Text Mining Project

Natural Language Processing for Airbnb listings

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Group 18

Diogo Barros, 20230555

Diogo Roiçado, 20230557

Pedro Guedes, 20230569

Rodrigo Rocha, 20230593

June, 2024

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

INDEX

1. Abstract

The rise of platforms like Airbnb has transformed how people book accommodations, offering diverse options to suit various preferences and budgets. However, both hosts and guests face uncertainty regarding the future availability of listings. To address this, NLP techniques were implemented to predict whether an Airbnb property will be removed from the platform in the next quarter.

The usual data exploration step revealed a class imbalance concerning the target label, with only 27.33% of the total properties marked as unlisted. Duplicated rows were kept due to their quantity and interpretability in the context. As for missing values, only two records contained them and were promptly eliminated due to their irrelevance. Lastly, sentiment analysis revealed mostly positive reviews, suggesting that sentiment alone might not be sufficient to predict listing status.

The preprocessing steps involved cleaning and standardizing the text data, removing irrelevant elements, and lemmatizing words to their root forms. We then used techniques like Bag of Words (BoW), TF-IDF, GloVe, LaBSE, and XLM-R to transform the text data into meaningful features. These features were combined with numerical data for model training.

Various classification models were used both with undersampling and oversampling techniques to address class imbalance. Gradient Boost, AdaBoost, and CatBoost models were the ones that stood out, with F1 validation scores around 0.88 for undersampling and 0.89 for oversampling. Ultimately, the model chosen to perform Grid Search on was Gradient Boost Machines, in order to optimize hyperparameters. The Grid Search was done using both under and oversampling, achieving slightly better results with oversampling. Finally, the optimal model was applied to the BoW preprocessed test set.

2. Introduction

The rise of websites such as Airbnb has completely changed how consumers research and reserve accommodations for their trips. Airbnb offers a wide selection of options to suit different preferences and budgets, with millions of properties listed globally. The uncertainty surrounding the future availability of advertised properties is an issue faced by both hosts and guests. We will use Natural Language Processing (NLP) techniques to forecast whether an Airbnb property will be removed from the platform during the upcoming quarter to tackle this challenge.

To create a prediction model, we want to use natural language processing (NLP) to examine actual Airbnb property descriptions, host profiles, and feedback from prior visitors. In the next quarter, our model will categorize each home as either still listed (0) or unlisted (1), giving hosts and prospective guests useful information.

To successfully complete the project, several key steps must be undertaken. These include preparing text data for analysis, developing and evaluating classification models, and using advanced NLP techniques to extract features. Our goal is to develop a strong prediction model that may help hosts optimize their property listings and guests make well-informed selections when booking their accommodations by combining NLP approaches with machine learning algorithms.

3. Data Exploration

To begin our exploration, we examined the types and amounts of data available in our datasets. The train dataset contained 6.248 rows, while the train reviews dataset consisted of 361.281 rows. We verified the data types for both datasets to ensure they are appropriate for our analysis. Specifically, the *description* and *host_about* columns in the train dataset, as well as the comments column in the train reviews dataset, were correctly labelled as object types, suitable for handling string data in pandas. This indicates that these columns primarily consist of text data. Additionally, we ensured that the unlisted column was correctly labelled as an integer type, appropriate for representing categorical data as binary values. This initial data validation ensured that our datasets are correctly structured for the upcoming text mining and NLP tasks.

3.1. Amount of Listed and Unlisted

Firstly, we observed that the distribution of the target variable was unbalanced, with 4550 listings marked as “listed” and 1708 as “unlisted”, where properties unlisted represented only 27,33% of all records. The visual representation of class distribution underscores this imbalance, with the varying heights of the bars reflecting the difference in instances between the classes (Figure 1 – Listing Status). This imbalance poses challenges during model training, potentially biasing the model towards the majority class, so it was something we considered for the future.

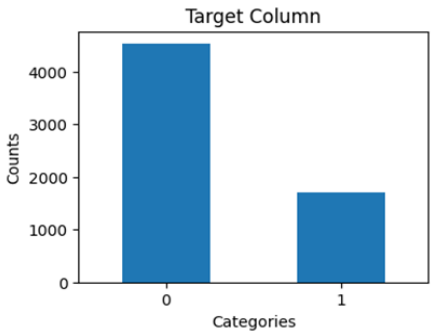

Figure 1 - Listing Status

3.2. Missing Values and Duplicated rows

We found two missing values in the *comment's* column, which were eliminated due to their irrelevance in the data set, as well as lack of interpretability since a review without commentary doesn't constitute a proper review. However, duplicated rows were encountered which raised the question of whether they should be removed, especially for specific cases where the same host had multiple different records with the same description.

After pondering, we decided to keep all the duplicated rows and treat them all as different houses, since the host in question could own, for example, an entire building with similar houses.

3.3. Characteristics of the Datasets

To gain more insights about our dataset we created a table with the word count, character count, average word length, stop words count and stop words ratio for each column.

Beginning with the analysis of the *description* column (Figure 2 - Summary of Description Column), it became evident that each sentence, on average, was composed of approximately 132 words (Figure 3 – Word Count of Description Column). Further examination also revealed an average word length of 5 characters per word. Notably, the presence of stop words within these sentences (Figure 4 – Most Frequent Words of Description Column) was significant, with an average count of 47 stop words. This accounted for approximately 34% of the total words used in each sentence.

Additionally, a word cloud (Figure 5 – Word Cloud of Description Column) visualization was generated to provide visual representation of the most frequently occurring words within the descriptions. This word cloud offered a comprehensive overview of the prevalent themes and topics addressed in the dataset, supplementing the quantitative analysis with qualitative insights.

