NOVA IMS Information Management School

Text Mining Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Group 3

Alex Morales, number: 20220658

Inês Rocha, number: 20220052

Janaina Santos, number: 20220640

Skander Chaabini, number: 20221041

June, 2023

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

I. Introduction

This project aims to build a Natural Language Processing model to predict whether an Airbnb property listing will be unlisted or not in the next quarter.

To help answer the question “Which type of listings are more likely to be unlisted in the next quarter?”, we were given 4 files, 2 of them with historical data and labeled if a listing was unlisted this quarter, where 'unlisted' means listings that were removed from the Airbnb platform.

train.xlsx - This file contains information about the Airbnb listings. Each row represents a listing and contains the description of that listing and a description of the host.

train_reviews.xlsx - This file contains the reviews made in Airbnb about each listing in our dataset. Each row represents a review/comment and it has an index column that corresponds to the listing index number.

The test data files(**test.xlsx**, **test_reviews.xlsx**) possess similar information for unlabeled listings regarding next quarter.

We will build an NLP pipeline that analyzes the text data to identify features correlated with unlisted listings. We will preprocess the text, using techniques used in class, such as, removing punctuation, stop words and tokenization. We will perform sentiment analysis on the review comments to identify negative sentiments and issues that may lead to delisting.

The pipeline aims to determine which factors will contribute to make a listing more likely to be removed.

II. Data Exploration

It is essential to first comprehend the data and get as many insights before preprocessing and modeling. In this part, variable types, missing values, target variable distribution, and text variables were examined. We checked the frequency of words and emojis and plotted word clouds. New features like number of reviews per listing and languages used were calculated and visualized.

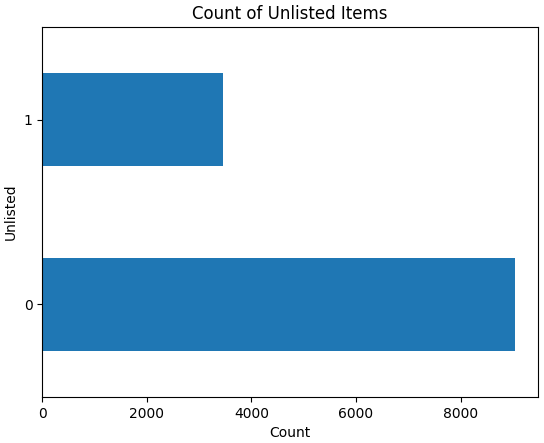
2.1 Missing Values

No missing values were found in any of the columns in the dataframes. However, 35% of the indexes in train_data don’t have any comments, generating missing values by left join.

2.2 Data imbalance

The target variable, 'unlisted', is imbalanced in the train dataset. Only 28% of the observations have an 'unlisted' value of 1. The vast majority, 72%, have remained listed. To

address this class imbalance, we will use strategies like class weights or distance measures that compensate for the uneven distribution of classes.



2.3 Emojis

The 'comments' column contains client feedback. Emojis were vastly found and had to be handled in the preprocessing step to extract information from them. The majority of the emojis used in the reviews dataframe are positive.

This indicates that emojis in the reviews are mostly correlated with positive reviews or that reviews that have emojis in their reviews are more prone to be positive messages in terms of sentiment analysis as Emojis are often used to convey the reviewer's sentiment and emotion. On the other hand, at train and test_data, on descriptions from hosts, few emojis were found. These emojis were more informational, pointing to a location or detail in the listing.

