Text Mining - Project Report

Group 2

Natural Language Processing as a Tool for Predicting Airbnb Listing Availability

Authors:

Guilherme Sá	20230520
Helena Mashayekhi	20230561
Raquel Rocha	20230188
Sebastião Rosalino	20230372

NOVA IMS

Academic Orientation: Bruno Jardim and Miguel Cruz

2023/2024

Contents

1 Introduction

The explosive growth of the sharing economy, exemplified by platforms like Airbnb, has significantly transformed the hospitality industry. As more property owners leverage these platforms to monetize their houses, understanding the dynamics of property listings becomes crucial. This project aims to harness the power of Natural Language Processing (NLP) to predict whether a property listed on Airbnb will be unlisted in the next quarter. This predictive capability is essential for hosts and the platform itself, offering insights that can drive decision-making, enhance user experience, and improve platform strategies.

In this project, we utilize a comprehensive dataset provided by Airbnb, which includes detailed property descriptions, host biographies, and guest reviews. The challenge lies in analyzing this vast corpus of textual data to discern patterns and indicators that might signal a property's likelihood of being unlisted. By leveraging advanced NLP techniques and machine learning models, we aim to build a robust classification model that can accurately predict the listing status of properties.

2 Data Exploration

The dataset of Airbnb listings was provided in the form of two train and two test datasets: "train_reviews", "train", "test_reviews" and "test".

The test and train sets comprise several features: an index that was renamed to property id, the description of the property, the host description and the binary unlisted label (only for train), which serves as the target variable. The test-reviews and train-reviews datasets comprise only the index, which again was renamed to property id, and the guests' comments about the property.

The size of each dataset is different, as seen in Table 2. Exploring the distribution of our target variable, it was apparent that 72.7% of the properties remained listed, Figure 1. Notable findings were, that whenever the reviews contained the expression "super host", 97.91% of the time the respective property remained listed. Furthermore, it could be observed that the distribution of review counts per property is heavily right-skewed, indicating that most properties have a relatively small number of reviews, Figure 2. The highest frequency is observed for properties with 0 to 50 reviews. This suggests that many properties have not received any reviews or have received very few. When looking at the distribution of review length (number of characters) a similar right-skewedness can be observed, Figure 3. Additionally, words such as "apartment", "Lisbon", and "stay" stood out in the word cloud, as seen in Figure 4. The top 10 most frequent words were all stop words, Figure 6. The top 10 least frequent words included unique words characterized by the presence of characters, numbers and emojis, Table 3.

The analysis revealed a significant relationship between the presence of reviews and the unlisted status of properties. Specifically, properties that lacked reviews were predominantly still listed, whereas those with reviews had a higher likelihood of being unlisted. As depicted in Figure 5, the vast majority of properties without any reviews remain listed. In contrast, properties that have received reviews exhibit a more balanced distribution between the listed and unlisted statuses. This suggests that having guest reviews might correlate with a higher probability of a property being unlisted, possibly indicating issues such as negative guest experiences or other factors leading to the property's removal from the platform.

During the exploration, it was observed that approximately 4,000 properties in the train dataset had no reviews. Among these properties, the majority remained listed, while only a small fraction turned unlisted, suggesting a potential correlation between the presence of reviews and the listing status of properties. The reviews were written in a variety of languages, indicating the global reach and diverse user base of Airbnb. The most common languages identified were English (232,332 reviews), French (53,262), Portuguese (22,447), Spanish (19,234), and German (14,209). Other languages included Italian, Dutch, Russian, Korean, Chinese (Simplified), and many more, reflecting the multilingual nature of the dataset. Notably, there were 918 reviews classified as 'Unknown' in language.

Upon closer examination, it was found that reviews marked as 'Unknown' were predominantly short comments consisting of emojis or brief ratings such as "5/5", indicating that these entries were either non-textual or too brief to be classified accurately by language detection algorithms.

Furthermore, our analysis identified 124 repeated reviews within the dataset. We decided not to remove these repetitions as it is a common scenario for multiple people to leave similar short comments like "good" or "bad" for the same property. This decision was made to reflect a more realistic representation of user reviews, in order to not lose any data and to capture all the guests' feelings about their stay.

3 Data Preprocessing

3.1 EXTRA: Expanding Contractions

To begin with thorough data preprocessing, we decided to examine the text for any contractions. Since contractions were found, we expanded them by converting contracted forms of words (e.g., "don't" to "do not") into their expanded versions. This expansion was applied to the property description and host about columns in both the train and test datasets, as well as in both reviews datasets. Notably, the expansion of contractions was performed only for the English language, which constitutes approximately 64% of all the reviews.

3.2 EXTRA: Converting Emojis to Text

After exploring the datasets and noticing that most emojis convey a certain connotation that could provide insight into the perception of a stay, we decided to convert emojis to their textual meaning. This conversion was applied to all textual columns (comments, description, and host_about) in both the train and test datasets. By doing so, we ensured that the further Feature Engineering techniques could correctly capture the meaning of the use of certain emojis, thereby retaining all emotional information for the classification models.

3.3 Stop Words Removal and Punctuation Removal

Previous exploration revealed that multiple stop words were among the most frequent words. Therefore, we removed stop words for all 29 languages supported by NLTK, totaling 7,130 stop words. This was applied to all textual columns ('comments', 'description', 'host_about') in both the train and test datasets.

