Airbnb Price Prediction using Machine Learning and Sentiment Analysis

Abstract

Context

The report focuses on building a machine learning algorithm that can analyze the sentiment of Airbnb reviews to predict the rental value of an Airbnb listing. The report aims to provide a tool for hosts to make informed pricing decisions and for guests to find the best value for their money.

Objective

The objective of the report is to analyze the sentiment of Airbnb reviews to determine their impact on rental value and provide a machine learning algorithm for predicting rental value based on this sentiment.

Method

The study utilized the public Airbnb dataset of New York City, which includes information on listings, reviews, and pricing. The sentiment of reviews needed to be manually labeled, which was done on 1000 random rows of listings located within New York City. Each participant in the study was tasked with rating the sentiment of the reviews based on a rating scale of 0 for neutral, 1 for negative, and 2 for positive. Two metrics, Percent Agreement and Fleiss Kappa, were used to assess the quality of the data labeling. Once the machine learning model was trained on the manually labeled sentiment data, it was then used to predict the sentiment on 10,000 rows of unlabelled reviews.

Results

The analysis found that sentiment has a significant impact on rental value. Many of the reviews in the unlabelled data were positive, which suggests a generally positive sentiment towards Airbnb in New York City. The algorithm was able to predict the sentiment of reviews on unlabelled data with a high degree of accuracy, with a distribution skewed towards positive reviews.

Conclusion

The study highlights the importance of analyzing the sentiment of reviews to determine the rental value of an Airbnb listing. The machine learning algorithm developed in the study can help hosts to set prices that are competitive in the rental market and can help guests to find the best value for their money. The study also suggests that Airbnb remains a popular short-term rental platform in New York City.

1. INTRODUCTION

1.1. Motivate the Problem

Despite the COVID-19 pandemic, Airbnb remains a popular short-term rental platform. Airbnb hosts face tremendous competition to attract visitors and maximize their income, while guests are constantly looking to get the best deal possible. Our goal with this project is to produce an accurate price estimate that can be used by both hosts and guests when deciding on pricing or renting strategies. We will do so by looking at a variety of elements, including the sentiment of reviews, which we feel can have a big impact on a consumer's decision-making process. By analyzing these reviews, we believe that we can relate the positivity of the reviews to the Airbnb listing's value, which allows us to make an accurate price prediction.

1.2. Provide Background

The background of the problem is for hosts that are struggling to competitively price their Airbnb properties. As Airbnb's are incredibly profitable rental properties, the number of hosts who offer their properties for rent are ever increasing, leading to a very competitive rental market. Therefore, hosts are in need of a tool that can guide them to make informed decisions in pricing their property rent.

We believe that reviews that are left by previous guests have a significant impact on the property' rental value. These reviews contain valuable information about the quality of the property, the host's communication and responsiveness, and their overall experience with their rental experience. Analyzing the sentiment of these reviews can provide insights into how guests perceive a property, which has a direct correlation to the rental value of an Airbnb listing.

To tackle this problem, we will train machine learning algorithms to analyze this high-volume data and provide accurate predictions of Airbnb prices based on the compiled sentiment of the reviews and other external factors. This can help hosts to set prices that are competitive in the rental market, and it can help guests to find the best value for their money when booking through Airbnb.

2. RESULTS

2.1. How was the new data labeled/collected?

The primary data source for this research was the public Airbnb dataset of New York City, which was obtained from InsideAirBnB [1]. This dataset contains a variety of information on Airbnb listings in New York City, including details about the listings, reviews of the listings, and pricing information. However, this dataset did not include labeled sentiment data, meaning that the sentiment of the reviews needed to be manually labeled to incorporate this information into the study.

To achieve this, 1000 random rows of listings located within New York City were selected for sentiment analysis. Each participant in the study was then tasked with rating the sentiment of the reviews based on a rating scale of 0 for neutral, 1 for negative, and 2 for positive. This method of manual labeling can be time-consuming and prone to inter-annotator disagreement. To address this, a tiebreaker was used to determine the final sentiment analysis for each review in cases where there was disagreement among the annotators.

