E178 Final Project

"Enhancing Decision Making in Airbnb Grading: A Machine Learning Approach combining Sentiment Analysis and Price Prediction Models"

Group 5
Brian Sim, Jovon Lim, Samuel Ang, Yonghe Jin

1 Introduction

1.1 Practical Motivation

Travelling and exploring a new country is part of every exchange student's overseas experience. When it comes to finding temporary accommodation in a foreign place, AirBnBs present themselves as an economical option for bigger groups of travellers. As the four of us exchange students were finding accommodation for our travels, we realized that some properties were exorbitantly priced, while others were hidden, cheap gems.

We also noticed that we gravitated towards worded reviews to learn more about a property despite the available star review system, which allows past renters to rate the property out of 5 stars. This was due to our inherent assumption that star ratings were inaccurate because of the different and arbitrary criteria used by different reviewers in the rating process.

1.2 Project Goals

Given the above premise, we aim to utilize regression tools and build a machine learning model to predict appropriate rental prices based on key features available in the owner's home, such that homes can be priced correctly and appropriately. We hope to convert this analysis into a predictor that is able to tell if a property is over-, under- or accurately priced based on its features.

Additionally, we aim to perform sentiment classification on the worded reviews given by users, then aggregate negative reviews and sieve out common complaints about the listing. This allows potential AirBnB renters to skip the tedious review reading process while still being able to gather a collective sentiment of past renters' opinions of the AirBnB property.

We then hope to combine both results and grade each AirBnB listing, such that future guests can make more informed decisions on whether to rent the listing based on a single grade, reducing the hassle and time taken in the decision making process.

1.3 Data Description

We used the "Boston AirBnB Open Data" dataset from Kaggle, containing data from 2016 to 2017, for our analysis. This dataset contains three csv files, namely '*listings.csv*' which has full descriptions and average review score of rental homes, '*reviews.csv*' which has a unique ID for each reviewer and detailed comments of homes and '*calendar.csv*' which contains listing ID along with the price and availability for each day.

1.4 Project Plan

1. We will first perform numerical and categorical **Exploratory Data Analysis** on *'calendar.csv'* and *'listings.csv'* to obtain a clean dataset of the useful features of an AirBnB listing that help us determine its price.
2. During **Model Prediction**, we will run this clean dataset through numerous models to select the best performing model. After training and tuning this model, we will obtain *'predicted_prices.csv'*. This contains our model's prediction of each AirBnB's suggested price compared to its actual price.
3. Concurrently, we will perform **Sentiment Analysis** on *'reviews.csv*'. After basic cleaning, we will pass this data through a Word Embedding Algorithm to obtain predictions of Sentiment Rates *'final_sentiment.csv'* on each AirBnB listing. This classifies how well received each AirBnB listing is.
4. Finally, we will merge the *'predicted_prices.csv'* and *'final_sentiment.csv'* for **Overall Grading and Rating of AirBnBs**. We will obtain overall scores of each AirBnB listing based on its deviation from its suggested price as well as its sentiment rate. This allows us to grade each AirBnB from A through C.

2 Exploratory Data Analysis

We first carried out exploratory data analysis of the data to better understand what we were working with. Our aim here was to clean the dataset to make it more palatable for future model prediction.

2.1 Price Trends Analysis

From “calendars.csv”, we identified AirBnB home price trends in Boston over different months and days of the year.

2.1.1 Average Price by Month

The bar chart below shows that the peak demand for Boston's AirBnBs occurs in Fall for the year 16/17.

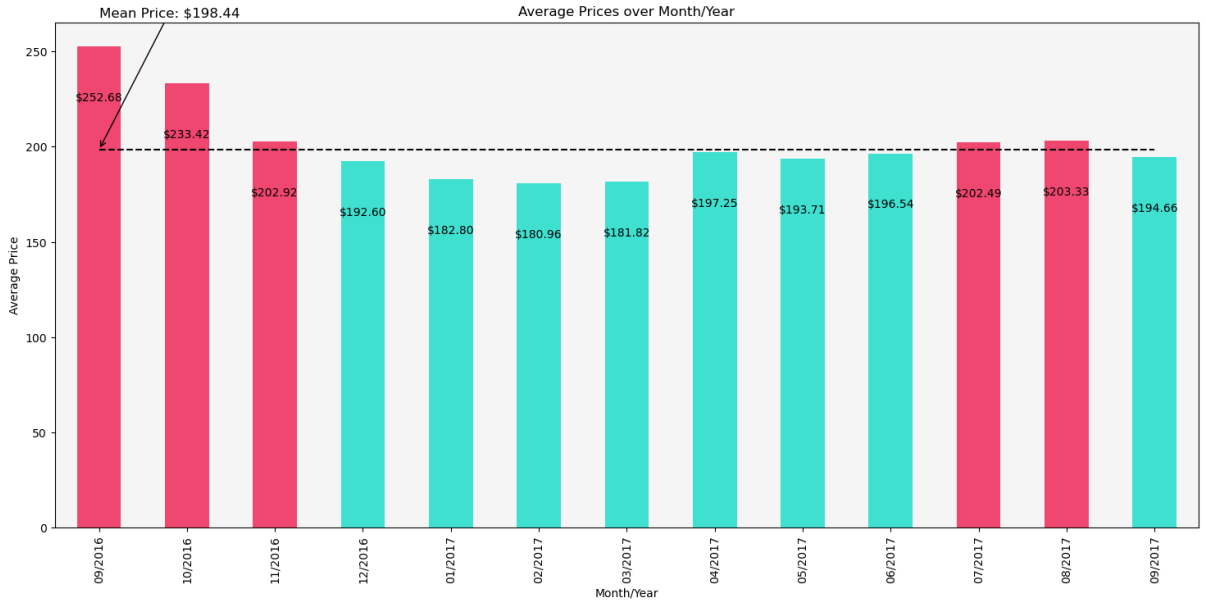

Figure 1: Mean AirBnB price by month over the year 2016 to 2017

This peak demand corresponds to the mid-Summer season (July, August) and the Fall season of September to November. There is a general low mean price trend for the cold Winter season from December to March. Prices are otherwise relatively constant until they peak out in Fall. This implies that Boston has many visitors from Summer to Fall.

