PREDICTING AIRBNB UNLISTING

Project developed by:

Daniel Branco, 20220599

Fernando Cruz, 20220646

Inês Ventura, 20220612

Maria Mendonça, 20220625

Contents

1. Data Exploration

The project's data comprises two sets: one for training and one for testing purposes. Each set includes information on all reviews, apartment descriptions, and host details associated with the Airbnb listings.

During the exploration of the data, one noteworthy finding is that not all Airbnb listings have reviews available. Secondly, the dataset exhibits an inherent class imbalance, with 9,033 listed instances compared to only 3,463 unlisted instances.

A detailed analysis reveals that among the unlisted Airbnb listings, a substantial majority (84.5%) does not have any associated reviews. In contrast, among the listed listings, an overwhelming majority (87.8%) has reviews available. This emphasizes the importance of considering the reviews in the predictive modeling process.

Additionally, language detection was performed on each entry in the dataframes, allowing for further insights into the multilingual nature of the dataset. By doing so, we ventured beyond the scope of the lectures as this was not covered in our coursework.

On the language detection part, we came to the conclusion that most of the texts were in english, in spite of this, it was decided that we wanted to take into account multiple languages. In order to do that the texts written in languages with residual representation were removed.

2. Data Preprocessing

To enhance the data quality, we initially removed HTML tags as a preprocessing step. This is relevant as it eliminates irrelevant markup and allows for the extraction of meaningful textual content, improving subsequent text mining analysis and modeling. Then, non-alphabetic words were also removed.

To further refine the data, tokenization, punctuation and stop word removal, lowercasing and lemmatization was implemented. Tokenization enables establishing a granular representation of the text, facilitating subsequent analysis and feature extraction. Tokenization was performed for the top most common languages within the dataframe. Punctuation was removed as it serves grammatical purposes but does not typically carry significant semantic meaning in most text analysis tasks. Lowercasing was also applied to have all words in lowercase since most of these tasks are case sensitive. Stop words removal was a crucial step as by removing these frequently occurring but less informative words, noise is reduced and the focus is on the more meaningful content of the text. With lemmatization, we consolidate the vocabulary and group together variations of the same word.

Emojis visual representation can pose challenges when processing text data. To address this, we developed a mechanism to replace emojis with their semantic equivalents or meanings.

By developing a custom function for sentiment analysis, it is possible to analyze the text and determine whether it conveys a positive, negative, or neutral sentiment, which is used later on the project.

In our project, we encountered complex and heterogeneous data, which led us to implement additional stages that were not covered in our class but proved to be essential. Specifically, we incorporated sentiment analysis and emojis treatment as crucial steps to effectively analyze and interpret the text data. By doing so, we enhanced our ability to capture the nuanced aspects of the text data, ultimately leading to more comprehensive and reliable results in our analysis and interpretation efforts.

As this stage is extremely time consuming, the group was forced to use only 7 comments per apartment_id, to accelerate the process. However, all house descriptions and host details were maintained.

3. Feature Engineering

The subsequent procedure involves systematically evaluating all selected word embeddings selected in conjunction with the chosen models explained in the section below, employing both unprocessed and processed data. It is noteworthy that the utilization of processed data entails a potential loss of valuable information originated by removing most comments to accelerate the preprocessing stage.

3.1. TF-IDF

TF-IDF is a numerical representation technique that assigns weights to words based on their frequency and importance in a collection of documents. It captures the relevance of a word in a document relative to its occurrence across all documents.

The TfidfVectorizer class is used to implement the TF-IDF algorithm. It takes the text data as input and performs both tokenization and vectorization. For this reason, we had to undo the tokenization performed during preprocessing. Also, for each apartment, all reviews, description, host_about and sentiment values were converted into a single column.

By setting max_features=10000, it was specified that the vectorizer should consider a maximum of 10,000 unique words as features during the vectorization process. This parameter limits the vocabulary size, and only the most frequently occurring words or important terms will be included as features in the resulting TF-IDF matrix. This was done so that the input layer for MLP models was able to receive data from both train and test sets, which would otherwise contain a different number of features.

3.2. GloVe embeddings

GloVe (Global Vectors for word representation) is used to generate word embeddings by utilizing global word co-occurrence statistics from a given corpus. It facilitates the representation of semantic relationships between words.