To ensure the quality of the data labeling, two metrics were used to determine the accuracy of the sentiment analysis. The first metric was the Percent Agreement metric, which calculates the percentage of instances in which the annotators agreed on the sentiment rating of a review. In this study, the Percent Agreement metric yielded a score of 98%, indicating a high level of agreement among the annotators.

The second metric used to assess the quality of the data labeling was Fleiss Kappa, which is an alternative method for calculating inter-annotator agreement that takes into account the possibility of chance agreement. Fleiss Kappa is generally considered a more robust metric for assessing inter-annotator agreement than Percent Agreement, as it accounts for the possibility of chance agreement between annotators. In this study, Fleiss Kappa yielded a score of 0.80, which is generally considered to indicate strong agreement among the annotators.

listing_id	comments	Hannah	Nikhil	Douglas	labelled_sentiment
44799007	This place is top notch . Very clean place to stay and excellent location.	2	2	2	2
44799007	The place isn't bad , they just have some plumbing issues that need to be worked on	0	1	1	1

Figure 1 Sentiment Analysis Tie Breaker Example

Overall, these methods for labeling the sentiment of the Airbnb reviews in the dataset were designed to ensure accuracy and reliability, and the high scores obtained on both metrics suggest that the labeling was performed with a high degree of consistency and agreement.

2.2. How does the newly added data compare with the original data?

Once the machine learning model was trained on the manually labeled sentiment data it was then used to predict the sentiment on 10,000 rows of unlabelled reviews. Compared to the original labeled data, the distribution was 9258 positively labeled reviews, 586 labeled neutral reviews and 156 negative reviews. The 10,000 rows of reviews that were originally from InsideAirbnb included 96 features such as city, state, zip code, market. However, the original data only had 17 features listed. This difference could be because of different time frames for the collection of data.

One thing to note is the distribution of the predicted sentiment on the unlabelled data is skewed towards positive reviews, with a small proportion of negative reviews and an even smaller proportion of neutral reviews. This suggests that many of the reviews in the unlabelled data are positive, which could reflect the overall positive sentiment towards Airbnb in New York City.