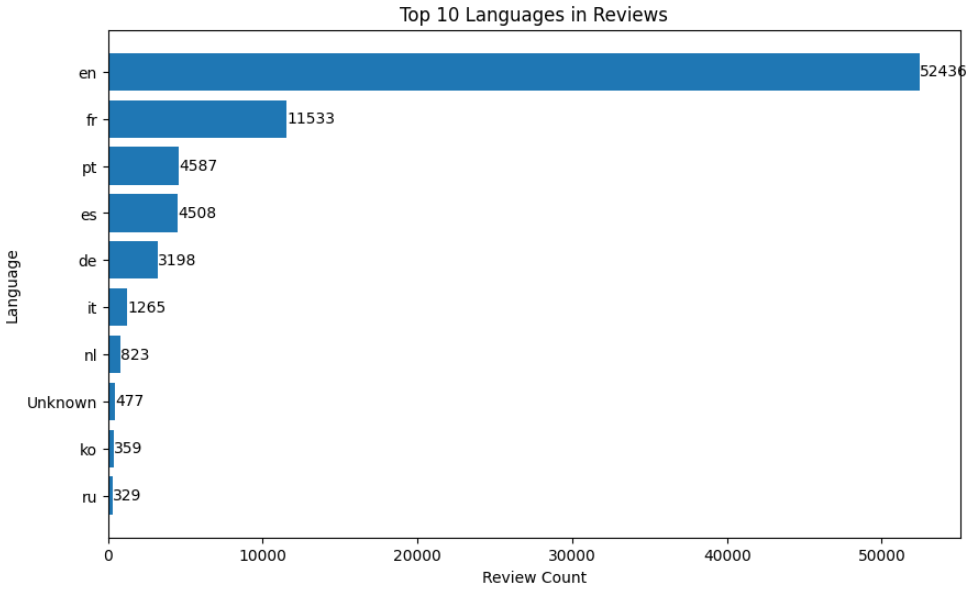
Overall, the test datasets exhibit similarities to the train datasets in terms of emoji usage patterns, with emojis in the reviews used mainly to express positive sentiment, and emojis in the data being used more literally or to convey a positive message.

2.4 Word Cloud

The most common words found both in train and validation, before the preprocessing, were Lisbon and apartment and also "x000D" that is likely an artifact of a text encoding or conversion process.

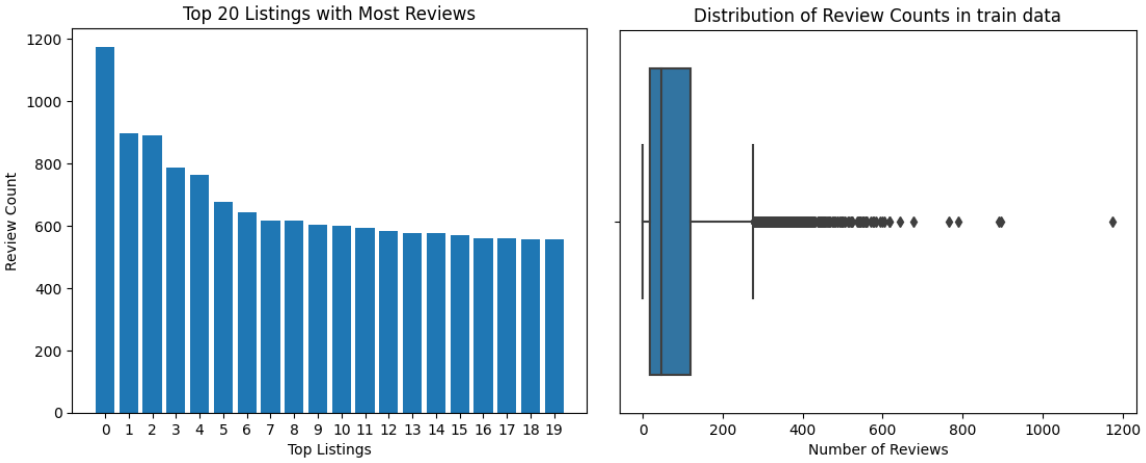
2.5 Languages

We used the language detect library to detect the language of the reviews. As we can see from the following figure from test_data, it is a multilingual dataset, with the main language found in the comments being English followed by French and Portuguese.



2.6 Reviews distribution

After grouping the reviews dataframe by index, we could get the number of reviews per listing. Based on the boxplot, we can see that the number of reviews is distributed between 0 and 300 with a median of 50 reviews.



III. Data Preprocessing

In the workflow, we opted for two approaches. First, focusing on language translation and then data cleaning, so that we have an English only dataset and try one language model on embeddings on them. The second approach proceeds directly to the data cleaning and is then used for multilingual models.

3.1 Data cleaning

We created a single function to perform the data cleaning of our dataset and organize it for subsequent analysis and model training. This function contains the following methods:

- **Stop Words Removal:** This operation eliminates the most common words in a language that do not contain much useful information.
- **Lemmatization and Stemming:** These are techniques used to reduce a word to its base form. Lemmatization takes into account the morphological analysis of the words and returns the lemma or the dictionary form of a word. S
- **Removing Special Characters:** Non-alphanumeric characters are removed from the text, leaving only alphabets and numbers.
- **Removing Single Characters:** Single characters, which are typically not informative, are removed from the text. One example was removing a single point (“.”), which was one of the most common comments from the reviews column.
- **Removing Multiple Spaces:** Extra spaces, possibly introduced by the previous preprocessing steps, are removed from the text.
- **Regular Expressions:** Regular expressions are used to find patterns in the text and replace them. Here it's used to replace currency symbols followed or preceded by numbers with a special token "#COST".
- **Converting to Lowercase:** This step normalizes the text by converting all letters to lowercase, ensuring that the same word in different cases is treated the same.
- **Removing HTML Tags:** Texts extracted from web pages often contain a lot of HTML tags. These are removed as they do not contribute to the meaning of the text.
- **Separating Merged Words:** This is a step where words that are mistakenly joined together are separated. This could be due to errors during data collection.
- **Removing Emojis:** Emojis are converted to text, which could help capture some emotional content, especially for sentiment analysis.
- **Removing Emojis with other formats:** as the current library used for this doesn't seem to fully handle all the emojis in the dataset as it only covers emojis like this “😀” and not like this “:)”, we added extra code to transform it to text.

In addition to the standard preprocessing steps taught in class, we performed the conversion of **emojis to text**, **HTML tags removal**, **translation**, and **sentiment analysis** as extra preprocessing steps to augment the dataset. The translation and sentiment analysis are described below.

3.2 Translation

As mentioned above, the first part of our process involves automatic language detection and translation.

Comments in our dataset, which may be in various languages, are detected and translated into English. This translation process ensures uniformity in our data and enhances our ability to extract meaningful insights during downstream analysis. In the second part of our process, we apply the same preprocessing steps to the translated text as we did in the other approach. One important aspect of our approach is that we use indexing to perform the preprocessing efficiently. Our preprocessing workflow translates and preprocesses the textual data, and saves the preprocessed results to disk as Parquet files. Parquet enables fast and efficient data access as it is a columnar optimized storage file format. By saving the preprocessed data to disk, we can resume the processing from where we left off in case of GPU failure or other issues, without having to redo the translation and preprocessing steps.

3.3 Sentiment Analysis

After the previous steps were completed, the next step is to perform sentiment analysis for the comments column. The library used for sentiment analysis is VADER (Valence Aware Dictionary and sEntiment Reasoner), specifically designed to handle social media texts where informal language, grammatical errors, and the presence of emojis and emoticons are common. In addition to improving our model, sentiment analysis also helps us identify negative sentiments in the reviews, this can help us detect issues of a listing that may lead to delisting in the next quarter.

In the feature engineering section,new features have been created: Negative, Positive, Neutral and negative_rate. Negative_rate calculates the ratio of negative comments to total reviews for each listing.

IV. Feature Engineering

4.1 Creating New Features

In the feature engineering section we first started by creating two new features called “total_reviews” and “negative_rate”.

- **total_reviews:**

  This feature counts the number of reviews that a listing has. A listing with a very high number of reviews may be more likely to remain listed, as it suggests that the listing has been active and popular for a longer period. This depends if they are positive or negative reviews. It could also have the opposite effect of having many negative reviews and being more prone to be unlisted.

  On the other hand, a listing with no reviews could indicate that the listing is either a new addition or possibly a scam listing.

By including this feature in our model, we can better account for the impact of review count on a listing's likelihood of staying listed.

- **negative_rate:**

As mentioned before, this feature calculates the ratio of negative comments to total reviews for each listing. By including this feature in our model, we can better account for the impact of negative reviews on a listing's overall rating. The formula used is shown below:

$$negative\ comments\ rate\ = \frac{\#\ negative\ reviews}{\#\ total\ reviews}\ ,\ per\ listing$$

4.2 TF-IDF

Term Frequency-Inverse Document Frequency, as seen in classes, has as its main advantage its simplicity and easy implementation. By considering the frequency of terms in a document and their significance in the entire collection, TF-IDF provides a direct way to represent documents as feature vectors. However, It disregards word order and semantic relationships, limiting its ability to capture context. It is also sensitive to document length and struggles with uncommon words. This is the reason why we tested other options of embeddings.

4.3 GloVe Embeddings

As seen in practical classes one of the main advantages of GloVe is that, unlike Word2vec, it does not rely just on local statistics but rather incorporates global statistics (word co-occurrence) to obtain word vectors. We used the **glove-wiki-gigaword-50** glove embedding, because it was trained on a very extensive corpus composed of Wikipedia 2014 and Gigaword 5. The large corpus of this specific embedding means that it has coverage of 400,000 vocab words . This wide coverage facilitates its applicability to many natural language tasks. GloVe embeddings have been shown to perform well on tasks involving capturing word semantic and syntactic relationships, such as word analogy and word similarity tasks. The downside of this specific approach is that it's only trained with an English corpus so it was only used in our already translated dataframe.