2.1.2 Average Price by Day

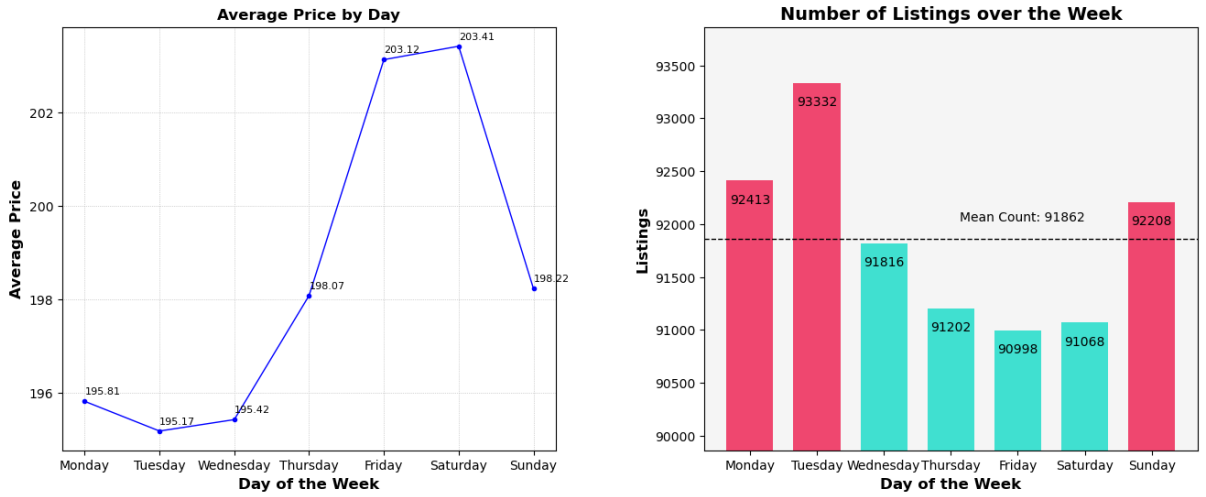

Figure 2: Mean AirBnB price by day over the year 2016 to 2017

Typical AirBnB rentals for each day start at 3pm on the day of rental and end on 11am on the next day. The line chart shows that prices for AirBnBs are higher from Thursdays to Sundays, with peaks on Fridays and Saturdays. This corresponds to the usual weekends and long weekends when working adults take work leave near weekends for an extended weekend. Trips out of town are more likely over these weekends and long weekends, which translate to a higher demand for an AirBnB to stay in.

The bar chart shows that there is a greater number of available listings for rent on Sunday, Monday and Tuesday, which correspond to the lower weekday prices of these days seen in the line chart. From a demand-and-supply standpoint, we can infer that the lower prices are due to a lower booking demand on Sundays, Mondays and Tuesdays, which by logical deduction means there will be a greater supply of available listings on these days.

One anomaly is Wednesday, with a low average price despite the relatively low number of available AirBnBs for rent. We presume that most renters have already found a rental before or on Wednesday for their weekend trip (which makes sense, due to proper planning).

2.1.3 Average Price by Holiday

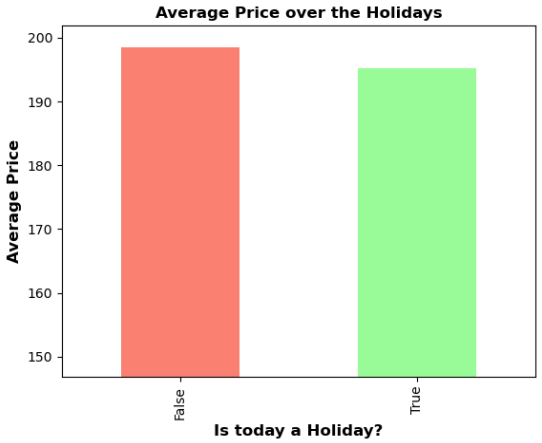

Figure 3: Mean AirBnB price by day over the year 2016 to 2017

We also realized that there is little effect of holiday surge pricing in Boston for the year 2016 to 2017.

2.2 Categorical Feature Analysis

From “listings.csv”, we investigated the effect of AirBnB features such as neighborhood location, property type, size and amenities on their price.

