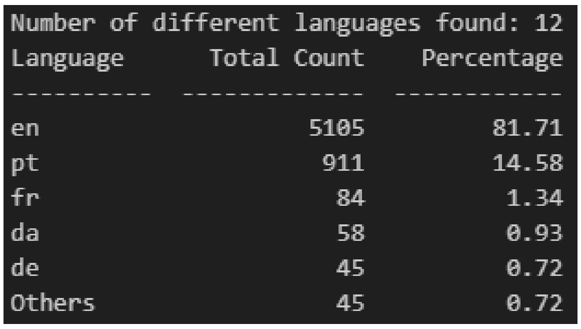


Figure 2 - Target's value distribution



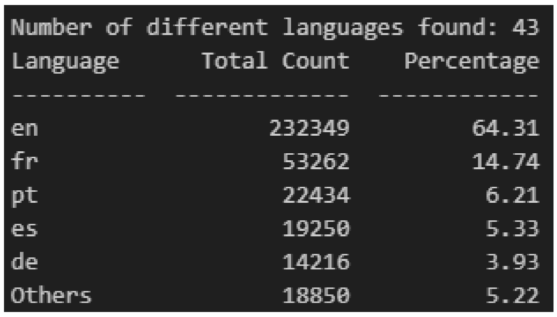
```
Number of different languages found: 30
Language      Total Count    Percentage
----------  -------------  ------------
en                   4441         71.08
pt                   1272         20.36
fr                    139          2.22
es                     85          1.36
de                     55          0.88
Others                189          3.02
```

Figure 3 - Language distribution in 'host_about'



```
Number of different languages found: 12
Language      Total Count    Percentage
----------  -------------  ------------
en                   5105         81.71
pt                    911         14.58
fr                     84          1.34
da                     58          0.93
de                     45          0.72
Others                 45          0.72
```

Figure 4 - Language distribution in 'description'



```
Number of different languages found: 43
Language      Total Count    Percentage
----------  -------------  ------------
en                 232349         64.31
fr                  53262         14.74
pt                  22434          6.21
es                  19250          5.33
de                  14216          3.93
Others              18850          5.22
```

Figure 5 - Language distribution in 'comments'