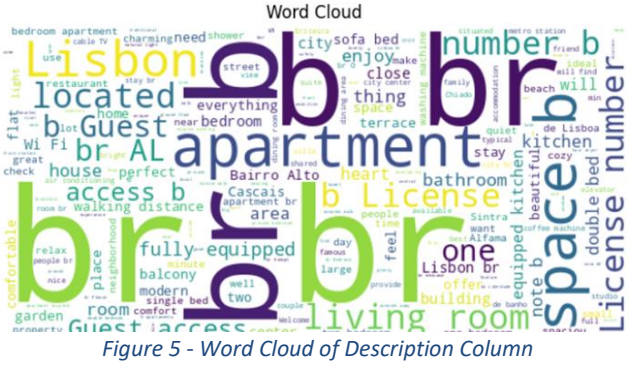

Figure 5 - Word Cloud of Description Column

Analysing the *host_about* column (Figure 6 - Summary of Host_about Column), we observed that the sentences provided have an average word count of 75 words (Figure 7– Word Count of Host_about Column) with a mean of 444 characters. The average word length was approximately 5 characters per word. Notably, there was also a noticeable presence of stop words (Figure 8 – Most Frequent Words of Host_about Column), with an average count of 31 stop words per sentence, accounting for around 39% of the total words used.

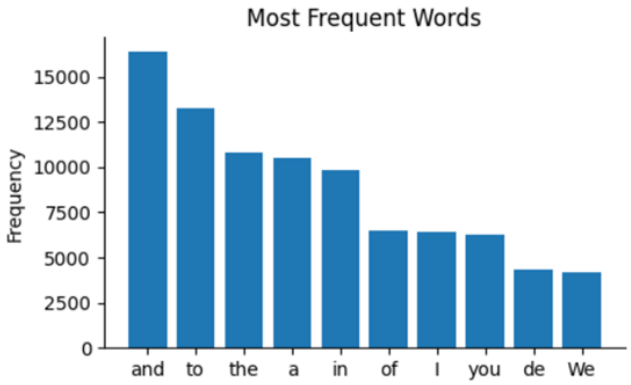

Figure 8 - Most Frequent Words of Host_about Column

In addition to the statistical summary, a word cloud (Figure 9 – Word Cloud of Host_about Column) visualization was generated to illustrate the most common words appearing in the *host_about* column.

To streamline our analysis process, we decided to focus on the number of reviews per property and the predominant languages present in the train reviews, particularly given the substantial volume of records in the training reviews. This approach aims to optimize our time and resources efficiently.

3.4. Languages

Further exploration revealed insightful statistics about the textual data in the train and reviews dataset. Firstly, we decided to examine the predominant languages in the *description, host_about* and *comments* column.

For the three columns, the most common languages were English, followed by Portuguese and then French. Despite this, the *host_about* column contained more diversity of languages than the *description* column. Our goal was to identify the languages of each column so that we could apply specific preprocessing operations to each Airbnb.

Upon examining the data, we immediately noticed some misclassified sentences. As it was too time-consuming to change all of them by hand, we decided to check the least common languages. Most of these were incorrect. For example, "*Swallow Hostel sinta-se em casa!*" was classified as Italian when it is indeed Portuguese, so we changed these to the appropriate language. In total, we changed 11 sentences in the description column.

For the *host_about* column, we found many more misclassified sentences, with most being classified by us as unknown since they were empty or contained only one character that could not be classified as any language.

Moreover, we noticed that most Danish descriptions were wrongly assigned, as they almost all had the format "* License number xxxxx/AL*". This was something we kept in mind for the next steps. In the host_about column, we applied the same logic of identification.

Additionally, it came to our attention that in the train dataset, the combinations of *description* and *host_about* languages were not always consistent within each row. For example, 702 rows had the pair (en, pt) and 286 (pt, en). This inconsistency in language pairs was something we considered for the next steps as it could affect the text processing and analysis.

3.5. Final Insights

We have uncovered an interesting pattern: nearly 72% of unlisted properties didn't have reviews associated with, providing valuable insights into why they remain unlisted. On the other hand, properties with reviews had a higher probability to be listed, almost 94%. This imbalance suggests that reviews were a significant predictor and should be carefully considered in modelling. Addressing this imbalance will be crucial. Techniques such as undersampling or oversampling, alongside a focus on appropriate performance metrics, can help mitigate bias and improve model accuracy.

In summary, our initial data exploration has provided valuable insights into the distribution of listings status, data types, missing values and textual characteristics of the datasets. These findings will inform our preprocessing steps and model development to ensure accurate predictions of Airbnb listing unavailability.

4. Data Preprocessing

To begin the data preprocessing steps across all the training data, we merged the languages detected before, for *description* and *host_about* by the *detect_language* library into our original dataframe.

4.1. Initial Changes in the texts

Our preprocessing workflow was designed to refine and standardize the textual data, ensuring its suitability for robust text analysis.

Initially, we took the task of converting all textual content into lowercase. This action promoted uniformity across the dataset, simplifying comparisons and subsequent analyses. Following that, we removed HTML tags from the text corpus. These tags, often irrelevant for textual analysis, possess the potential to introduce noise into the data. Additionally, we identified and systematically replaced certain regular expressions, such as the tag 'License number' with 'license', we've also removed some strings that aren't informative, like '_x000d_' replacing it by an empty string, and some more informative placeholders, thereby enhancing the overall clarity and coherence of the

dataset. Likely the previous step, we found some symbols like: "▲", "☼", "★" and remove them because the algorithm only recognizes characters and numbers.

To enhance the readability and consistency of the text, we deployed an algorithm to expand contractions like "*you'll*" to "*you will*". These measures not only augmented the textual uniformity but also contributed to noise reduction, facilitating subsequent analytical efforts. This act was followed by the removal of all the punctuation in the text data. Lastly, in order to fortify the integrity of the dataset, we created a mechanism that transformed all numerical values into a string *number*.

4.2. Extra Preprocessing - WordNinja

We applied an additional method, Wordninja, to improve the readability of the datasets. Specifically, we used it to split concatenated words into individual words, but only for entries where the text was in English, assigning a certain probability to a string being written separately.

4.3. StopWords

Further refinement of the dataset was achieved through the strategic removal of stopwords. As we have mentioned before, there were distinct language sentences, some of them which were not covered by the LNTK library so we couldn't apply stopword removal to the following languages: traditional chinese, simplified chinese, latin, afrikaans, estonian, vietnamese and Croatian. However, since the prevalence of these languages was rare in the dataset, we were able to apply the method to roughly 99% of it.