2.2.1 Price By Area

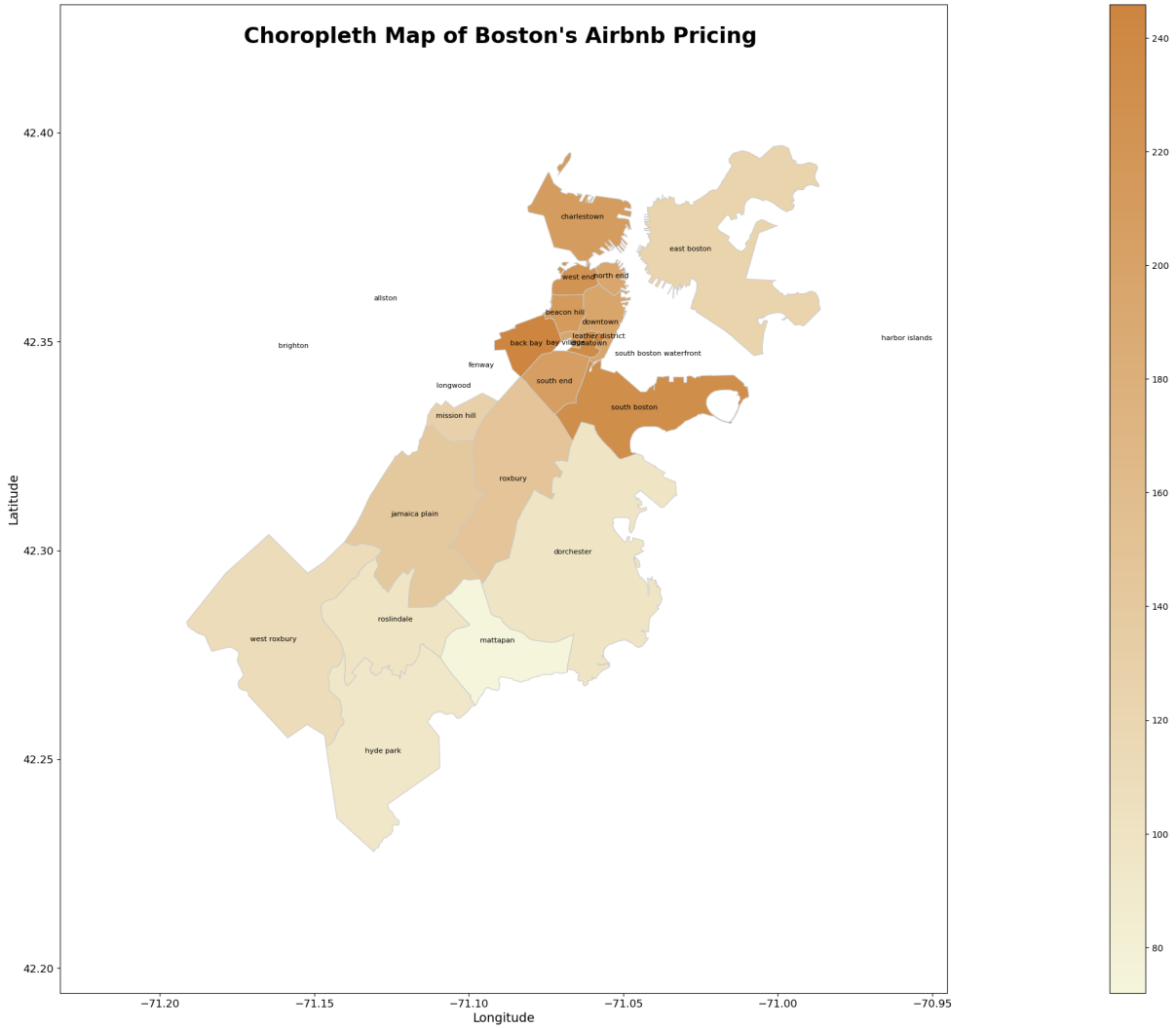

Figure 4: Chloropleth map of AirBnB prices by area

We noticed that certain districts of Boston have generally higher AirBnB prices, and the prices decrease radially away from those neighborhoods. This price distribution corresponds to the general selling price of properties in the area.

Figure 5: Swarmplot of AirBnB prices by neighborhood

On further analysis, the swarmplot seems to show that Allston-Brighton, Jamaica Plain, and Dorchester have a higher concentration of lower-priced AirBnB properties, while data is more inconclusive for the other neighborhoods. We also notice the existence of extreme anomalies in each neighborhood.

2.2.2 Accessibility and Amenities

The “transit” column in the dataset was made of words of the types of available transportation methods. This had to be converted into a numerical form. We performed an iterative count to convert the number of transport options in this column into an integer number for the number of transport types as a measure of accessibility. Similarly, we counted the number of available amenities and converted the categories into an integer representation.

2.2.3 Price by Room and Property Type

We analyzed the room type distribution in Boston. As we expected, shared rooms were the cheapest, followed by private rooms and then entire homes.