Additionally, we removed punctuation, HTML tags, and other characters, such as underscores from the same columns. This process included tokenization and conversion to lower case letters to ensure consistency. Removing stop words and punctuation reduces the size of the vocabulary, focusing on meaningful words and enhancing model generalizability.

3.4 EXTRA: Translation of Property Descriptions and Host About to English

To ensure compatibility with GloVe Embeddings, which only support English tokens, we translated the 'description' and 'host_about' columns in both the train and test datasets to English. This process took 2 hours. Although we considered translating all comments to English as well, we abandoned the idea due to the extensive time it would require.

3.5 Regular Expressions

Regular expressions were employed to standardize specific elements within the textual features. They enabled us to replace certain types of words, such as numbers, dates, weekdays, months, and geographical names, with corresponding default tokens.

Regular Expression usages:

- Entirely numeric strings with the token 'number'.
- Common date formats (e.g., 'MM-DD-YYYY') with the token 'date'.
- Weekday names (e.g., 'Monday') with the token 'weekday'.
- Month names (e.g., 'January') with the token 'month'.
- Country and geographical region names with the token 'geographical'.

This process helped to reduce the size of the vocabulary, hence decreasing the dimensionality of the dataset for the classifications models by focusing on the semantic content of the text rather than on specific numeric or date details.

3.6 Lemmatization and Stemming

In all textual columns of both train and test datasets, we applied lemmatization and stemming to standardize the vocabulary. This preprocessing aimed to reduce the dimensionality of the dataset and enhance the models' generalizability. Lemmatization was performed using language-specific NLP models for the seven most popular languages used for reviews in the dataset (English, German, Portuguese, French, Spanish, Italian and Dutch), converting words to their base forms. Stemming was carried out using the SnowballStemmer for 15 supported languages, producing word stems. For each textual field, both columns containing the lemmatized and stemmed forms of the words were stored.

3.7 Creating the Review Length Feature

In this step, we added a new column named 'review_length' to both the train reviews and test reviews datasets. This column contains the length of each review in tokens. The purpose of this feature is to

provide valuable insights for further data exploration and to serve as a potentially important feature for the classification model. By quantifying the length of reviews, we aim to capture additional context and nuances that could enhance the predictive performance of our models.

3.8 More Data Exploration

To gain deeper insights into our data, we performed several exploratory analyses:

a) Distribution of Review Length: We plotted a histogram to visualize the distribution of review lengths by the number of tokens. This helped us understand the general length of reviews and identify any patterns in the data, Figure 7.

b) Frequency of Regular Expression Default Tokens: We counted and plotted the frequency of the default tokens produced by the Regular Expression, such as 'number', 'date', 'weekday', 'geographical', and 'month' in the comments. This analysis helped us understand the prevalence of these normalized tokens in our data, Figure 8

c) Top 10 Most Frequent Words in Lemmatized Reviews: We identified and plotted the top 10 most frequent words in the lemmatized reviews, giving us a snapshot of the common terms used in guest feedback, Figure 9.

d) Word Cloud of Lemmatized Reviews: We generated a word cloud to visualize the most common words in lemmatized reviews, providing an intuitive way to understand the content of the data, Figure 11.

e) Top 10 Least Frequent Tokens in Lemmatized Reviews: We explored the least frequent tokens in the lemmatized reviews to identify potentially rare but significant words.

f) Top 10 Most Frequent Words in Stemmed Reviews: Similar to the lemmatized reviews, we plotted the top 10 most frequent words in stemmed reviews to understand common terms after stemming, Figure 10.

g) Word Cloud of Stemmed Reviews: We generated a word cloud for the stemmed reviews to visually depict the frequency of terms, Figure 12.

h) Top 10 Least Frequent Tokens in Stemmed Reviews: We explored the least frequent tokens in the stemmed reviews to understand rare words.

i) Top 10 Most Frequent Words in Property Descriptions: We plotted the top 10 most frequent words in property descriptions to understand common terms used by hosts in describing their properties, Figure 13.

j) Top 10 Most Frequent Words in Host Descriptions: We identified and plotted the top 10 most frequent words in host descriptions to understand the common phrases and terms used by hosts to describe themselves, Figure 14.

k) Word Cloud of Property Descriptions: We generated a word cloud for property descriptions to visualize common terms in descriptions, Figure 15.

l) Word Cloud of Host Descriptions: We generated a word cloud for host descriptions to visualize common terms used by hosts, Figure 16.

These exploratory steps provided valuable insights into the textual data and helped guide our feature engineering and modeling efforts.

3.9 EXTRA: Sentiment Analysis on Guests' Reviews

We performed sentiment analysis on guests' reviews using a pre-trained Transformer model from Hugging Face, specifically "nlptown/bert-base-multilingual-uncased-sentiment". This model assigns a sentiment score ranging from 1 to 5, where 1 indicates the worst sentiment and 5 indicates the best sentiment.

The analysis was conducted on the lemmatized comments. Since we are using a transformer model, it was crucial to maintain tokens in their original forms, as stemming might reduce words to non-existent forms. We applied the sentiment analysis model to each review and recorded the sentiment scores. This process was also extended to the test dataset.

The sentiment scores provide valuable insights into the guests' perceptions and serve as an important feature for our classification models. This approach enhances our understanding of guest experiences and might improve the predictive power of our models.

3.10 Train/Validation Split

To prepare our data for model training and evaluation, we performed a train/validation split. We started by creating the feature set from the training data by dropping the 'unlisted' and 'property_id' columns. The 'unlisted' column was extracted separately to create the target dataset.