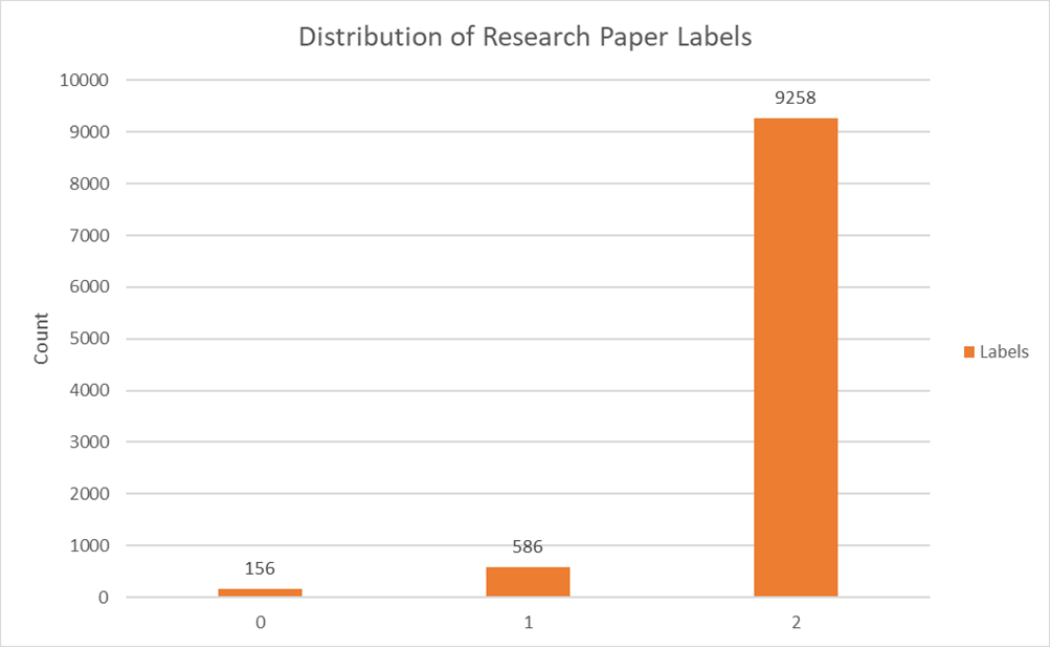

Figure 2 Distribution of Research Paper Labels

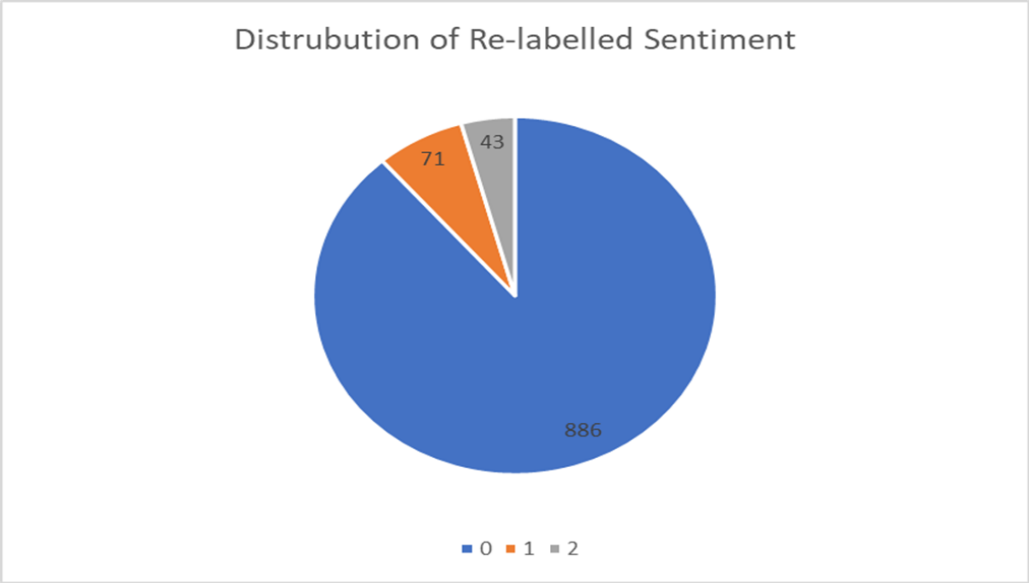

Figure 3 Distribution of Re-Labelled Sentiment

2.3. How was the data preprocessed?

NLP Preprocessing:

To prepare the 1000 rows of data prior to manually labeling them, we filtered the reviews to only keep reviews after the year of 2021 to avoid overlapping with the original dataset. From those reviews we take 50 of the listing IDs that have over 20 reviews. Lastly, we take 20 English reviews each from those listing IDs. The reason why we do this is to ensure each listing has enough sentiment data to properly support whether the listing is reviewed positively or negatively.

After getting the 1000 rows of English reviews compiled from 50 different Airbnb listings, we can start to preprocess the text prior to training our sentiment analysis machine learning model. First, we tokenize each review, splitting the sentences into words. Then we remove all stop words. Then we remove punctuation which includes emoticons, then we spell check each word to ensure that they are properly spelled. Lastly, we utilized a snowball stemmer to stem each word. The finished result will be a listing of strings.

With the list of strings, we utilized a bag of word approach using hashingTF, as well as TF-IDF to vectorize each word to generate our machine learning model.