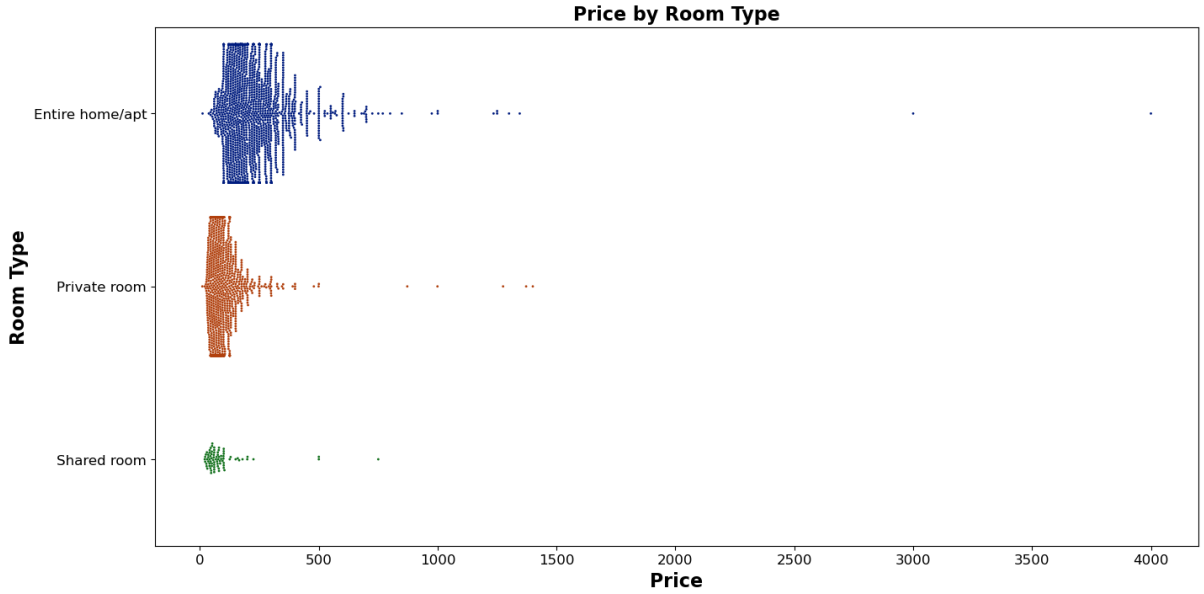

Figure 6: AirBnB prices by room type

Further analysis of the “entire home” category showed that majority of houses are priced similarly, but apartments are priced differently and have an even distribution across a large price range.

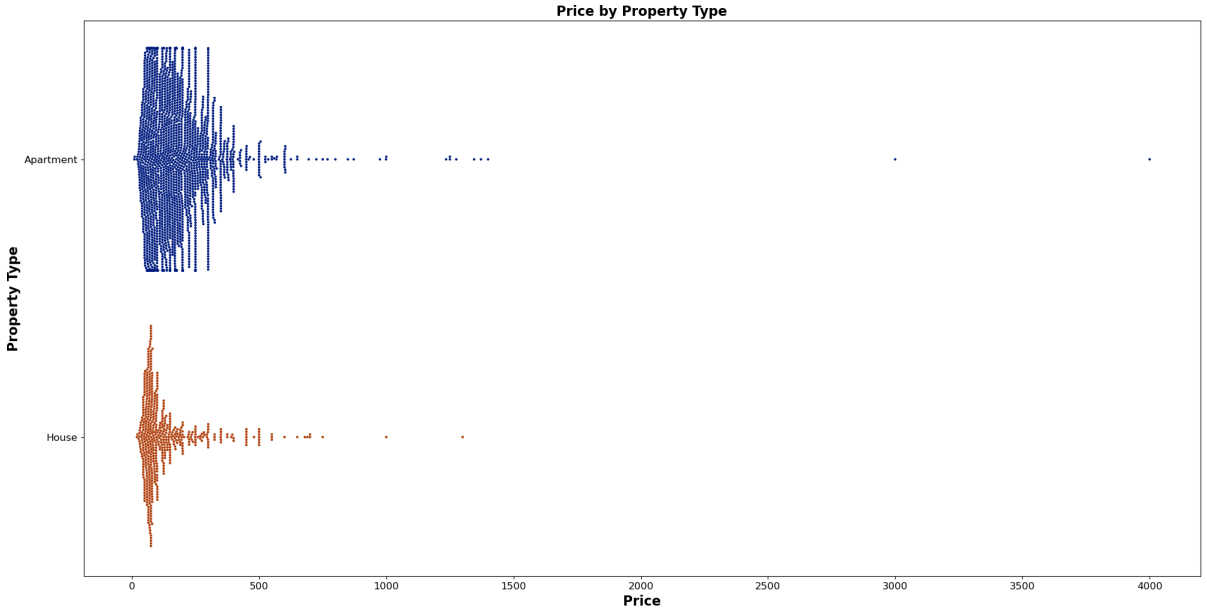

Figure 7: AirBnB prices by property type (entire homes)

2.3 Numerical Feature Analysis

We performed a correlation analysis to determine the correlation between numerical features and AirBnB prices. We used a 0.35 correlation as the threshold for important features. The important features of square_feet, number of beds, number of bedrooms and number of guests are colored green. However, we realized that square_feet had 98% NaN values, and removed that feature as well.