4.4. Extra Preprocessing – Spacy Lemmatization

Both stemming and lemmatization were considered and, even though stemming is computationally less expensive and faster, lemmatization is generally preferred for tasks that require higher precision and a better understanding of context. This method uses a dictionary to map words to their base forms, ensuring that the resulting lemma is a valid word. We came across one problem that was handley solved, the method was transforming "*u*" in "*us*" so we replaced all the occurrences with of "*u*" in "*you*". Both lemmatization and SpaCy Lemmatization were applied, yielding very similar results. SpaCy Lemmatization is a method that leverages SpaCy's robust natural language processing capabilities to reduce words to their base or dictionary form. SpaCy uses part-of-speech tagging and a comprehensive language model to accurately identify the context and morphological structure of words. In the end, we decided to move further with the application of SpaCy.

4.5. Exploration

Following the preprocessing steps, we proceeded with the same word counting and graphing procedures for the train dataset. To express the results, the following information is about the column *description.* As expected, the actual values displayed in (Figure 10 - Summary of Description

Column), proved notably more favourable to analysis compared to the previous ones in (Figure 2 - Summary of Description Column).

This improvement highlights the effectiveness of our preprocessing efforts. Upon comparing the word clouds generated before and after preprocessing, a clear enhancement in data quality and informativeness was evident. The word cloud generated post-preprocessing (Figure 11 - WordCloud of description column) contains a richer representation of the textual data compared to (Figure 5 – Word Cloud of Description Column).

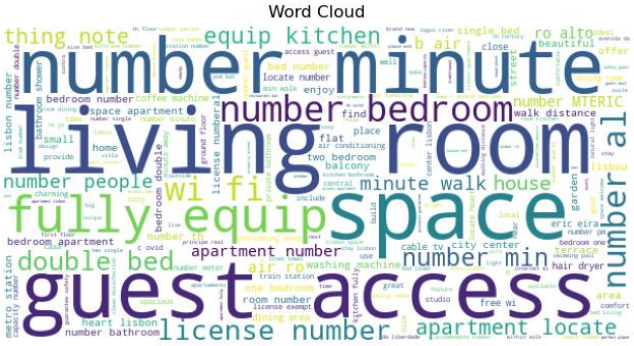

Figure 11 - WordCloud of Description Column

4.6. Extra Exploration - NER

In this section of our analysis, we employed a Named Entity Recognition (NER) to extract entities from the textual data in our Airbnb dataset. NER is a natural language processing task that identifies and classifies named entities (such as persons, organizations and more) within a body of text. Using SpaCy's pre-trained model to process the text and extract entities, we divided the text into manageable chunks to optimize processing efficiency. The entities extracted from each chunk were aggregated, resulting in a comprehensive list of entities for each column. As we can see in (Figure 12 - Most Common Entities in Description) the most common entities are people, cardinals and organization related.

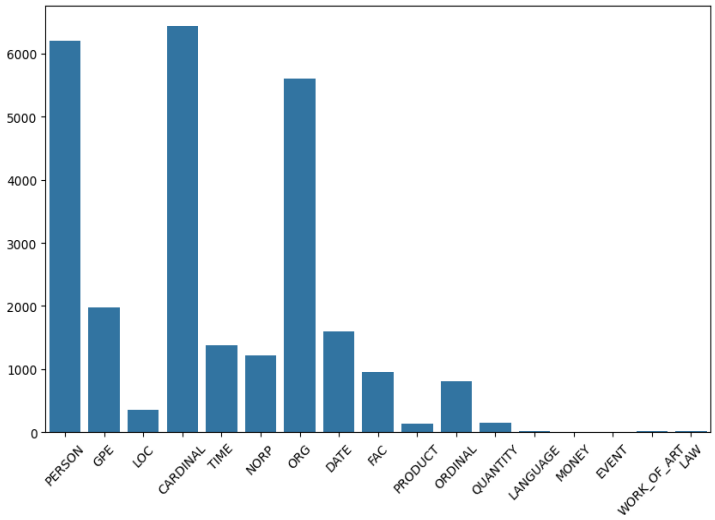

Figure 12 - Most Common Entities in Description

We then categorized the extracted entities based on the listing status to discern any potential differences in entity distribution between listed and unlisted properties, enabling us to analyze whether certain types of entities were more prevalent in one category than the other. Next, we calculated the percentage of occurrence for each entity type in both listed and unlisted properties (Figure 13 - Percentage of occurrences in host about per entity type) and (Figure 14 – Percentage of occurrences in descriptions per entity type)