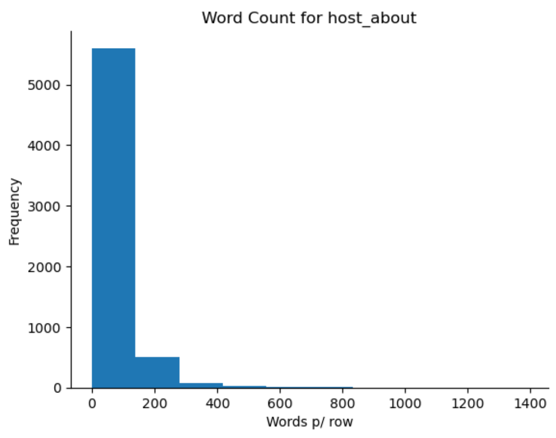


Figure 6 - 'host_about' word count p/ row

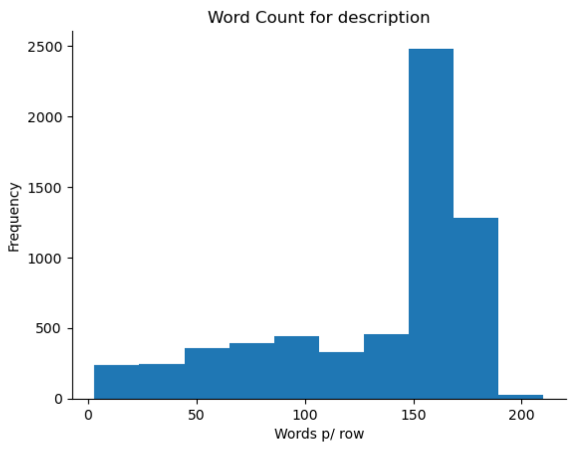


Figure 7 - 'description' word count p/ row

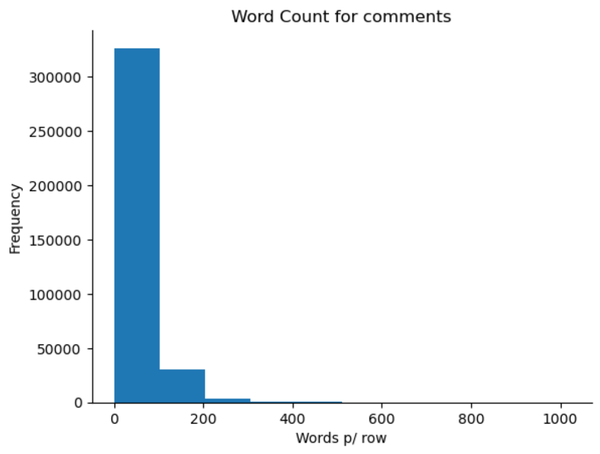


Figure 6 - 'comments' word count p/ row

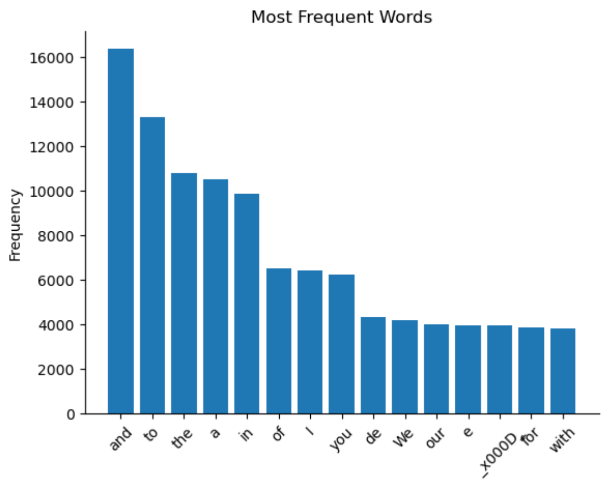


Figure 7 - 'desciption' word frequency

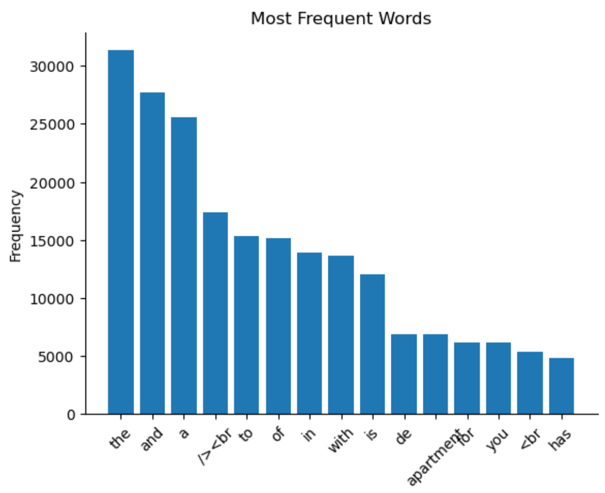


Figure 10 - 'host_about' word frequency

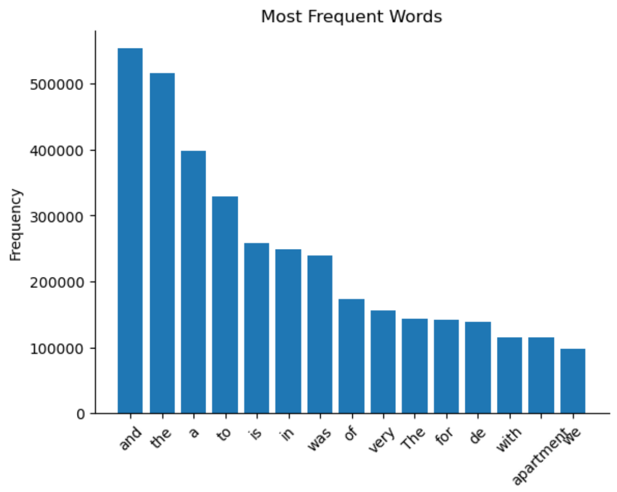


Figure 11 - 'comments' word frequency

Table 1 - Mean Words p/ row

Column name	Mean Words per row
Description	132.86
Host about	73.56
Comments	47.97