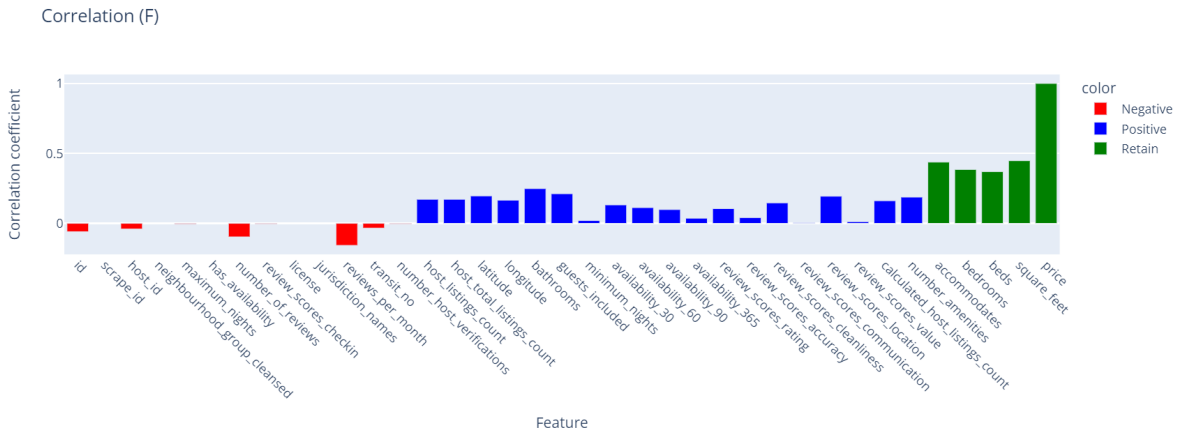

Figure 8: Correlation analysis between numerical features and AirBnB price

2.4 Label Encoding and Removing Outliers

Our final hurdle to a cleaned dataset was label encoding the categorical variables. We encoded the object variables of property_type, room_type, and neighborhood into numerical labels. We then used the 1.5 Interquartile Range (IQR) rule to identify outliers based on the decision range closest to a Gaussian distribution. After removing outliers according to the 1.5 IQR Rule, we reached our checkpoint of a cleaned dataset and saved it as 'cleaned_listing.csv'.

3 Model Prediction using Regression Methods

3.1 Train-Test Split

We performed a train-test split on our cleaned dataset, 'cleaned_listing.csv', with 70% training data and 30% test data.

3.2 Single-Variate Linear Regression

We ran each of the variables in our cleaned dataset against price to identify the most important feature in predicting AirBnB prices.