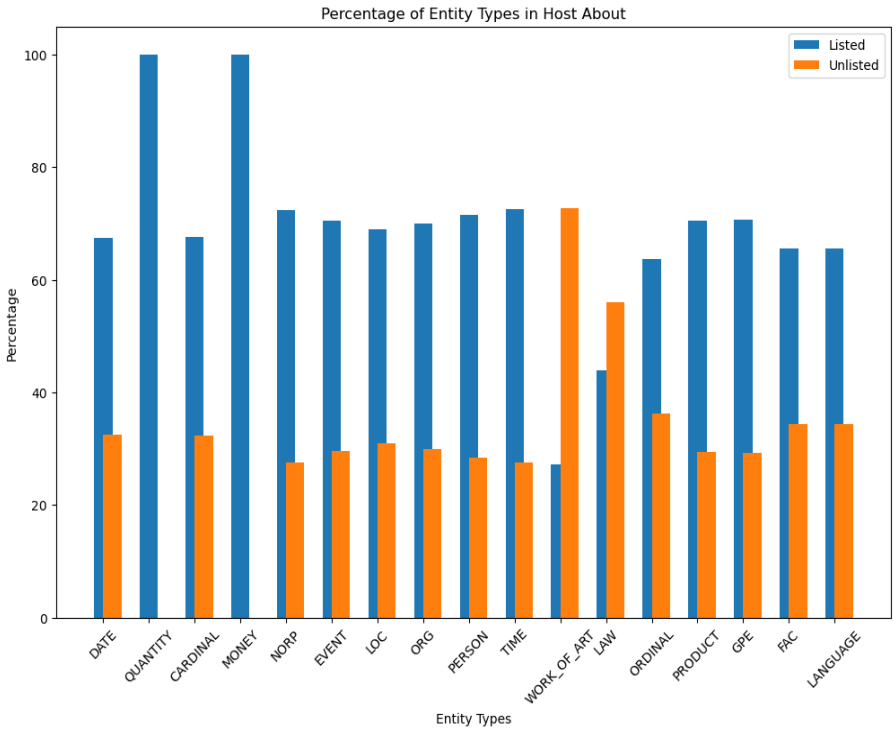

Figure 13 - Percentage of Occurrences in and Host_about per Entity Type

This analysis allowed us to identify any significant disparities in the distribution of entity types across different listing statuses. We have the example of 'LAW' and 'WORK_OF_ART' that are expressions from the *host_about* column that are much more frequent on the unlisted airbnb's. While 'MONEY' and 'QUANTITY' are predominantly in Listed properties. These occurrences stand in stark contrast to the other entity types, which demonstrate a more balanced distribution across both listed and unlisted properties, ranging from 20 to approximately 35%. This distribution closely resembles the imbalance observed in the target variable. The distinct prominence of 'MONEY' and 'QUANTITY' entities in unlisted properties suggests that they may play a significant role in determining listing status, inviting further investigation into the underlying factors driving this association.

4.7. Sentiment Score Implementation and Analysis

We used Hugging Face's Transformers library's nlptown/bert-base-multilingual-uncased-sentiment model for sentiment analysis to gain insights from the comments. This model was chosen since it supports several languages, something crucial given our dataset's diversity.

Since the sentiment score computation inherently involves some level of text preprocessing (tokenization and truncation), we opted not to perform additional cleaning steps on the train reviews dataset before applying the sentiment analysis. We considered the built-in preprocessing sufficient for our needs, focusing on obtaining sentiment scores for each review.

Several conclusions can be drawn by contrasting these sentiment scores with the listing status in our training dataset, (Figure 15 - Sentiment Score distribution).

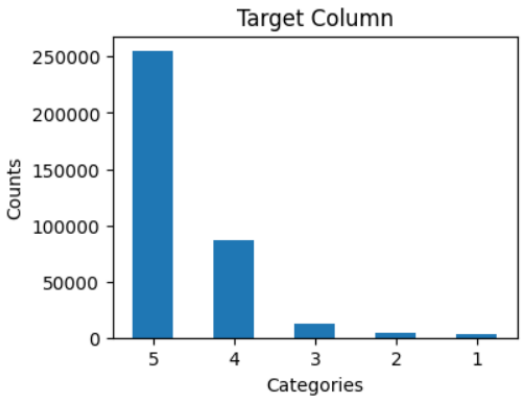

Figure 15 - Sentiment Score Distribution

First off, most evaluations fall into the higher sentiment categories (4 and 5), indicating that the overall sentiment is primarily positive. This suggests that most guests are happy with their stays, which benefits the listing's status, and that visitor experiences across all Airbnb listings are typically favourable. It is important to note that the distribution of the goal variable (listed/unlisted) is far less skewed than the sentiment analysis results. Although 27% of the ads are not listed, only roughly 5% of reviews have neutral or negative ratings. This disparity raises the possibility that sentiment analysis by itself might not be adequate to predict the target variable and so, we decided against using it for the models.

4.8. Split Data

After completing the preprocessing steps, we proceeded to split our training data to create a validation subset, essential for evaluating the performance of our models. Subsequently, we implemented both oversampling and undersampling techniques to address class imbalance in our dataset, ensuring that the model is trained on a more representative sample of data, techniques that will be applied several times. Following this, we applied the StandardScaler to both the oversampled and undersampled training features, ensuring that each feature contributes equally to the model's training process.

5. Feature Engineering

We started by using the Bag of Words (BoW) model to transform our training data, applying both oversampling and undersampling techniques with the StandardScaler.

After transforming the text data using the Bag of Words approach, we proceeded to analyse bi-grams and tri-grams. Bi-grams and tri-grams are useful for understanding common phrases and sequences in the text, providing deeper insights into the content and context of the descriptions. For the *description* column, we identified the most frequent bi-grams, which are illustrated in (Figure 16 - Bi-gram for description) and (Figure 17 - Bi-gram for host_about).

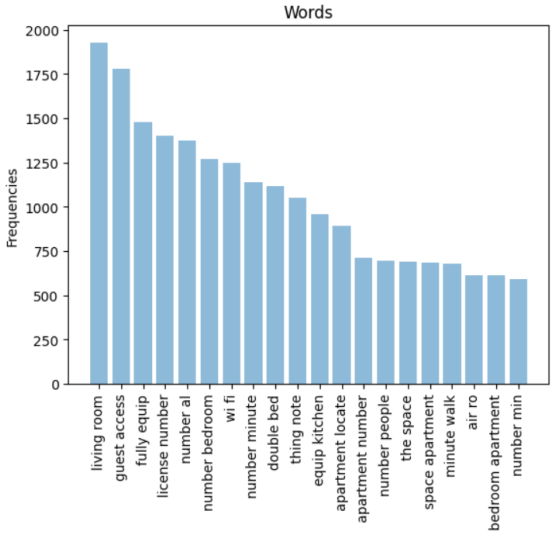

Figure 16 - Bi-Gram for Description

The most common bi-grams in the *host_about* column were '*love travel*' and *'number year*', both of which had been appropriately transformed in our preprocessing steps. Similarly, we extended our analysis to tri-grams, which are illustrated in (Figure 18 - Tri-gram for description) and (Figure 19 - Tri-gram for host_about)