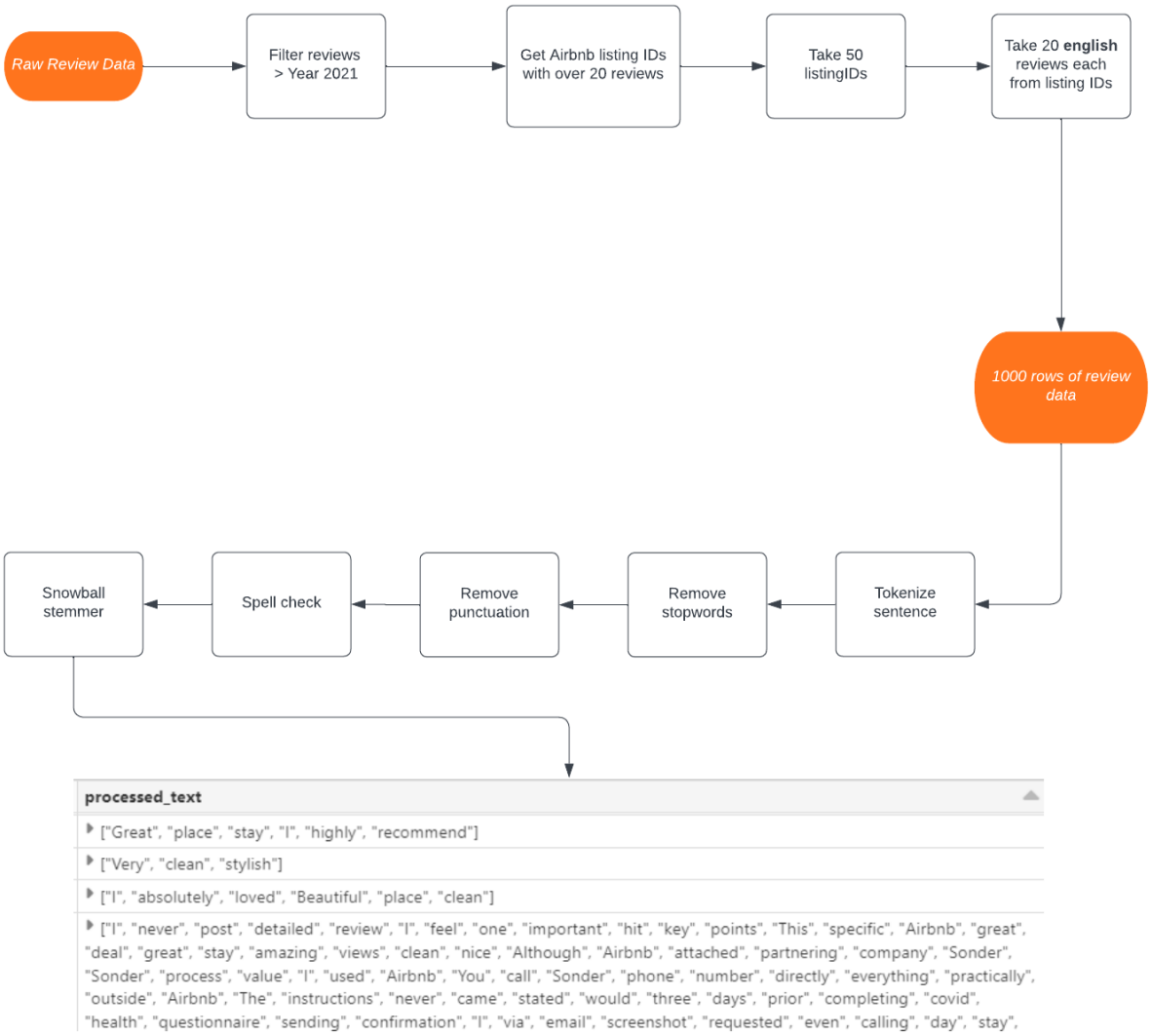

Figure 4 NLP Preprocessing Steps

Price Prediction ML Model Preprocessing:

To utilize the built-in libraries of PySpark's machine learning models, we need to assemble all the features we wish to include in the training of the machine learning model into a vector column. We have three categories of features that need to be compiled numerically into a vector column. They are the list of words that need to be compiled into a sentiment value, numerical features such as number of reviews, and categorical features such as type of room. This second stage uses very different preprocessing techniques that is very different than NLP preprocessing.

With the processed text, we need to find the mean sentiment of all the reviews compiled for a certain listing. This will be achieved using HashingTF to create a numerical representation of the bag of words. Since there are many reviews per Airbnb listing, we compile all the sentiment and categorize a listing as 0 or negatively reviewed, 1 for neutrally reviewed, and 2 for positively reviewed overall. We opted for this approach rather than just simply taking the meaning of all sentiment because we want the sentiment to have a big impact on predicting a Airbnb's predicted value. Simply scaling the mean sentiment can be a viable strategy but we ran out of time to explore such an option.