Next, we performed the train/validation split, ensuring the stratification on the 'unlisted' target to preserve the original distribution of the labels. This split resulted in 80% of the data allocated for training and 20% for validation. We then checked the resulting shapes of the training and validation sets to confirm the split and verified that the label distribution in the target variable was maintained in both sets.

Finally, for the test dataset, we created the X_test set by dropping the 'property_id' column to retain only the features needed for prediction. This ensures that our model has consistent input data for evaluation and prediction purposes, avoiding data leakage.

4 Feature Engineering

4.1 TF-IDF for Property Description and Host About using Lemmatization and Stemming

To create our features set, we applied Term Frequency-Inverse Document Frequency (TF-IDF) transformation on the property descriptions and host about columns using both lemmatized and stemmed versions of the text. First, we combined the 'description' and 'host_about' columns for both lemmatized and stemmed versions into single text entries for the training, validation, and test datasets.

For the lemmatized text, we fitted the TF-IDF vectorizer on the combined text from the training data and transformed the training, validation, and test datasets. The resulting TF-IDF matrices had shapes of (4998, 17710) for training, (1250, 17710) for validation and (695, 17710) for test data.

Similarly, for the stemmed text, we fitted the TF-IDF vectorizer on the training data and transformed the validation and test datasets. The TF-IDF matrices for the stemmed text had shapes of (4998, 15697) for training, (1250, 15697) for validation and (695, 15697) for test data.

This approach allowed us to capture the significance of words in different contexts, providing a robust feature set for our classification models.

4.2 Glove Embeddings for Property Description and Host About

It was decided to implement Global Vectors for Word Representation (GloVe) embeddings for the property description and host about columns to obtain the vector representations for the words of these columns. GloVe is an unsupervised learning algorithm for obtaining vector representations for words, and we utilized the GloVe model with 840 billion tokens, developed by Jeffrey Pennington et al. (2015).

For the implementation, GloVe embeddings were set up to 300 DIM, so each word is represented by a 300-dimensional vector. Additionally, two functions were defined, the first one to retrieve the GloVe vectors for a given word and sets words that do not exist in the GloVe embeddings to zero to ensure that every token in the dataset is mapped to a fixed-size vector. And the second one, to create document matrices, where each token is a GloVe vector corresponding to a token in the texts. This is done since the subsequent models expect numerical arrays or vectors. As a final step, the created functions were applied to the relevant columns in the training, validation and test data.

4.3 EXTRA: Sentence Embeddings on the Guests' Reviews, using BERT BASE Multilingual

We utilized the BERT BASE Multilingual Uncased model from Hugging Face to generate sentence embeddings for the guests' reviews (Hugging Face, 2024). The BERT model captures semantic information, providing contextual embeddings for each review. First, we loaded the pre-trained BERT tokenizer and model.

To generate sentence embeddings, we tokenized the original comments text and passed it through the BERT model, extracting the last hidden state. We computed the average of the last hidden state to obtain a single vector representation for each review. This process was repeated for both the training and test review datasets, ensuring that we captured the semantic nuances of the reviews.

The resulting embeddings were used to enhance our feature set, providing a robust representation of the reviews for model training. This approach leverages the powerful language understanding capabilities of BERT to improve the performance of our classification models.

4.4 EXTRA: Sentence Embeddings on the Guests' Reviews, using `XLM-RoBERTa` Multilingual

As an alternative to BERT BASE, we leveraged the XLM-RoBERTa base model from Hugging Face to generate sentence embeddings for the guests' reviews (Hugging Face, 2024). XLM-RoBERTa is a robust multilingual transformer model that provides contextual embeddings, capturing semantic nuances across different languages. We began by loading the pre-trained XLM-RoBERTa tokenizer and model.

To generate sentence embeddings, we tokenized each review and passed the tokens through the XLM-RoBERTa model, extracting the last hidden state. We computed the average of the last hidden state to

obtain a single vector representation for each review. This process was applied to both the training and test review datasets.

This approach ensures that the semantic information contained in the reviews is effectively captured, enhancing our feature set for subsequent model training. The use of XLM-RoBERTa embeddings provides a powerful foundation for improving the generalizability of our classification models.

5 Classification Models

5.1 K-Nearest Neighbour with TF-IDF (Lemmatized Version and Stemmed Version)

For our classification task, we implemented the K-Nearest Neighbour (KNN) Classifier using TF-IDF vectors derived from both lemmatized and stemmed versions of the textual data. We conducted a thorough Grid Search with cross-validation to optimize the hyperparameters for each version. The parameter grid included different values for the number of neighbors (n_neighbors), distance metrics (metric), and weight functions (weights).

For the lemmatized version, the Grid Search revealed that the best parameters were metric='cosine', n_neighbors=5, and weights='distance'. This model achieved a Standard F1-Score in the positive label of 0.53 and a Weighted F1-Score of 0.769 on the validation set.

Similarly, for the stemmed version, the best parameters were identical: metric='cosine', n_neighbors=5, and weights='distance'. The model achieved a Standard F1-Score in the positive label of 0.52 and a Weighted F1-Score of 0.766 on the validation set.

As can be seen in Figure 17 and Figure 18 the confusion matrix slightly improves with the implementation of lemmatization.