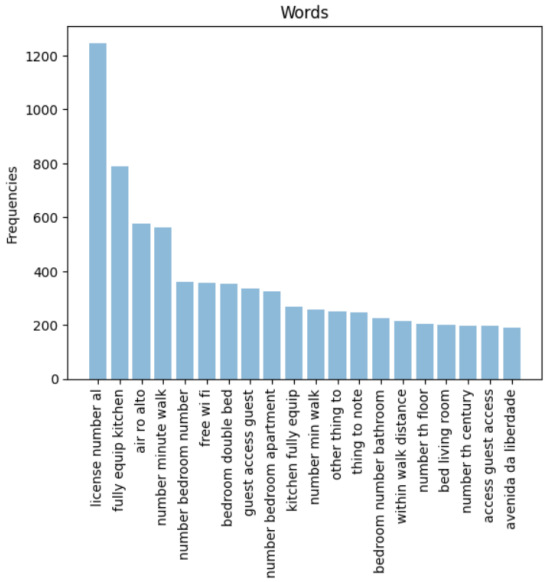

Figure 18 - Tri-gram for Description

In the *description* column, the most frequent tri-grams included phrases such as "*fully equipped kitchen*" and "*number minute walk*", which often describe the property's features. In the *host_about* column, common trigrams like *"meet new people*" and "*feel like home*" emerged, reflecting the hosts' welcoming and personable attitudes.

Next, we implemented the Term Frequency-Inverse Document Frequency (TF-IDF) model with both undersampling and undersampling, separately. TF-IDF helped us understand the importance of words in the context of the entire dataset, providing a more refined feature set for our models.

We used the pre-trained embedding from 'word2vec-google-news-300' to implement the GloVe model to convert each word into 300-dimensional vectors, applying undersampling and oversampling separately. By averaging these word vectors, we created fixed-length feature vectors for each row, which allowed us to capture the semantic meaning of the text. This method was particularly useful for handling the varying lengths of text data.

5.1. Extra Feature Engineering – LaBSE & XLM-R

We implemented the LaBSE (Language-agnostic BERT Sentence Embedding) model from Sentence Transformers to generate embeddings for the *description* and *host_about* columns. LaBSE is designed for multilingual tasks and can handle 109 languages, providing language-agnostic sentence embeddings useful for capturing semantic similarities. Additionally, we used the XLM-R (XLM-RoBERTa) model, a transformer-based model optimized for cross-lingual understanding, pre-trained in 100 languages. XLM-R excels in multilingual text classification by capturing complex linguistic patterns. For both models, we combined the embeddings with numerical features to create comprehensive feature sets for our classification tasks. Additionally, we applied oversampling and undersampling techniques to balance the dataset, ensuring robust analysis and performance in the models.

6. Classification Models

We trained our models using both undersampling and oversampling techniques, though the implementation for each follows a similar approach, differing only in the respective features.

To train our models, we defined various dictionaries where the key represents the feature engineering type (BoW, TF-IDF, GloVe, LaBSE, and XLM-R). We started by initializing models with the desired algorithm, with the default parameters. The decision not to apply Grid Search to all the models was due to the computational expenditure it would require. As such, with the parameters chosen, we would first strive to find the best models and only after that apply the Grid Search on them to maximize performance. We then defined the training and validation features for each type of feature engineering. For the training process, we used a function called *evaluate_model* to train,

predict, and calculate various metrics. The function works as follows: the model is trained on the training data, predictions are made on both the training and validation sets, key performance metrics (accuracy, F1 score, precision, recall) are calculated for both the training and validation sets, and these metrics are returned in a structured format for easy comparison.

6.1. Extra Classification Models – Catboost & LightGBM

To complete the usual models, Logistic Regression, MLPClassifier, KNN, SVM, Random Forest, GBM, Adaboost and XGBoost we decided to apply Catboost and LightGBM. CatBoost, developed by Yandex, is an advanced gradient boosting algorithm known for its ability to handle categorical features directly, reducing preprocessing time and enhancing accuracy while mitigating overfitting. LightGBM, created by Microsoft, is another gradient boosting algorithm optimized for speed and efficiency. It employs a histogram-based approach to quickly find optimal splits, making it ideal for large datasets with lower memory consumption and faster training times.

6.2. Undersampling

The results on undersampling for various models and embeddings show some clear trends. Models like Logistic Regression, MLPClassifier, KNN, and SVM perform worse on validation scores compared to others. This suggests that these models might not be the best fit for our data or the undersampling method we used. It could be due to their simplicity, like with Logistic Regression, or because they're sensitive to imbalanced data, as is the case with KNN.

We see a significant overfitting problem in models like Random Forest, XGBoost, and LightGBM. These models perform almost perfectly on the training data but significantly worse on validation data. This means they're capturing the training patterns very well but failing to generalize to new data.

On the bright side, Gradient Boost, AdaBoost, and CatBoost show more balanced performance with higher validation scores, meaning they generalize better than the other models. A closer look shows Gradient Boost with validation F1 scores between 0.876 to 0.882 across different embeddings, indicating solid and consistent performance. AdaBoost has slightly lower scores but still strong results, between 0.857 to 0.864. CatBoost matches Gradient Boost with validation F1 scores also ranging between 0.876 to 0.882.

For embeddings, traditional text representations like BoW and TF-IDF generally perform well, with BoW often slightly better. Modern embeddings like GloVe, LaBSE, and XLM-R perform strongly, especially with more advanced models like CatBoost and Gradient Boost. GloVe and XLM-R show consistently higher F1 scores across several models, indicating they capture semantic information effectively.

6.3. Oversampling

The analysis of the oversampling results reveals some different patterns. As expected, the simpler models like Logistic Regression, MLPClassifier, KNN, and SVM consistently show the lowest F1 validation scores, making them less suitable for further work. These models often show overfitting, with training accuracy and F1 scores close to 1, while validation accuracy and F1 scores are significantly lower, around 0.77. However, there are exceptions, such as the SVM with TF-IDF, which achieves validation scores of 0.86 and training scores of 0.96.

The Random Forest model displays decent validation scores, with F1 scores ranging from 0.818 to 0.837 depending on the embedding used, but it suffers from overfitting, indicated by its training scores being close to 1. This pattern suggests that while the model learns the training data very well, it struggles to generalize to new data.