In order to implement this model, we downloaded Pre-trained Word Vectors and loaded this into a dictionary, which has returned 400000 word vectors.

We converted the input data to strings because in this case NLP algorithms need to have the data represented as a string and facilitate feature engineering steps. To prepare the data for training of NLP models, we tokenized the text and converted it to a sequence of numbers. To ensure that all the sequences have the same length of 100, we used the pad_sequences function.

Finally, we generated a word embedding matrix for each word in the word index and if a word does not have an embedding available in GloVe it will be represented by a zero matrix.

3.3. Transformed-based embeddings

We selected DistilBERT as our transformer-based embedding model for its ability to produce contextual word embeddings. These embeddings capture the surrounding words in a sentence, enabling our

classification models to better understand the subtle nuances and meaning of the text they process. This, in turn, can enhance the performance of our classification models.

DistilBERT is a compact and faster variant of BERT (Bidirectional Encoder Representations from Transformers), making it a suitable choice considering our group's computational resources.

To implement the DistilBERT embeddings, we tokenized the text data using the tokenizer provided by the DistilBERT library. This involved encoding the text and incorporating special tokens, padding, and truncation to ensure consistent input lengths.

We then created a TensorDataset comprising the input IDs, attention masks, and labels. This dataset was subsequently split into training and validation sets, with a 70% and 30% split, respectively. Given the substantial size of our dataset, we opted to use DataLoaders for training and testing. By utilizing DataLoaders, we can efficiently iterate over the data in batches during both training and testing phases. This approach allows us to process the large dataset in manageable chunks that fit into memory.

Finally, we leveraged the DistilBERT model to generate the embeddings for our text data. These embeddings serve as rich, context-aware representations of the text, which can be further utilized by downstream classification models for various natural language processing tasks.

Regrettably, due to insufficient available RAM, the group encountered difficulties in executing the code for generating Transformer embeddings with LSTM and Transformer models. However, for reference, the code will be included in our final delivery, allowing further examination and verification of the proposed approach.

4. Classification Models

4.1 Logistic Regression

Logistic regression is a statistical model used for binary classification tasks. It is a popular and widely used algorithm especially for its simplicity. It was first used as a base model for each word embedding technique as it is extremely easy to implement and interpret.

4.2 MLP

MLP is a type of ANN that performs a series of linear followed by non linear transformations in the input features, being one of the most powerful models used nowadays.

The last Dense layer, with a single neuron, and the sigmoid activation function are used, since the problem in hand is binary classification. Unfortunately, the initial results obtained showed significant overfitting. For this reason, Dropout layers were added, as well as Dense Layers with fewer neurons than the initial ones (with 128 neurons), in an attempt to learn general patterns rather than memorizing the training data.

4.3 LSTM

LSTM is a recurrent neural network architecture that is designed to effectively model and process sequential data. It was introduced to overcome the limitations of traditional RNNs in capturing long-term dependencies. That means that it can selectively remember or forget information, allowing it to capture long-range dependencies and effectively process sequences of varying lengths.

In particular, it is worth mentioning that masking was used to deal with variable-length sequences. It allows the model to ignore specific values in the input sequence that are considered as padding or irrelevant.

5. Evaluation and Results

For the metrics and evaluation of the results, setting the threshold below 0.5 is justified due to the significant class imbalance in the dataset, where the minority class is underrepresented (being unlisted), and a lower threshold helps to prioritize the correct identification of positive instances, thereby mitigating the impact of the class imbalance.

Considering all models tested with both TF-IDF and GloVe Embeddings, it is safe to say that GloVe Embeddings helped us get better results. This result was not surprising, as GloVe embeddings are pre-trained on large corpora, allowing them to capture contextual information about how words are used in various contexts. This contextualization helps in capturing the meaning and nuances of words beyond their isolated occurrences. TF-IDF, on the other hand, only considers the local frequency of words in individual documents without considering the broader context.

Although some exceptions were noticed, the group also obtained better results with the preprocessed text data over the unprocessed.

Considering the models used, as expected, the logistic model was outperformed by all others, as its ability to predict the less represented class (unlisted houses) was poor.