Existing numerical features can be directly assembled via the vector assembler.

For categorical features such as type of room. These features are categorized by strings such as hotel and apartment. To convert these features into numeric values, we utilized the One-Hot Encoder approach. We first use string indexer to numericize the strings, then we use the One-Hot Encoder to convert them into vectors.

Finally, all features are compiled via a vector assembler, and we used a standard scale to scale all data to remove potential bias and uneven weighting. We then use this feature column to train various machine learning models.

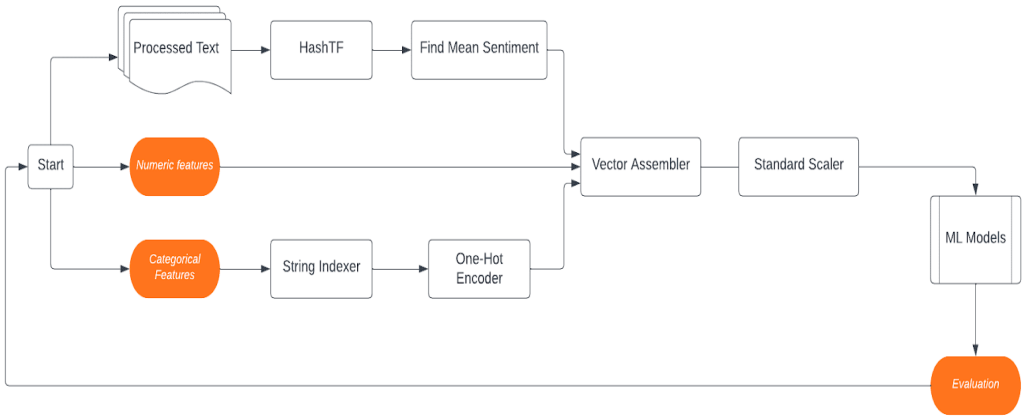

Figure 5 Price Prediction ML Model Preprocessing Pipeline

2.4. How do the models perform on the original data vs the new + original data?

In this step, we will be referring to the sentiment model derived from sentiment analysis on 1,000 rows of labeled reviews, which is the old data. The performance of this sentiment model is crucial as it was used to label the new data, which consisted of approximately 10,000 rows of unlabeled reviews. It is important to note that the new data could not be tested previously as it was unlabeled. Therefore, in this discussion, we will only focus on testing related to the old data. The table below presents the test results of the sentiment model derived from the old data.

	Random Forest Classifier	Weighted Random Forest Classifier	Naive Bayes	HashingTF with Multi-class Logistic Regression	TF-IDF with Multi-class Logistic Regression
Precision for negative (0)	0.00	0.18	0.00	0.29	1.00
Precision for neutral (1)	0.00	0.08	0.00	0.11	0.85
Precision for positive (2)	0.91	0.90	0.89	0.93	0.98
Recall for negative	0.00	0.13	0.00	0.13	0.73
Recall for neutral	0.00	0.19	0.00	0.14	0.81
Recall for positive	1.00	0.82	1.00	0.93	1.00
F1	0.87	0.78	0.83	0.84	0.97
True Positive (TP)	0.00	4.00	0.00	3.00	17.00
True Negative (TN)	292.00	253.00	297.00	273.00	294.00
False Positive (FP)	0.00	44.00	0.00	24.00	3.00

False Negative (FN)	16.00	17.00	21.00	18.00	4.00

Figure 6 Performance Results of the Sentiment model derived from the old data

A quick note: these metrics are not suitable for evaluating our price prediction model, as it is a regression model and not a binary classification model. In reference to the confusion matrix, the “positive” label is 1(neutral) and the negative label is 0(negative) and 2(positive). True Positives (TP) indicate the number of correct predictions for label 1. True Negatives (TN) represent the cases where the prediction was 0 when the actual value was either 0 or 2, and the cases where the prediction was 2 when the actual value was either 0 or 2. False Positives (FP) denote the cases where the prediction was 1 but the actual value was either "0" or "2". False Negatives (FN) refer to the cases where the prediction was either 0 or 2 when the actual value was 1.