Moving on to the boosting models, XGBoost and LightGBM show training scores of 1 and validation scores around 0.88. For instance, XGBoost with TF-IDF achieves a validation F1 score of 0.879, while LightGBM with BoW shows a validation F1 score of 0.881. These models are overfitted, as their training scores are perfect or near-perfect. CatBoost also falls into this category but with slightly less overfitting. Some embeddings for CatBoost result in training scores of 0.92, such as GloVe, which has a training F1 score of 0.928 and a validation F1 score of 0.881.

Given these observations, the most promising models left to work with are Gradient Boosting CatBoost. These models do not exhibit the same level of overfitting and maintain a better balance between training and validation scores. For example, Gradient Boost shows validation F1 scores between 0.881 and 0.885 across different embeddings, while AdaBoost achieves validation F1 scores between 0.862 and 0.871 and CatBoost with scores between 0.884 and 0.888.

In terms of embeddings, traditional text representations like BoW and TF-IDF generally perform well, with BoW often slightly better. For instance, Gradient Boost with BoW achieves a validation F1 score of 0.882, while TF-IDF achieves 0.881. Modern embeddings like GloVe, LaBSE, and XLM-R also perform strongly, especially with more advanced models like CatBoost and Gradient Boost. GloVe and XLM-R show consistently higher F1 scores across several models, indicating their effectiveness in capturing semantic information.

6.4. Grid Search

We conducted a grid search to enhance the performance of the Gradient Boosting Classifier for both undersampling and oversampling and Catboost for oversampling only, focusing on various hyperparameters. For the Gradient Boosting Classifier we performed grid search for the following parameters: *n_estimators* = [100, 200]; *learning_rate* = [0.05, 0.1]; *max_depth* = [5, 7]; *min_samples_split* = [5, 10]; *min_samples_leaf* = [1, 4]. And for Catboost, the following ones: *iterations* = [100, 200]; *learning_rate* = [0.05, 0.1]; *depth* = [4, 6]; *l2_leaf_reg* = [3, 5].

In the undersampling case, we explored different models employing some feature engineering techniques. For each model, we initiated a GridSearchCV instance with a predefined parameter grid. Utilizing 4-fold cross-validation, we systematically searched for the best combination of hyperparameters. Subsequently, we applied the identified optimal parameters to train the model on the undersampled data and assessed its performance using our '*evaluate_model*' function.

Across various techniques, the best parameters found kept the results consistent for both Gradient Boosting Classifier and Catboost (Figure 20 - GridSearch Results).

	Model		Train				Validation			
			Accuracy	F1	Precision	Recall	Accuracy	F1	Precision	Recall
Undersampling	GBM	BoW	0,892	0,892	0,892	0,892	0,875	0,877	0,881	0,875
		TF-IDF	0,996	0,996	0,996	0,996	0,874	0,877	0,881	0,874
		GloVe	0,999	0,999	0,999	0,999	0,877	0,879	0,882	0,877
		LaBSE	0,999	0,999	0,999	0,999	0,874	0,876	0,880	0,874
		XLM-R	0,999	0,999	0,999	0,999	0,875	0,877	0,879	0,875
Oversampling	GBM	BoW	0,956	0,956	0,956	0,956	0,887	0,888	0,889	0,887
		TF-IDF	0,971	0,971	0,971	0,971	0,882	0,883	0,885	0,882
		GloVe	0,999	0,999	0,999	0,999	0,872	0,872	0,871	0,872
		LaBSE	1,000	1,000	1,000	1,000	0,882	0,882	0,882	0,882
		XLM-R	0,998	0,998	0,998	0,998	0,882	0,882	0,883	0,882
	CatBoost	BoW	0,920	0,920	0,920	0,920	0,881	0,883	0,886	0,881
		TF-IDF	0,931	0,931	0,932	0,931	0,885	0,886	0,889	0,885
		GloVe	0,983	0,983	0,983	0,983	0,883	0,884	0,885	0,883
		LaBSE	0,988	0,988	0,988	0,988	0,888	0,889	0,890	0,888
		XLM-R	0,983	0,983	0,984	0,983	0,882	0,883	0,885	0,882

Figure 20 - GridSearch Results

6.5. Cross Validation & Confusion Matrix

The model that retained the best performance was the Gradient Boosting Classifier with Oversampling. This model showed very similar results to Catboost and slightly outperformed the undersampling case, yielding validation F1 scores around 0.88. Notably, GloVe, LaBSE, and XLM-R exhibited significantly higher training accuracy and F1 scores compared to BOW and TF-IDF, suggesting significant overfitting on the training data.

Importantly, we performed cross-validation with 3 folds, which helps ensure that the model's performance is robust and not dependent on a particular train-test split. Cross-validation provides a more reliable estimate of the model's performance by averaging the results across multiple train-test splits.

From the confusion matrix (Figure 21 - Confusion Matrix), we can observe that the Gradient Boosting Classifier with Oversampling (BOW) correctly classified 824 instances of class 0 and 280 instances of class 1. There were 84 instances of class 0 that were misclassified as class 1, and 62

instances of class 1 that were misclassified as class 0. This indicates a good balance in identifying both classes but still shows room for improvement in reducing false positives and false negatives.

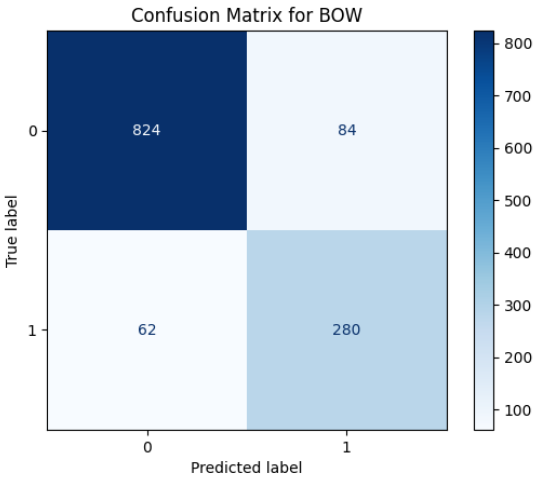

Figure 21 - Confusion Matrix

The evaluation metrics further support these observations, with the model achieving a Train Accuracy of 96.32%, Train F1 of 96.34%, Train Precision of 95.83%, and Train Recall of 96.88%. On the validation set, the model achieved a Validation Accuracy of 92.52%, Validation F1 of 92.75%, Validation Precision of 90.05%, and Validation Recall of 95.62%. These results highlight the effectiveness of the Gradient Boosting Classifier with Oversampling in handling imbalanced datasets, leading to robust performance across both training and validation sets.