Since the KNN model with lemmatized TF-IDF vectors performed just slightly better than the stemmed version, we decided to proceed with the lemmatized textual fields for subsequent classification models. This decision was based on the marginal performance improvement and the added semantic clarity provided by the lemmatized data.

5.2 Logistic Regression with TF-IDF (Lemmatized Version)

A Logistic Regression model using TF-IDF vectors derived from lemmatized textual data was implemented. To optimize the model, we conducted a Grid Search with cross-validation, exploring various hyperparameters including the regularization parameter (C), regularization techniques (penalty), optimization algorithms (solver), and the maximum number of iterations (max_iter). The Grid Search identified the best hyperparameters as C=10, penalty='l2', solver='saga', and max_iter=100. When evaluated on the validation set, the model achieved a Standard F1 score in the positive label of 0.54 and a Weighted F1-Score of 0.78. Furthermore, the confusion matrix in Figure 19 shows some improvements when compared to the KNN models.

5.3 Multi-Input Long short-term memory using GloVe Embeddings

We implemented a Multi-Input Long Short-Term Memory (LSTM) model using GloVe embeddings. First, we determined the maximum length of tokens for the 'description' and 'host_about' columns in the training dataset, finding the lengths to be 260 and 783 tokens, respectively. This information was used to pad the sequences in these columns to ensure consistent input lengths for the model.

To handle missing values, empty entries in the 'description_lemmatized' and 'host_about_lemmatized' features were replaced with zero matrices, matching the GloVe embedding dimension of 300. This preprocessing step was applied to the training, validation, and test datasets.

Next, we padded the sequences for the 'description_lemmatized' and 'host_about_lemmatized' features using the maximum lengths identified earlier, converting them to numpy arrays of type float16 (we tried 32 at first, but we had to lower it back to 16 due to RAM constraints). This prepared the data for input into the LSTM model.

The model architecture included two input layers, each corresponding to the 'description' and 'host_about' features, with masking layers to ignore padding vectors. LSTM layers processed these inputs, and their outputs were concatenated. A dense layer with ReLU activation followed, culminating in an output layer with a sigmoid activation for binary classification. Hyperparameters of LSTM units, dense units, dropout rate, and learning rate were optimized using Optuna.

The model was trained with callbacks for early stopping (stopping the fitting if the validation loss does not improve for 3 epochs), model checkpointing (saving the best model as the validation loss decreases), and learning rate reduction (if the validation loss does not improve for 2 epochs). After training, the best model, as determined by validation loss, was loaded to make predictions on the validation set. The model achieved a Standard F1-Score in the positive label of 0.37 and a Weighted F1-Score of 0.72. The best hyperparameters found were 90 LSTM units, 38 dense units, a dropout rate of 0.133, and a learning rate of 0.024. The confusion matrix for this model indicates again a significant improvement in identifying the correct label for listed properties as can be seen in Figure 20 however gets worse at predicting the correct unlisted property.

5.4 EXTRA: Random Forest using Sentence Embeddings from BERT Multilingual and XLM-RoBERTa, Sentiment Analysis and Number of Reviews

In this section, we enhanced our predictive model by integrating various features derived from sentence embeddings, sentiment analysis, and the number of reviews. We employed sentence embeddings from the BERT Multilingual and XLM-RoBERTa Transformers, alongside sentiment scores for each review and the number of reviews per property.

First, we grouped the reviews by property ID, calculating the average sentiment and stacking the sentence embeddings into matrices for each property. This aggregation was crucial for our classification task, which aims to predict whether a property will remain unlisted in the next quarter.

Next, we imputed missing values in the sentence embeddings matrices with a predefined array of 1,000 values and filled missing sentiment values with zero. This approach was designed to signal to the model which properties had no reviews by inputting high values in the embeddings and a zero in the average

sentiment. This zero value falls outside the typical sentiment scale of 1 to 5, helping the model identify patterns in properties without any reviews.

We computed the average of each column in the sentence embeddings matrices and expanded these averages into 768 new columns, corresponding to the dimensionality of the embeddings. This step ensured that each property had a consistent set of features.

We merged the processed data with the train and test datasets, ensuring consistency and including the target labels for the training dataset. After, we split the training data into training (80%) and validation (20%) sets, stratified by the target variable.

For the classification model, we used a Random Forest classifier. We performed a Grid Search with 3-fold cross-validation to tune the hyperparameters (initially 5 folds were attempted, but reduced to 3 due to time constraints). The best hyperparameters using the BERT Sentence Embeddings were 'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100, and for XLM-RoBERTa they were 'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 300. Both models achieved a Standard F1-Score of 0.77 for the positive label and a Weighted F1-Score of 0.86 on the validation set, suggesting that both sentence embeddings have similar contributions to the performance. This is also supported by the confusion matrix that has the same results for both approaches, Figure 21.

5.5 EXTRA: Stacking Ensemble Classifier

Given that the Random Forest classifiers utilizing Transformer-based sentence embeddings (BERT and XLM-RoBERTa) exhibited equivalent performance, we proceeded with a Stacking Ensemble Classifier using the XLM-RoBERTa embeddings, sentiment analysis, and the number of reviews.

First, we separated the features and target from the dataset and split the data into training (80%) and validation (20%) sets, stratified by the target variable 'unlisted'. Min-max scaling was applied, fitted on the training set, and then transformed on the training, validation, and test sets to avoid data leakage and ensure proper scaling for base models sensitive to the scale, such as K-NN, LR, MLP, and NB.