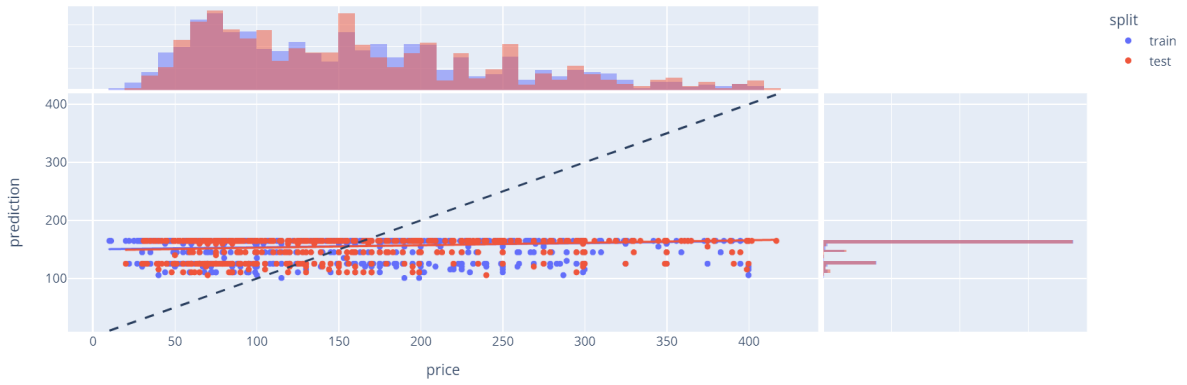

Figure 9: Property Type against Price

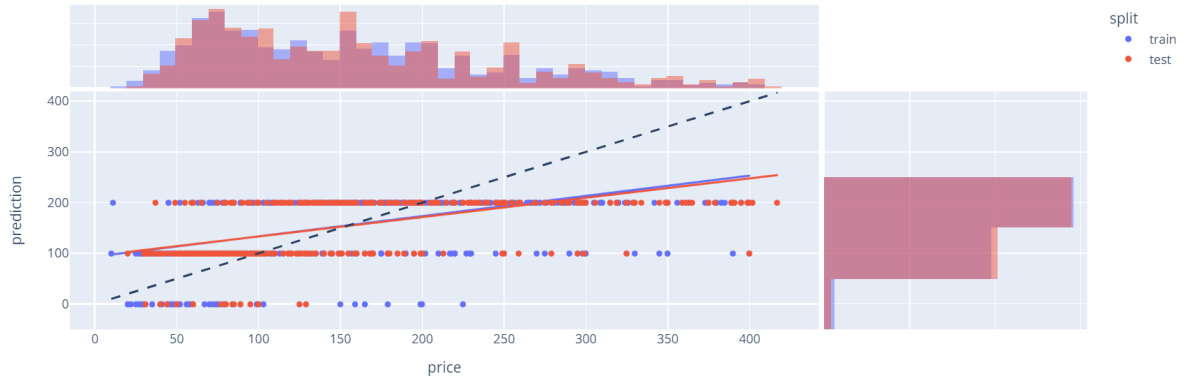

Figure 10: Room type against Price

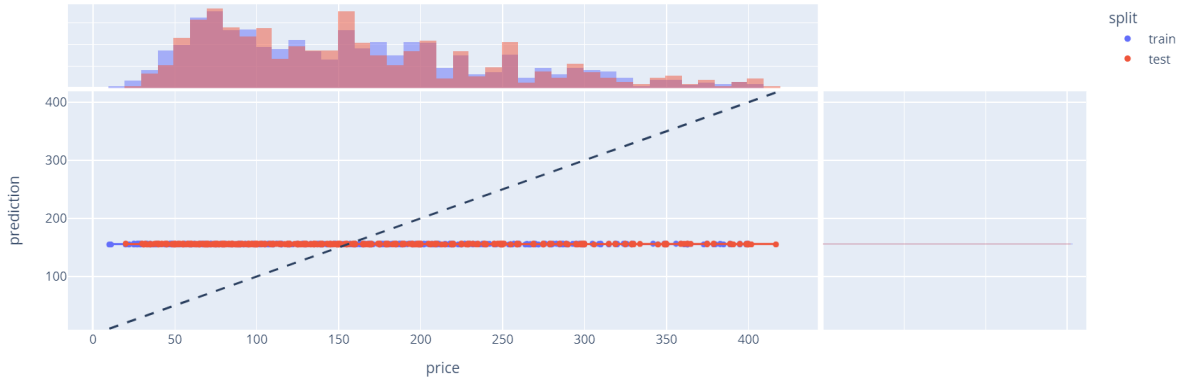

Figure 11: Neighborhood against price

Figure 12: Housing Capacity against Price

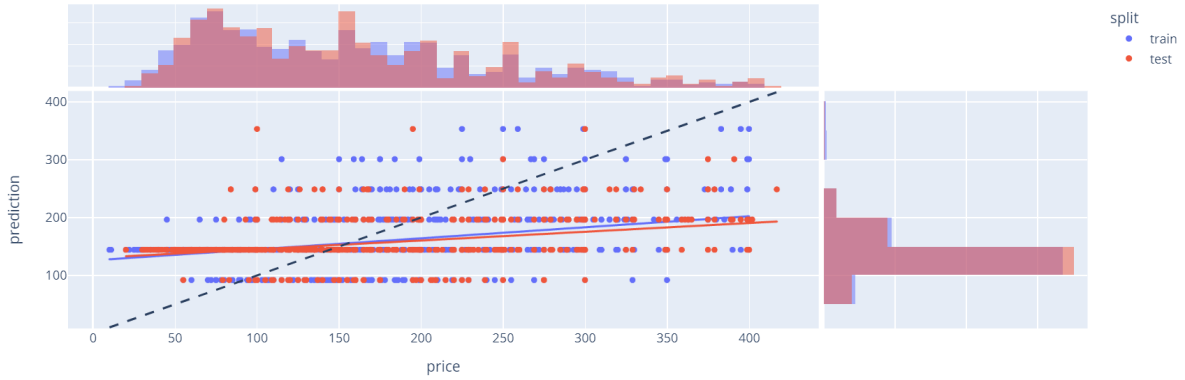

Figure 13: Number of bedrooms against Price

Figure 14: Number of beds against price

Noting that the blue and red solid lines are the regression lines for train and test data respectively, it is visually obvious that the data is badly fit.

Variable	Train (70%)		Test (30%)	
	R^2	MSE	R^2	MSE
property_type	0.035565	7164.17	0.049305	7405.75
room_type	0.399159	4463.27	0.399975	4674.09
neighborhood	0.000041	7428.06	-0.000891	7796.76
capacity (accommodates)	0.318617	5061.56	0.294400	5496.50
bedrooms	0.190775	6011.21	0.147845	6638.14
beds	0.188676	6026.81	0.147103	6643.92

Figure 15: Goodness of Fit of Single-Variate Linear regression against Price

From the train set, only 'room_type' and 'accommodates' have better (but still weak) explained variance values of above 0.3, while neighbourhood has close to 0 explained variance. All of the features have poor MSE with values above 4400.

From the test set, 'room_type' and 'accomodates' have slightly lower values of explained variance (below 0.29) compared to the train set, while neighbourhood remains at almost 0 explained variance. All of the features have slightly worse MSE values compared to the train set, with values above 4600.

The low explained variance and high MSE of both the train and test data further substantiate that single variable models are not effective in predicting AirBnB prices.

3.3 Ensemble Bagging Methods

We also tried using a single-variate Random Forest Regression model to test if ensemble bagging would improve model performance.

Variable	Test Data MSE	
	Linear Regression	Random Forest Regressor
property_type	7405.75	7466.45
room_type	4674.09	4420.97
neighborhood	7796.76	5930.75
capacity (accommodates)	5496.50	5099.26
bedrooms	6638.14	5432.49
beds	6643.92	6132.25

Figure 16: Comparison between Linear Regression and Random Forest Regressor

In general, the MSE of the test data are worse with the single-variate random forest regressor. The ensemble bagging method is unsuitable for model prediction as well.

3.4 Multivariate Linear Regression

Next, we ran linear regression on all the variables together against price in a single model. With a greater number of predictors, we expect the results to be better than the single-variate models due to increased variance during training.

Figure 17: Multivariate linear regression against price

Variable	Train (70%)		Test (30%)	
	R^2	MSE	R^2	MSE
Multivariate	0.4975	3732.58	0.4761	4081.10
room_type	0.3992	4463.27	0.4000	4674.09

Figure 18: Comparison between multivariate and best performing single variable

We see that the explained variance has improved, with 0.498 on the train set and 0.476 on the test set. MSE has also been greatly reduced, with 3732 on the train set and 4081 on the test set. By comparing the multivariate regression results to the best performing single variable (room_type), the comparative major improvement is evident.

3.5 More Outlier Removal and Train-Test Splitting

After performing both single and multivariate linear regression and obtaining bad regression results, we looked back at our data and realized that numerous anomalous points still exist despite the removal of outliers using the 1.5 IQR rule. We believe that a second round of outlier removal would reduce noise even further within the training dataset, thus improving performance.

The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection technique that calculates the local density deviation of a particular data point with respect to its neighbors. Samples that have a significantly lower density than their neighbors are regarded as outliers.

Using this method, we removed 173 outliers from our dataset. We then resplit our data again into 70% training set and 30% test set.