7. Evaluation of the Test Set

Since Gradient Boost Machines achieved better performance in Grid Search, we chose this model to evaluate on the test set with Bow. The parameters used were learning rate=0.1, max_depth=7, min_samples_leaf=4, min_samples_split=5 and n_estimators=200. Given that overfitting is an important factor when we choose the model, Bow demonstrates less overfitting than the other techniques used like TF-IDF, Glove, LaSBE and XLM-R.

For the evaluation, we applied the same preprocessing techniques as for the training data, in order to make the data as similar as possible. The Gradient Boosting model with Bow was applied in order to be able to make the predictions for the labels on this data and then we saved them to a csv containing only two columns, an id column related to the properties and a column for the predictions.

8. Further Work

In our previous analysis, we aimed to identify the most effective models and feature engineering techniques, achieving notable success with Gradient Boosting Machines. However, we encountered challenges in accurately auto-classifying sentences by language, particularly during the initial data exploration phase. To enhance our work, it is crucial to improve the accuracy of auto-classifying all sentences by language. Implementing techniques like part-of-speech tagging and dependency

parsing can provide a deeper understanding of each sentence's linguistic context, thereby enhancing classification accuracy.

We could have initially performed a parameterization on all models to know which one was truly the best, rather than using the default settings and only later doing a grid search for the one we thought was the best. This approach might have provided a clearer comparison across all models. Ensemble methods, which combine predictions from multiple models, could enhance overall performance. Doing a thorough error analysis to understand the types of sentences that are misclassified would give insights into specific areas needing improvement, making our classification strategy better.

Additionally, domain adaptation techniques could be explored to fine-tune the model for specific contexts, such as Airbnb reviews. This would improve the model's ability to handle the unique language and expressions used in this domain. Integrating user feedback mechanisms could also be helpful, allowing the model to continuously improve based on corrections and suggestions from users.

By addressing these areas and focusing on the auto-classification of all sentences, we could aim to develop a more accurate sentence classification system. This would allow us to analyse the full dataset more effectively, leading to deeper insights and more actionable recommendations for improving the Airbnb experience based on user feedback.

9. Conclusion

To summarize, our project was designed to forecast the future availability of Airbnb properties through the analysis of their descriptions and related data. By examining factors such as property descriptions, host profiles, and guest feedback, we aimed to provide insights into whether a listing would remain accessible in the upcoming period. We cleaned and processed the text data, ensuring clarity and removing any unnecessary elements. Additionally, we employed advanced techniques to extract meaningful information from the text, such as identifying important entities and understanding the context better.

Upon analysing the data, we identified several machine learning models that showed promise in predicting listing availability. Models like the Gradient Boosting Classifier and CatBoost exhibited strong performance. Through parameter optimization and fine-tuning, we further improved the accuracy and reliability of these models. Our rigorous testing process involved evaluating the best-performing model on new data, which yielded positive results. This validation indicates the effectiveness and reliability of our approach in predicting the availability status of Airbnb properties.

Ultimately, our project contributes valuable insights to both hosts and guests in the Airbnb community. By leveraging data-driven predictions, users can make more informed decisions when selecting and booking accommodations. This empowers them to optimize their experiences and enhances the overall satisfaction of the Airbnb platform.

10.Annex

	unlisted	word_count	char_count	average_word_lenght	stopwords_count	stopword_ratio
count	6248.000000	6248.000000	6248.000000	6248.000000	6248.000000	6248.000000
mean	0.273367	132.119878	816.135083	5.365817	46.853073	0.341740
std	0.445724	47.742357	280.150662	0.926862	20.386065	0.081594
min	0.000000	3.000000	12.000000	3.333333	0.000000	0.000000
25%	0.000000	99.000000	629.750000	4.946429	32.000000	0.306642
50%	0.000000	156.000000	1000.000000	5.205882	51.000000	0.354037
75%	1.000000	166.000000	1000.000000	5.523260	62.000000	0.393939
max	1.000000	196.000000	1000.000000	21.000000	99.000000	0.588235

Figure 2 - Summary of Description Column

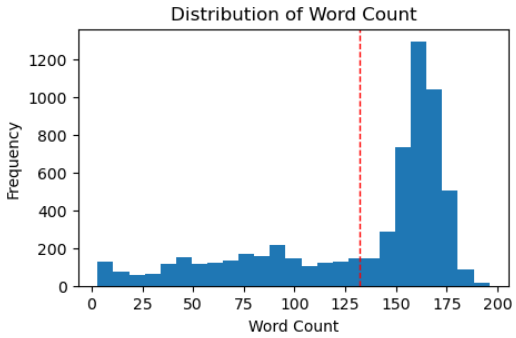

Figure 3 - Word Count of Description Column

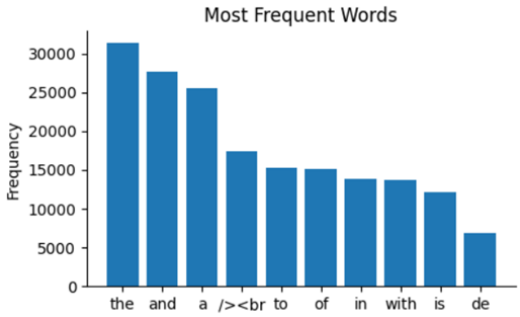

Figure 4 - Most Frequent Words of Description Column

	unlisted	word_count	char_count	average_word_lenght	stopwords_count	stopword_rate
count	6248.000000	6248.000000	6248.000000	6248.000000	6248.000000	6224.000000
mean	0.273367	75.194942	444.008163	5.090609	31.055858	0.398516
std	0.445724	81.384186	505.223612	2.699575	29.804355	0.126985
min	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	0.000000	26.000000	151.000000	4.448049	11.000000	0.368421
50%	0.000000	56.000000	325.000000	4.857143	24.000000	0.427673
75%	1.000000	103.000000	604.000000	5.201754	42.000000	0.474895
max	1.000000	1408.000000	8683.000000	113.000000	324.000000	0.727273