The base models included in the stacking ensemble were:

- K-Nearest Neighbors (KNN) with all Scikit-Learn's default parameters
- Logistic Regression (LR) with max_iter=1000 and random_state=42, rest with Scikit-Learn's default parameters
- Multi-Layer Perceptron (MLP) with max_iter=1000 and random_state=42, rest with Scikit-Learn's default parameters
- Random Forest (RF) with Scikit-Learn's default parameters and random_state=42
- Gaussian Naive Bayes (NB) with all Scikit-Learn's default parameters
- Support Vector Machine (SVM) with probability=True and random_state=42, rest with Scikit-Learn's default parameters

The stacking ensemble was configured with these base models, using Logistic Regression as the final estimator and 5-fold cross-validation to enhance robustness.

After fitting the stacking model on the training data, the evaluation on the validation set yielded a Standard F1-Score in the positive label of 0.77 and a Weighted F1-Score of 0.86. Since there was no difference in the confusion matrix compared to the last two models, Figure 21 displays the prediction results.

6 Evaluation

In the following table the key metrics of our models such as the Unlisted label (positive class) F1-Score, Weighted F1-Score, Unlisted label Precision, Unlisted label Recall, and Accuracy can be observed:

Metric	F1-Score	Weighted F1-Score	Precision	Recall	Accuracy
KNN Lemmatized	0.53	0.77	0.66	0.44	0.78
KNN Stemmed	0.52	0.77	0.65	0.44	0.78
LR Lemmatized	0.54	0.78	0.74	0.42	0.80
LSTM	0.37	0.72	0.70	0.25	0.77
Random Forests	0.77	0.86	0.70	0.85	0.86
Stacking Ensemble	0.77	0.86	0.70	0.85	0.86

Table 1: Evaluation Table

Since all these 3 models:

1) EXTRA: Random Forest using Sentence Embeddings BERT Multilingual, Sentiment Analysis, Number of Reviews

2) EXTRA: Random Forest using Sentence Embeddings XLM-RoBERTa (base-sized model), Sentiment Analysis, Number of Reviews

3) EXTRA: Stacking Ensemble Classifier

Exhibited the best results (and they all performed equally the same) it was decided to use the EXTRA: Random Forest using Sentence Embeddings XLM-RoBERTa (base-sized model), Sentiment Analysis, Number of Reviews model for predicting on the Test data because it is a simpler model compared to the Stacking Ensemble and performs exactly the same as it's other version using the BERT Sentence Embeddings.

7 Conclusion

We believe that we achieved a robust solution despite significant computational and algorithm size constraints. These challenges motivated us to explore and implement increasingly more efficient text mining techniques, resulting in substantial learning and development throughout the process. The fact that the best results were attained using the models: Random Forest with sentence embeddings from BERT/RoBERTa multilingual, sentiment analysis, and the number of reviews and a Stacking Ensemble, demonstrates the effectiveness of our comprehensive approach.

Our models demonstrated a high capability of capturing and learning patterns of properties that will

be unlisted in the near future, which can be a valuable tool for the business. Predicting unlisted properties can help Airbnb optimize its platform strategies, enhance user experience, and assist hosts in improving their property listings to reduce unlisting rates.

We attribute our good results to the thorough Data Preprocessing and effective Feature Engineering techniques used. Expanding contractions, lemmatization, stemming, and sentiment analysis were crucial steps that improved feature quality. The use of powerful Transformer-based embeddings, particularly, played a significant role in extracting the most out of the dataset. Since the Random Forest models with sentence embeddings from both BERT and RoBERTa achieved the exact same results, we believe both these embeddings extract similar patterns. As a result, we would try different Transformer-based embeddings in future work to explore potentially better alternatives.

For future improvements, we would translate all reviews in both datasets to English, which we estimate would take approximately five days. This would enhance the model's ability to capture user sentiments and deeper characteristics of every property, regardless of the language used. Finally, we would explore fine-tuning Transformer models specifically for our classification task to further enhance model performance.

Ultimately, our efforts resulted in a model that not only predicts the target variable with high accuracy but also provides valuable insights into the factors influencing a property's likelihood of turning unlisted.

8 Bibliography

References

[1] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," *NLP Stanford*, 2015. [Online]. Available: `https://nlp.stanford.edu/projects/glove/`. [Accessed: May 15, 2024].

[2] Hugging Face, "BERT," 2024. [Online]. Available: `https://huggingface.co/docs/transformers/model\_doc/bert`. [Accessed: May 15, 2024].

[3] Hugging Face, "XLM-RoBERTa," 2024. [Online]. Available: `https://huggingface.co/docs/transformers/model\_doc/xlm-roberta`. [Accessed: May 15, 2024].