4.4 Doc2Vec

We performed Doc2vec embedding on our text variables as an extra method for feature engineering. Doc2Vec is an extension of Word2Vec that learns embeddings on a document level. Doc2vec does not have language limitations and can be applied to text data in any language, as it learns directly from the patterns and relationships present in the text corpus, assuming that the meaning of a word is reflected in its context. It represents entire documents, in our case reviews, descriptions, and about hosts as fixed-length vectors by considering the surrounding words in the document. Doc2Vec embeddings can capture the overall context and meaning of a document, enabling them to be used as features for document-level tasks. By applying Doc2Vec, we can derive meaningful representations of documents that enable more accurate analysis and modeling of textual data.

4.5 Multilingual Distilbert Transformer

As our second extra Transformer-based word embedding model, we chose to use the **distilbert-base-multilingual-cased** transformer on our multilingual dataframe.

We chose this transformer because it was pretrained on 104 languages and our dataset contains more than 20 languages and which are the most frequent. According to the hugging face page this model was pretrained with the supervision of bert-base-multilingual-cased on the concatenation of Wikipedia. This extensive multilingual corpus improves the model in learning embeddings that generalize well to our goal. We opted for the distilled version of multilingual Bert because it is faster and has a smaller memory footprint. Its size makes it more feasible to run on typical computing resources.

V. Classification Models

For the classification models section, we proceeded with the implementation of some machine learning models and NLP transformers to find the most suitable classifier for our problem.

From the full dataset, we the data was already splitted into:

- 90% for training
- 10% for testing

From the training dataset (90% of the full data) we splitted into:

- 80% for model training
- 20% for model validation

We used a total of 5 models, these were: Logistic Regression, KNN Classifier, MLP, DistilBERT transformer and DistilRoBERTa transformer.

The model with the highest train and validation accuracies, as well as the highest F1-score on the validation set, will be selected as the best model.

5.1 Logistic Regression

This model was chosen as the model for this problem for a few key reasons. It is a simple and interpretable model that can act as a useful baseline before exploring more complex algorithms. Class weights were crucial given the imbalanced data, and logistic regression allows specifying weights to address this. As a proven classification method, logistic regression often produces strong results, and its probabilistic output lends itself to easy interpretation of predictions. The following weight were attributed to each target value: {0: 0.6903314917127071, 1: 1.8134978229317853}

5.2 KNN

K-Nearest Neighbors was selected for its simplicity and ability to handle non-linear classification tasks by considering the similarity of data points. The following model was selected: KNeighborsClassifier(n_neighbors = 5, metric = 'euclidean',weights='uniform')

When using the distance weights setting, closer neighbors are weighted with greater importance when formulating predictions. Neighbors closer to the point are thought to contribute more critical information for categorization ,therefore their weights are increased.

To optimize the KNN model for our problem, the distance weighting approach is giving more importance to the nearest neighbors - regarding their listing characteristics - this aims to improve the model's ability to classify the relevant listings correctly.

5.3 MLP

Multilayer Perceptron, a type of neural network, was chosen for its capability to capture complex relationships and patterns in the data.

Through this hyperparameter optimization process, the following model was selected: MLPClassifier(solver='lbfgs', hidden_layer_sizes=(6,3,5), activation='relu', random_state=3, warm_start=True, alpha=0.0001)

This model exhibited the best performance on the validation set out of all the various combinations and hyperparameters tested.

5.4 HuggingFace models

As a transfer learning option, it has been decided to test DistilBERT from HuggingFace transformer-based models, pre-trained on large-scale datasets. DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than bert-base-uncased, runs 60% faster while preserving over 95% of BERT’s performances as measured on the GLUE language understanding benchmark.