The performance of the LSTM and MLP models exhibited a close resemblance, with LSTM demonstrating a slight advantage over MLP.

The most promising outcome was achieved by utilizing preprocessed input features encompassing reviews, description, host_about, and sentiment columns. These features were transformed into word embeddings using GloVe embeddings, which were subsequently utilized as input for the LSTM model.

The analysis conducted by the group was limited by the utilization of only a subset of the dataset, comprising a mere 7 comments per house. Furthermore, the constrained availability of RAM consistently posed challenges, impeding the ability to process the complete dataset effectively and some techniques, including Transformers Embeddings.

However, overall, the group is satisfied with the predictions obtained, as the majority of Airbnbs in the test dataset, originated by splitting the train dataset provided by the professors, is correctly being predicted with the corresponding label.

The final predictions can be found in a csv file called ‘predictions.csv’.

6. Annexes

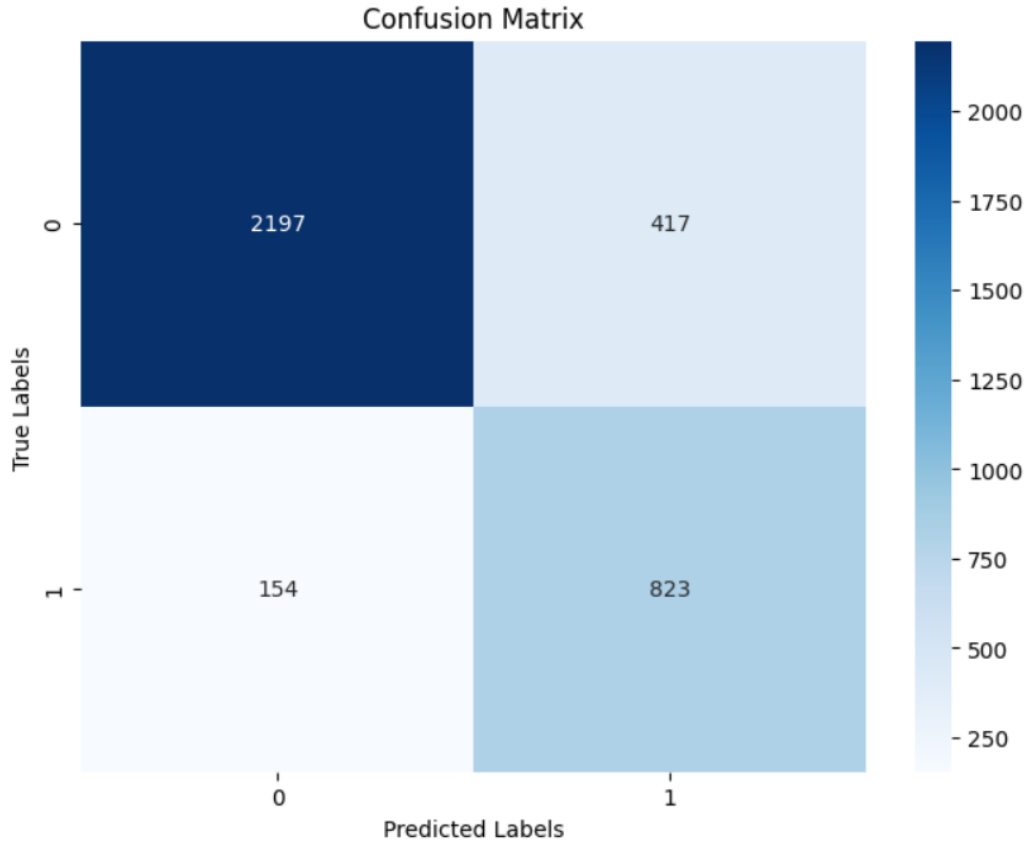

Image 1: Confusion matrix for test dataset of the best model obtained (LSTM with GloVe Embeddings)

	precision	recall	f1-score	support
0.0	0.92	0.89	0.90	2614
1.0	0.72	0.78	0.75	977
accuracy			0.86	3591
macro avg	0.82	0.84	0.83	3591
weighted avg	0.86	0.86	0.86	3591

Image 2: Statistics for test dataset of the best model obtained (LSTM with GloVe Embeddings)