Based on the table above, relying solely on f1 scores may not provide sufficient insight when comparing the performance of the various tested models. However, it is apparent that the Random Forest Classifier and Naive Bayes models consistently predicted label "2", as evident from their high precision/recall scores for "2". Conversely, the scores for labels "0" and "1" were close to 0, indicating poor performance. While the weighted Random Forest Classifier showed slight improvement in precision/recall scores for "0" and "1", its results were similar to those of the Random Forest Classifier and Naive Bayes models. On the other hand, the Multiclass Logistic Regression model exhibited slightly better performance compared to the weighted Random Forest Classifier. Nevertheless, the latest sentiment model, TF-IDF with Multi-class Logistic Regression, demonstrated exceptional performance with high precision and recall scores across all three labels (0, 1, 2). Furthermore, this model also achieved the highest number of True Positives (TP) and True Negatives (TN), along with the lowest number of False Positives (FP) and False Negatives (FN).

2.5. How does the performance of the models change based on the choice of hyperparameters?

To develop an effective price prediction model, our team conducted thorough testing of three distinct machine learning algorithms: Linear Regression (Linear), Decision Trees (Non-Linear), and Gradient Boosting (Non-Linear). The performance of these models was assessed by comparing their train/test R2 (coefficient of determination) and RMSE (root mean squared error), which are crucial metrics for evaluating their accuracy and predictive capabilities. The results of these evaluations are presented in the table below, providing valuable insights into the performance of each model.

	Linear Regression (With scaled data)	Decision Trees	Gradient Boosting

Train RMSE	83	65	56
Test RMSE	114	122	123
Train R2	0.59	0.75	0.82
Test R2	0.37	0.28	0.27

Figure 7 Price Prediction performance results with different ML models

Although the findings may not be entirely conclusive, our analysis suggests that Linear Regression outperforms both Decision Trees and Gradient Boosting models. While Decision Trees and Gradient Boosting exhibit higher train R2 scores (0.75 and 0.82, respectively), their test R2 scores are significantly lower, indicating potential overfitting. In contrast, Linear Regression shows more stable results across both train and test R2 scores. Therefore, based on our analysis, we conclude that Linear Regression is the most favorable model for our dataset.

To improve our model's performance, we experimented with the hyperparameters of the Linear Regression machine learning algorithm, specifically the elasticNetParam, regParam, and maxIter. The elasticNetParam parameter determines the balance between L1 (Lasso) and L2 (Ridge) regularization in the Elastic Net regularization, taking values between 0 and 1. A value of 0 corresponds to Ridge regression, 1 corresponds to Lasso regression, and 0.5 represents an equal balance between the two. The regParam parameter controls the strength of L2 regularization, with larger values indicating stronger regularization. Lastly, the maxIter parameter controls the maximum number of iterations for the optimization algorithm used to find the best-fit coefficients for the model. Increasing maxIter may lead to a more accurate model, but it may also increase computation time. The table provided below presents the multiple values that were chosen for the hyperparameters, while the subsequent table displays the results obtained from the model utilizing the best hyperparameter settings.

```
paramGrid = ParamGridBuilder() \
    .addGrid(lr.regParam, [ 3.0, 5.0, 6.0]) \
    .addGrid(lr.elasticNetParam, [0.3, 0.7, 1.0]) \
    .addGrid(lr.maxIter, [10, 50, 100]) \
    .build()
```

Figure 8 Manipulating Hyperparameter of Linear Regression ML model

	Model With Best Hyperparameters	Model With Default Hyperparameters
Train RMSE	88	83
Test RMSE	112	114
Train R2	0.55	0.59
Test R2	0.40	0.37

Figure 9 Price Prediction performance results for model with modified hyperparameters and with default parameters

From figure 9 it is apparent that despite utilizing improved hyperparameter settings, there was no significant improvement in the price prediction model. This could potentially be attributed to the limited amount of data available in the model (only around 544 rows), which may not have been sufficient to yield noticeable changes with the variation of hyperparameters.