9 Annex

Appendix A - Dataset Features

Dataset	Number of Records	Features
Train	6248	Property ID, Description, Host About, Unlisted
Train Reviews	361281	Property ID, Comments
Test	695	Property ID, Description, Host About
Test Reviews	41866	Property ID, Comments

Table 2: Dataset Information

Appendix B - Data Exploration Plots

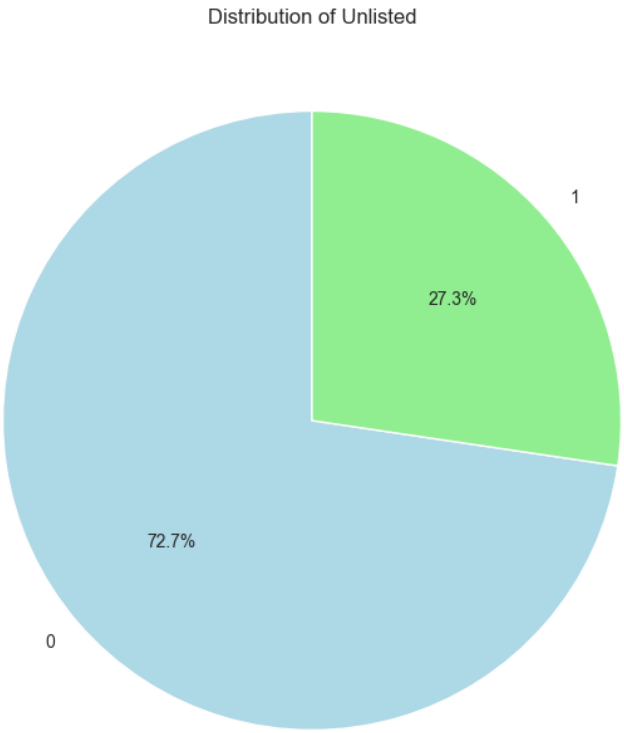

Figure 1: Distribution of Unlisted

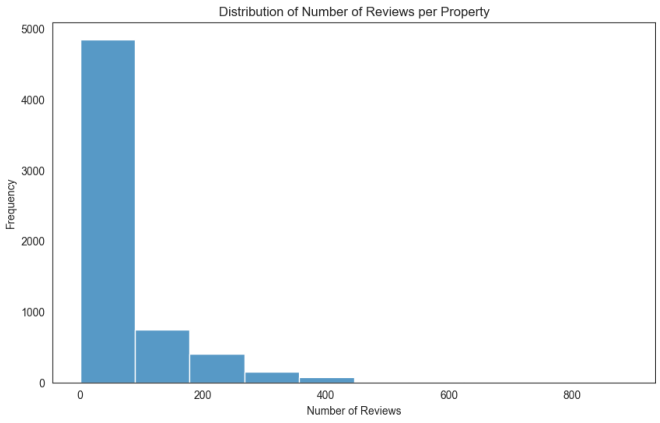

Figure 2: Distribution of Review Counts

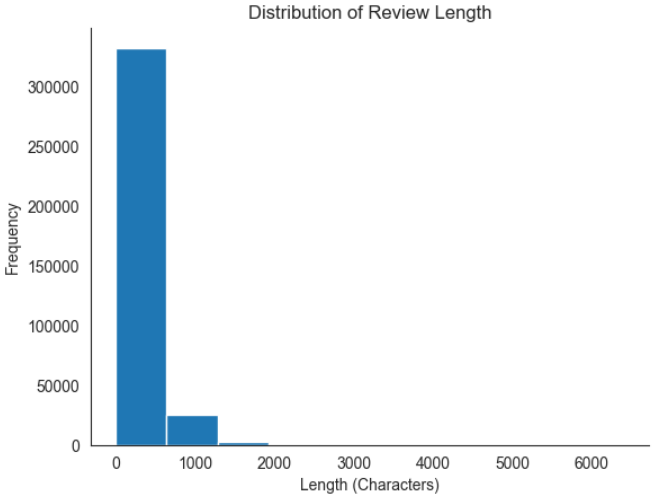

Figure 3: Distribution of Review Length

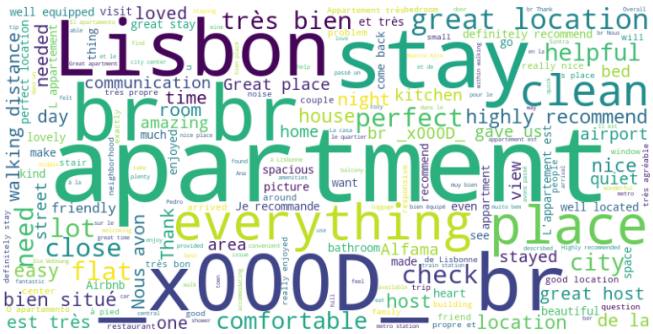

Figure 4: Plotting the word cloud

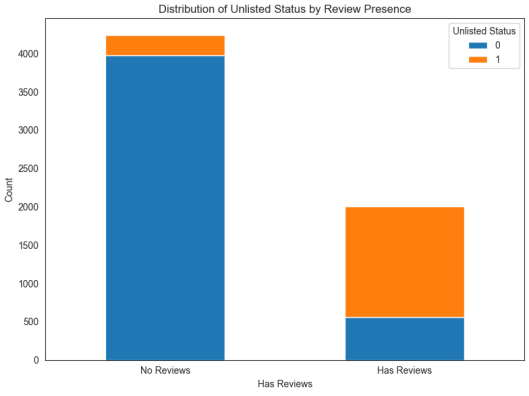

Figure 5: Distribution of Unlisted

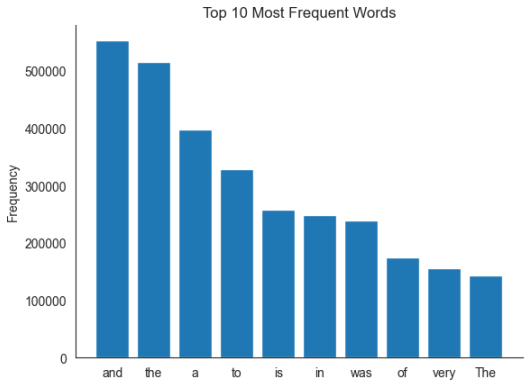

Figure 6: Top 10 Most Frequent Words

Word	Count
kitchen/bathroom/etc.	1
investis_Reviews	1
accomidating. I	1
321	1
excelente...deu	1
...	1
antigüedades	1
tulajdonos	1
segítőkész,	1