3.6 Multivariate Linear Regression (LOF)

Following the outlier removal using LOF, we performed multivariate linear regression and got better results.

Figure 19: (Slightly) improved multivariate linear regression

Multivariate linear regression	Train (70%)		Test (30%)	
	R^2	MSE	R^2	MSE
Before LOF	0.4975	3732.58	0.4761	4081.10
After LOF	0.4897	3700.97	0.4840	4040.56

Figure 20: Goodness of Fit of multivariate linear regression before and after LOF

There was a general improvement in our linear regression model after removing 173 outliers. While explained variance decreased from 0.498 to 0.488 on the train set, it increased from 0.476 to 0.484 on the test set. MSE also decreased slightly from 3733 to 3701 on the train set and decreased from 4081 to 4041 on the test set. This is a very big improvement compared to the uncleaned dataset, especially on the test set, which means that the model is better able to predict unseen data.

4 Pycaret for Model Selection

Being new into the field of data science, we found ourselves lacking in knowing which models to use for our specific situation. Pycaret is a Python low-code library that helps us perform model selection and hyper-parameter tuning on a wide range of available models, simplifying the model selection process.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lightgbm	Light Gradient Boosting Machine	40.0089	3068.3440	55.2458	0.5862	0.3574	0.3083	6.3770
gbr	Gradient Boosting Regressor	41.1478	3140.5395	55.9412	0.5761	0.3613	0.3173	6.0570
rf	Random Forest Regressor	41.3889	3333.3456	57.6123	0.5493	0.3659	0.3143	5.5710
xgboost	Extreme Gradient Boosting	40.9909	3357.3688	57.8657	0.5434	0.3721	0.3112	5.8760
knn	K Neighbors Regressor	42.8154	3500.0219	59.0795	0.5262	0.3768	0.3247	6.4330
et	Extra Trees Regressor	42.9474	3652.9113	60.2666	0.5066	0.3819	0.3267	5.9220
lar	Least Angle Regression	47.4167	3894.8397	62.2785	0.4747	0.4583	0.3864	6.1490
br	Bayesian Ridge	47.4258	3894.9293	62.2790	0.4747	0.4551	0.3866	11.8340
ridge	Ridge Regression	47.4190	3894.8123	62.2782	0.4747	0.4573	0.3865	5.7180
lr	Linear Regression	47.4167	3894.8397	62.2785	0.4747	0.4583	0.3864	6.3670
lasso	Lasso Regression	47.5357	3899.6604	62.3182	0.4741	0.4409	0.3882	5.7560
llar	Lasso Least Angle Regression	47.5356	3899.6431	62.3181	0.4741	0.4409	0.3882	6.4020
ada	AdaBoost Regressor	49.2130	3908.3907	62.4238	0.4719	0.4273	0.4239	6.3180
huber	Huber Regressor	46.6085	3948.2432	62.6963	0.4677	0.5063	0.3588	5.9300
dt	Decision Tree Regressor	45.0644	4221.8664	64.7176	0.4287	0.4010	0.3349	6.4600
en	Elastic Net	52.4799	4497.6640	66.9553	0.3930	0.4624	0.4576	5.3030
omp	Orthogonal Matching Pursuit	57.4863	5243.5527	72.3209	0.2904	0.4954	0.5030	5.7370
dummy	Dummy Regressor	70.3966	7437.7647	86.1677	-0.0046	0.6061	0.6709	5.9430
par	Passive Aggressive Regressor	74.7348	9454.7036	92.1292	-0.2896	0.6301	0.6663	9.5110

Figure 21: Pycaret results of prediction models used

4.1 Light Gradient Boosting Machine

From the model selection process, we can see that the best prediction model is the Light Gradient Boosting Machine (LGBM).

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	36.4888	2567.8046	50.6735	0.6441	0.3496	0.3044
1	36.5830	2653.8053	51.5151	0.6264	0.3309	0.2748
2	37.8864	2553.4996	50.5322	0.5981	0.3506	0.3022
3	38.1014	2569.5545	50.6908	0.6582	0.3297	0.2785
4	41.7501	3406.1219	58.3620	0.5453	0.3712	0.3267
5	40.6408	3040.4463	55.1402	0.5542	0.3621	0.3071
6	43.3481	3546.6216	59.5535	0.5546	0.3819	0.3406
7	43.0660	3831.9495	61.9027	0.5632	0.3824	0.3380
8	42.4496	3448.6439	58.7252	0.5227	0.3644	0.3068
9	39.7749	3064.9924	55.3624	0.5953	0.3511	0.3043
Mean	40.0089	3068.3440	55.2458	0.5862	0.3574	0.3083
Std	2.4942	448.0616	4.0312	0.0432	0.0176	0.0208

Figure 22: 10 fold regression results of LGBM model using Pycaret

Prediction Model	R^2	MSE
(After LOF) Multivariate Linear Regression	0.4839	4040.56
LGBM	0.5862	3068.3440

Figure 23: Comparison between multivariate linear regression and LGBM

4.1.1 Hypertuning the Model

We used Pycaret's inbuilt hypertuning method to optimize our model. According to Pycaret's documentation, the *tune_model* function tunes the hyperparameters of the model using 10-fold cross-validation.