Figure 6 - Summary of Host_about Column

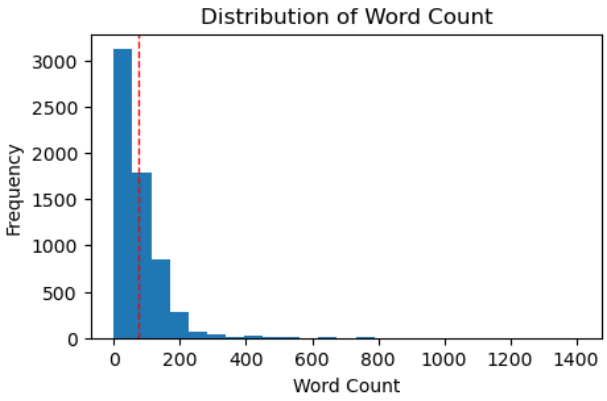

Figure 7– Word Count of Host_about Column

Figure 9 – Word Cloud of Host_about Column

	unlisted	word_count	char_count	average_word_lenght
count	6248.000000	6248.000000	6248.000000	6248.000000
mean	0.273367	79.901088	545.925896	5.910205
std	0.445724	28.776744	193.681379	0.678340
min	0.000000	2.000000	10.000000	3.600000
25%	0.000000	61.000000	416.000000	5.544053
50%	0.000000	91.000000	630.000000	5.850287
75%	1.000000	99.000000	680.000000	6.201974
max	1.000000	164.000000	882.000000	25.666667

Figure 10 - Summary of Description Column

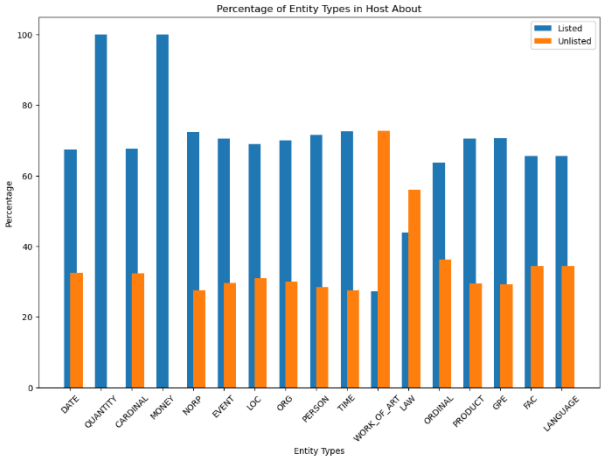

Figure 14 - Percentage of Occurrences in Descripion per Entity Type

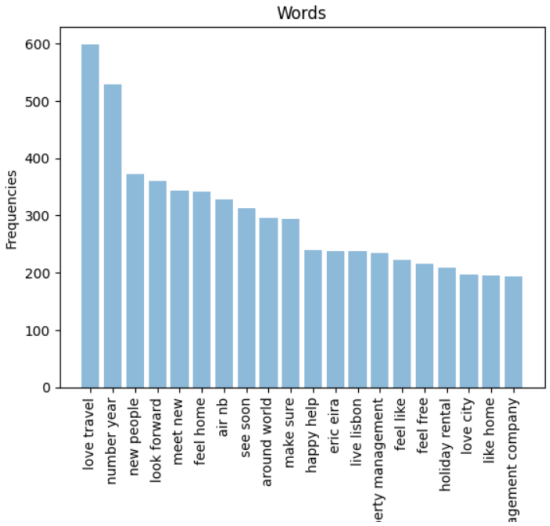

Figure 17 - Bi-gram for Host_about

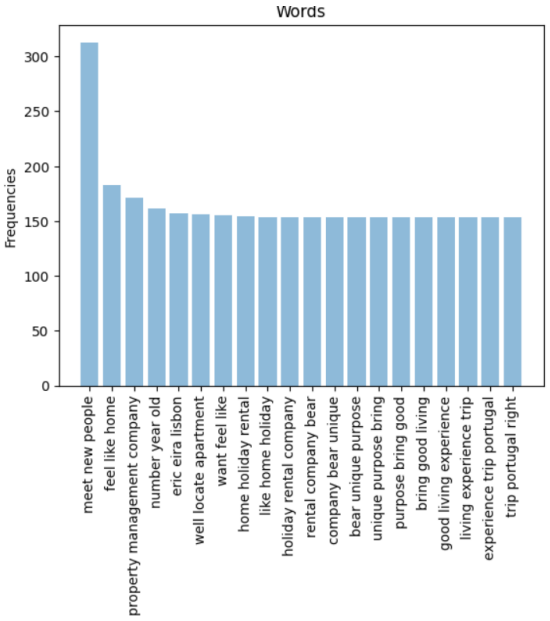

Figure 19 - Bi-gram for Host_about

11. References

Anderson, Derek. “Word Ninja.” *GitHub*, 22 Feb. 2023, github.com/keredson/wordninja.

“CatBoost - State-of-The-Art Open-Source Gradient Boosting Library with Categorical Features Support.” *Catboost.ai*, 2019, catboost.ai/.

“Home.” *Insideairbnb.com*, insideairbnb.com/.

“Lemmatizer · SpaCy API Documentation.” *Lemmatizer*, spacy.io/api/lemmatizer.

“Sentence-Transformers/LaBSE · Hugging Face.” *Huggingface.co*, huggingface.co/sentence-transformers/LaBSE. Accessed 8 June 2024.

“Welcome to LightGBM’s Documentation! — LightGBM 3.2.1.99 Documentation.” *Lightgbm.readthedocs.io*, lightgbm.readthedocs.io/en/latest/index.html.

“XLM-RoBERTa.” *Huggingface.co*, huggingface.co/docs/transformers/model_doc/xlm-roberta

“Named Entity Recognition.” *GeeksforGeeks*, 27 May 2021, www.geeksforgeeks.org/named-entity-recognition/.