2.6. How are the misclassifications of the best performing model distributed?

We had decided to prioritize the in-depth analysis of misclassifications made by the sentiment model, rather than the price prediction model. We believed that the majority, if not all, of the incorrect predictions could be attributed to the sentiment model used to derive sentiment values for the 10,000 rows. As a side note, we had identified a total of 51 misclassifications when applying the sentiment model to the 1,000 rows. Figure 10 displays some examples of the misclassifications:

comments	processed_text	labelled_sentiment	prediction
Very dirty!!	["Very","dirty"]	0	2
Dirty; bathroom tile floors/tub had mold in the grout in several places. Ants on the night stand upon approval. Bedding felt like a Red Roof Inn. Elevators slow. Would never stay here again. Will tell others not to as well. 1st time and now last time I will ever sway from the Marriott Marquis.	["Dirty","bathroom","tile","mold","grout","several","places","Ants","night","stand","upon","approval","Bedding","felt","like","Red","Roof","Inn","Elevators","slow","Would","never","stay","Will","tell","others","well","time","last","time","I","ever","sway","Marriott","Marquis"]	0	2
Better than expected. At this price I didnt expect much, but I was pleasantly surprised. Dont need more than this :)	["Better","expected","At","price","I","didnt","expect","much","I","pleasantly","surprised","Dont","need"]	2	0

Figure 10 Misclassified Sentiment

In the first example, it was clear to us that "Very dirty" conveyed negative sentiment. However, since most of our reviews were positive in our data, our model likely associated the word “very” with positive sentiment, as it can be used in a positive or negative context. From this example, we learned that the model tended to be confused by short reviews. We thought a possible fix for this would be to set a minimum word limit for reviews we use.

In the second example, the model did not accurately capture the negative sentiment expressed in the words "dirty bathroom," "mold," "grout," "ants," "elevators slow," and "never stay." The model could have focused more on the positive word "approval". We suspected that the model failed to capture the contextual clues of this piece of text because it was a long review. However, almost all words in this review were negative, which was interesting since the model still got it wrong.

In the third example, the model incorrectly predicted the sentiment by possibly focusing too much on the negative words "didn't" and "expect" and ignoring the positive words "better than expected" and "pleasantly surprised." The model failed to capture that "didn't" and "expect" cancelled each other out to make the phrase convey a positive sentiment. In this case, the bi-gram approach might have been better suited to handle this specific phrase. The model also ignored the positive emoticon ":)" at the end of the review. Our potential next step was to include all emoticons in our sentiment analysis as they can provide insight into whether an emoticon conveys a positive or negative sentiment.

3. DISCUSSIONS

Nikhil Naikar:
Our model has the potential to assist in providing feedback on the pricing of Airbnb listings by categorizing them as great value, fair value, or poor value based on sentiment analysis of feedback received for that listing, along with other relevant listing information. For instance, if a listing has a price that is higher than the average and a customer is deliberating whether to choose it by reading reviews, our model can streamline the process by conducting sentiment analysis on the feedback and factoring in the listing information to determine a "should be" price in comparison to similar listings. If the determined price aligns closely with the listing price, it would be categorized as fair value. If it is considerably higher, it would be classified as great value, and if it is significantly lower, it would be considered poor value. This feedback could prove beneficial for Airbnb hosts in accurately pricing their listings, and it could also aid customers in making informed decisions when selecting a particular listing.

Hannah D'Souza:
The sentiment analysis feature of the model provides valuable insights into how guests perceive a property. By analyzing the sentiment of guest reviews, property owners and managers can identify areas of improvement and capitalize on their strengths to enhance the guest experience. For instance, if guests consistently express satisfaction with the property's location, the property can highlight this aspect in its marketing efforts. Conversely, if guests often complain about the cleanliness of the rooms, the property can take steps to improve its housekeeping services. Furthermore, sentiment analysis allows property owners and managers to track changes in guest sentiment over time, enabling them to promptly address emerging issues and prevent them from escalating. In short, leveraging sentiment analysis can help properties make data-driven decisions, improve the overall guest experience, and ultimately attract more potential guests.