Table 3: Top 10 Least Frequent Words

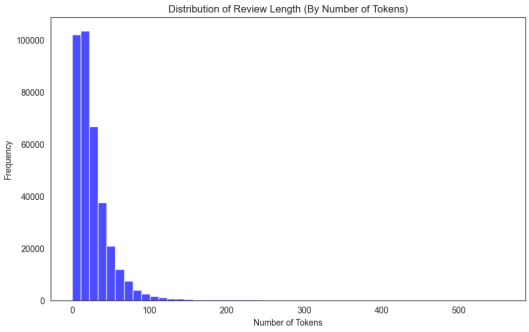

Figure 7: Distribution of Review Length

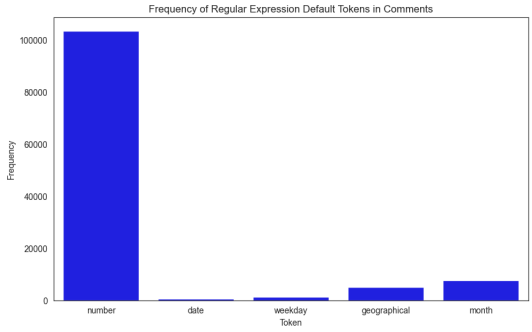

Figure 8: Frequency of Regular Expression Tokens in Comments

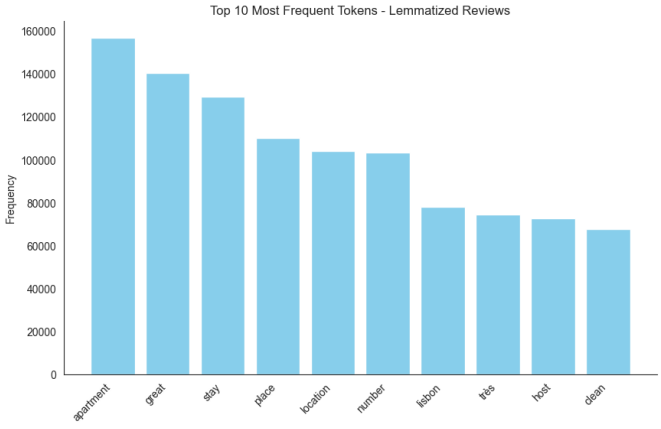

Figure 9: Top 10 Most Freq Lemmatized Words

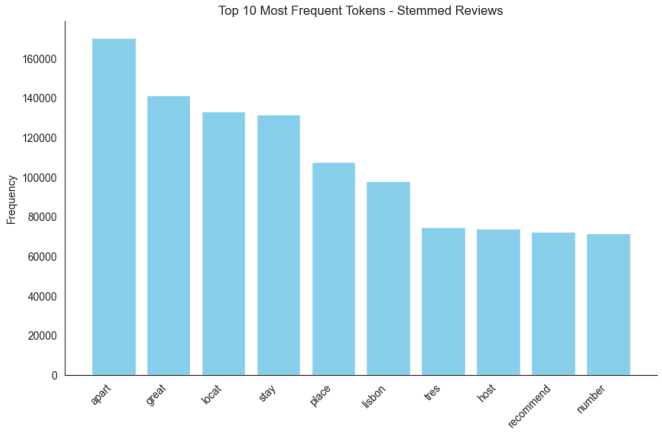

Figure 10: Top 10 Most Freq Stemmed Words

Figure 11: Word Cloud Lemmatized Words

Figure 12: Word Cloud Stemmed Words

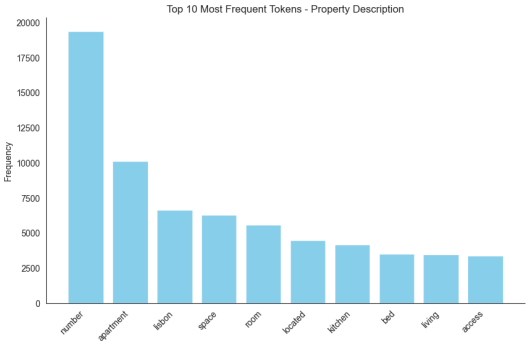

Figure 13: Top 10 Most Frequent Property Description