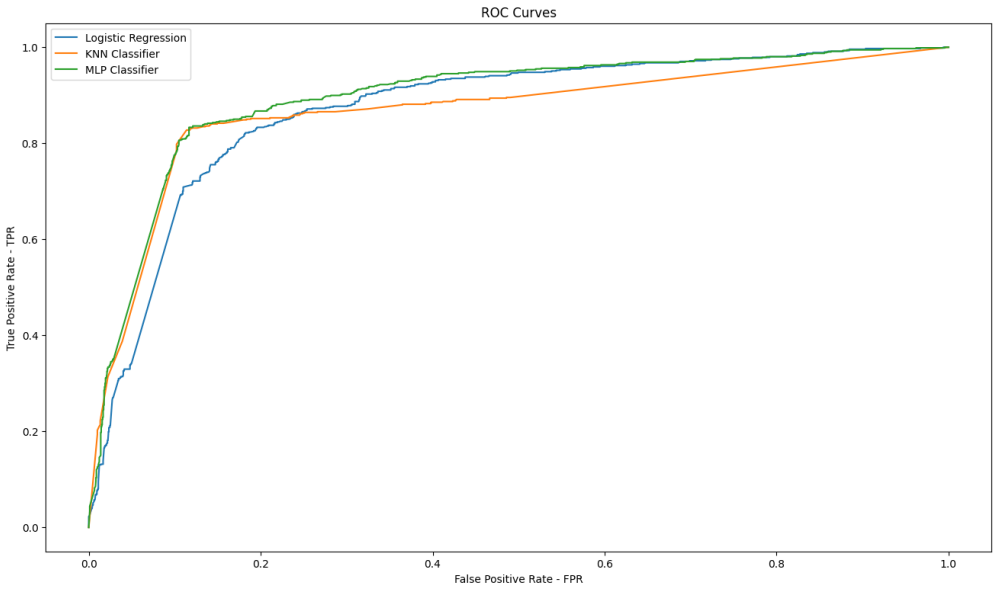
The model only was used with string values, concatenating the datasets and applying the model. Due to computational power, it was not possible to continue with this approach. Time was of the essence and the group decided to work with the previous models. Also, the results were not so promising. Even though the f1 score for label 0 was 0.84, for label 1 was 0.42.

VI. Evaluation and Results

The table below summarizes the performance of the models we compared on our dataset, using the selected parameter values and features.

Models	Train Accuracy	Val. Accuracy	Train F1-Score	Val F1-Score
LR	0.79	0.75	0.70	0.66
KNN	0.97	0.87	0.95	0.77
MLP	0.93	0.87	0.86	0.77

Based on these results, KNN achieved the highest accuracy on the training, and tied on the validation accuracy. We will now use the KNN model to make predictions on the test dataframe, to generate the final submissions file called Predictions_3.csv.



Since the results were so similar for the MLP and the KNN, we chose to plot the roc curves of our models. The AUC (area under the curve) shows us that this model has the best relation between False Positive rate and True Positive rate on top of our good results. As we can infer from the plot the KNN and MLP have identical curves in the top left corner.

While the MLP and KNN performed well based on our results, there is still room for improvement. In future works, we want to:

- Further, optimize the hyperparameters of the MLP and KNN to potentially improve performance.
- Utilize the transformer models not just for generating embeddings, but also for the classification task itself. Our predicting transformer did not perform as well in our initial experiments, as we expected but with proper optimization it may be able to attain better results.

References:

- Inside Airbnb. (n.d.). Retrieved from http://insideairbnb.com
- Kim, T., & Wurster, K. (n.d.). Emoji. Retrieved from https://carpedm20.github.io/emoji/docs/#links
- Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108. Retrieved from https://huggingface.co/distilbert-base-multilingual-cased
- Mimino666. (n.d.). langdetect. GitHub. Retrieved from https://github.com/Mimino666/langdetect
- ssut. (n.d.). py-googletrans. GitHub. Retrieved from https://github.com/ssut/py-googletrans
- Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks (pp. 45-50). ELRA. Retrieved from http://is.muni.cz/publication/884893/en
- Wang, L. (2019). Light on Math ML: Intuitive Guide to Understanding GloVe Embeddings. Towards Data Science. Retrieved from https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010
- Codebasics. (2022). Word_vectors_spacy_text_classification.ipynb. Retrieved from https://github.com/codebasics/nlp-tutorials/blob/main/14_word_vectors_spacy_text_classification/word_vectors_spacy_text_classification.ipynb
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in {P}ython. Journal of Machine Learning Research, 12, 2825--2830.
- Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). {API} design for machine learning software: experiences from the scikit-learn project. In ECML PKDD Workshop: Languages for Data Mining and Machine Learning (pp. 108--122).
- Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Retrieved from https://nlp.stanford.edu/pubs/glove.pdf