Douglas Yau:
Aside from directly using our model to predict Airbnb prices, we can extend our model to other rental properties such as apartment complexes and hotels. Additionally, this can be an interesting study to marketing teams to study the correlation between sentiment of reviews

and market value of any products. Lastly, our model can be used by the Airbnb company to predict its yearly revenue by predicting the rental value of all its listings. This is incredibly valuable if it is accurate and can be a strong selling point to attract investors.

4. CONCLUSIONS

The project's objective was to develop a machine learning-based model for predicting Airbnb prices using sentiment analysis. The team collected 1000 rows of reviews (20 reviews per listing, 50 listings) from insideairbnb.com [1] and manually labeled them with sentiment labels of 0 for negative, 1 for neutral, or 2 for positive, retaining the most commonly agreed-upon sentiment value. Next, the team preprocessed the data by tokenizing sentences, removing stop words and punctuation, spell-checking words, and using the snowball stemmer before building a sentiment analysis machine learning model with these labelled reviews. The bag-of-words approach with hashingTF was used to vectorize the remaining list of words into a single column.

Three initial machine learning algorithms, namely random forest, naive bayes, and logistic regression, were experimented with, and logistic regression performed the best while the other two did not yield satisfactory results. The team then obtained 10,000 rows of unlabelled reviews (20 reviews per listing, 500 listings) from a research paper. Manual labeling of such a large volume of data was impractical, so the sentiment model derived from the best-performing algorithm was used to predict sentiment values for the 10,000 rows. The sentiment values for the 10,000 rows and the initial 1000 rows of data were then grouped by listing IDs, and the mean of the sentiment values was calculated. Based on the mean, sentiment values were categorized as follows: if the mean was greater than or equal to 1.75, it was set as 2 (indicating positive sentiment), if the mean was between 1.5 and 1.75 (inclusive), it was set as 1 (indicating neutral sentiment), and if the mean was less than 1.5, it was set as 0 (indicating negative sentiment).

Next, the team obtained listing information from two different listing files, one for the 1000 rows and the other for the 10,000 rows. Features such as neighborhood group, room type, number of reviews, availability, and derived sentiment were selected for further analysis. The combined dataset of 10,000 and 1000 rows was then used to build the final price prediction model. Three different models, linear regression, decision trees, and gradient boosting, were experimented with, and linear regression performed the best. The team attempted to optimize the hyperparameters of linear regression, including elasticNetParam, regParam, and maxIter, but despite using improved hyperparameter settings, there was no significant improvement in the price prediction model. Upon reviewing the wrong predictions, the team concluded that the poor performance of the models was likely due to the lack of data (only 544 rows) and the need for improvement in the derived sentiment model.

Based on feedback from the professor, the team attempted two strategies to improve the sentiment model. The first strategy involved adding weights to the sentiment values to account for data imbalance and feeding this weight column into the random forest machine learning algorithm. Although there was a slight improvement in the performance of this model, it was still not satisfactory and performed worse than the initial sentiment model. The second strategy

involved using TF-IDF instead of hashingTF after the preprocessing step and feeding this into the Multiclass logistic regression machine learning algorithm. The outcome of this new model was significantly better than the initial sentiment model and showed promising results for further analysis.

5. REFERENCES

1. Murray Cox.Get the Data. Inside Airbnb. http://insideairbnb.com/get-the-data, 2016.
2. DATAtab. Fleiss Kappa [Simply Explained]. https://www.youtube.com/watch?v=ga-bamq7Qcs&t=224s, 2022.
3. Ardian Umam. Apache Spark Tutorial - 08 Sentiment Analysis of Twitter Data. https://www.youtube.com/watch?v=DBLFWpWn7UM, 2018.
4. Yasmine Hejazi. Beginner's Guide to Linear Regression with PySpark. Towards Data Science. https://towardsdatascience.com/beginners-guide-to-linear-regression-with-pyspark-bfc39b45a9e9, 2023
5. MulticlassMetrics. Apache Spark. https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.mllib.evaluation.MulticlassMetrics.html
6. Classification and Regression. Apache Spark. https://spark.apache.org/docs/latest/ml-classification-regression.html