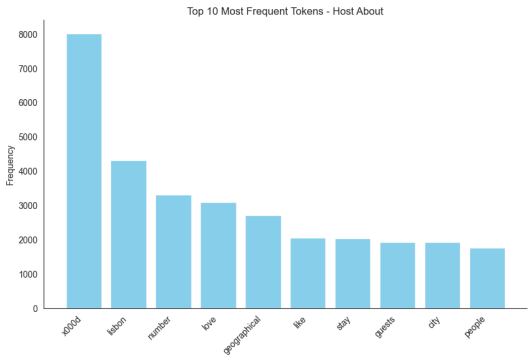

Figure 14: Top 10 Most Frequent Host About

Figure 15: Word Cloud Property Description

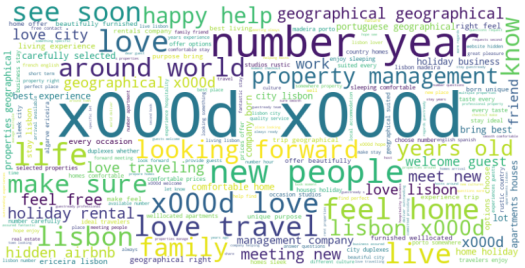

Figure 16: Word Cloud Host About

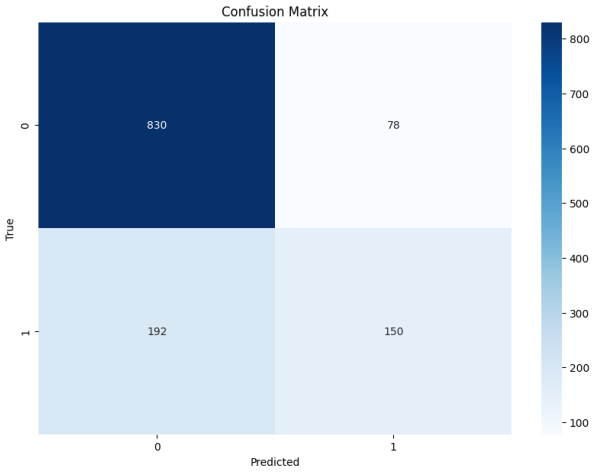

Figure 17: KNN Lemmatized

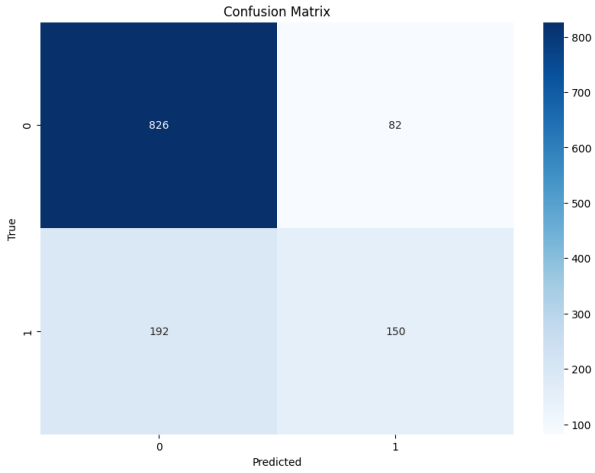

Figure 18: KNN Stemmed

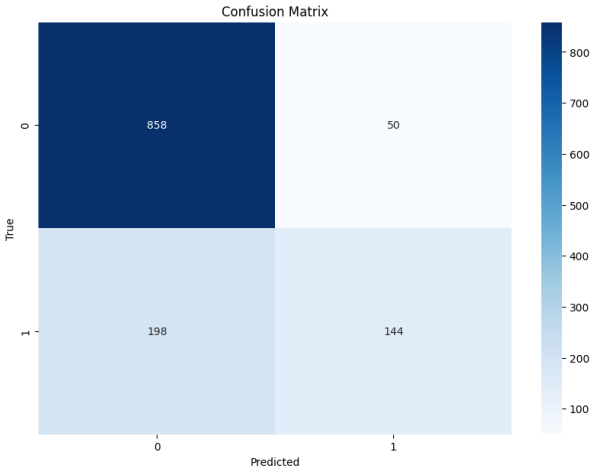

Figure 19: Logistic Regression

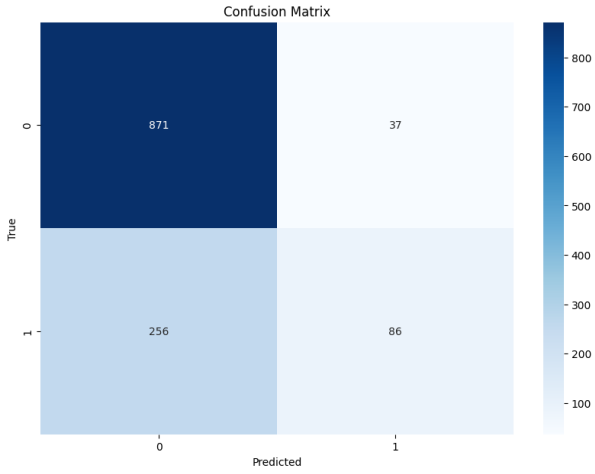

Figure 20: Multi-Input LSTM

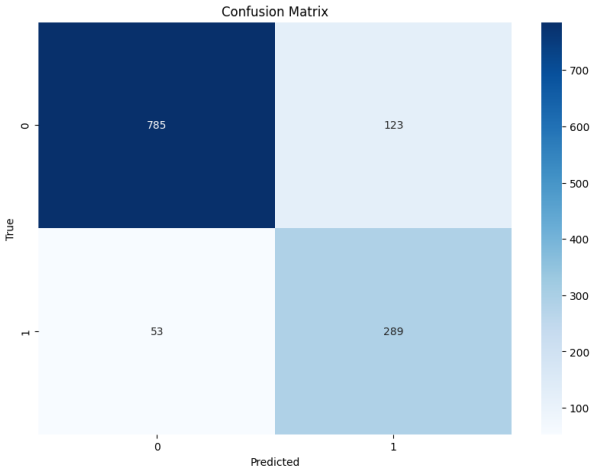

Figure 21: Random Forests and Stacking Ensemble Classifier