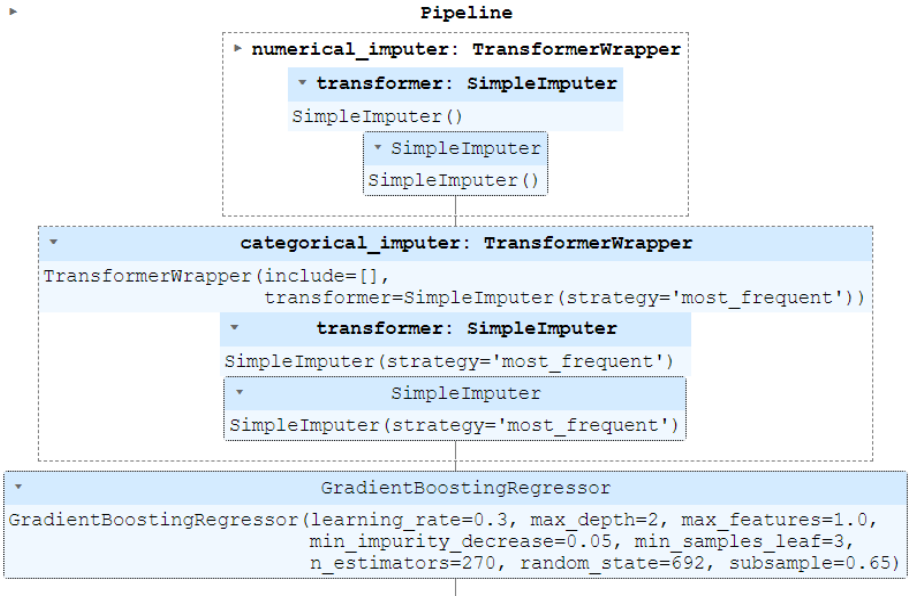

Figure 24: LGBM Model Prediction pipeline

4.1.2 LGBM Regressor Model Evaluation

The plot below shows the explained variance of our model on the train and test sets.

Figure 25: Explained Variance of the LGBM model

As seen from the results, the model performed decently well on both train and test sets, with explained variance being lower at 0.620 on the train set as compared to 0.636 on the test set.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Light Gradient Boosting Machine	39.8425	2929.2446	54.1225	0.6357	0.3515	0.3113

Figure 26: MSE and explained Variance of test set

MSE is higher at 3010 on the train set compared to 2929 on the test set which shows our model is good at preventing over and under fitting of data. This is currently our most accurate model so far.

4.2 Price Prediction using LGBM Regressor Model

We then used the model to predict AirBnB listing prices.

	id	property_type	room_type	neighbourhood	accommodates	bedrooms	beds	price	predicted_price
1	3075044	0	1	22	2	1.0	1.0	65	82.420041
6	5706985	0	0	30	3	1.0	2.0	100	189.599518
7	2843445	8	1	22	2	1.0	1.0	75	70.500912
8	753446	4	1	22	2	1.0	2.0	58	92.259216

Figure 27: Sample of dataset with predicted price

5 Sequential Keras Model

We will now create the Sequential Deep Learning Model using Keras. We've explained more about Keras and our methodology in the appendix.

5.1 Setting Up and Training the Keras Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               896

 dense_1 (Dense)             (None, 256)               33024

 dropout (Dropout)           (None, 256)               0

 dense_2 (Dense)             (None, 256)               65792

 dense_3 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_4 (Dense)             (None, 64)                8256

 dense_5 (Dense)             (None, 1)                 65

=================================================================
Total params: 140,929
Trainable params: 140,929
Non-trainable params: 0
_________________________________________________________________
```

Figure 28: Layers in our Neural Network

The figure above shows how we set up our 3 layers : the input, hidden layers and the output layer.

5.2 Predicting and Evaluating the Keras Model
After training the model, we used it to predict it on our test dataset and these are the results that we got.

```
Goodness of Fit of Model        Test Dataset
Explained Variance (R^2)        : 0.48365028998272463
Mean Squared Error (MSE)        : 3734.8425827911897
```

Figure 29: Keras Model Evaluation

The Keras model performed better than our Multivariate Linear Regression model but worse than our Light Gradient Boosting Machine model. The Keras model had similar Explained Variance values of 0.484 on the test set, but had a lower Explained Variance value than our Light Gradient Boosting Machine model's Explained Variance value of 0.636 on the test set. The Keras model also had a lower MSE value of 3735 on the test set compared to the MSE value of 4041 in our Multivariate Linear Regression Model on the test set but higher than our MSE value of 2929 in our Light Gradient Boosting Machine model on the test set.

This may be due to the fact that our Keras model will improve exponentially with more data due to its deep learning nature, and we do not have enough data to feed the model.

5.3 Finalizing our Model Selection
From the previous sections, we concluded that our tuned model of 'Light Gradient Boosting Machine' selected by Pycaret is the most accurate model to predict prices based off all our variables, since it has the highest explained variance value of 0.636 and the lowest MSE value of 2929 on the test set.

Now, we will finalise our dataframe on our price predictions of the Airbnb listings and have obtained the "predicted_prices" data frame of predicted prices of AirBnB listings created by our 'Light Gradient Boosting Machine' model. We will use it later on together with our Sentiment Analysis results to obtain a final prediction of how worth it the AirBnB is.

6 Sentiment Analysis

We next performed sentiment analysis to better understand the worded comments made on the guests' AirBnB experiences. For simplicity, we only retained comments written in English using .

6.1 Lemmatization

Generally, written comments contain some key words and many irrelevant filler words. Our aim in this section was to removing punctuation, tokenize words, removing stopwords and finally to lemmatize the words into their root form. This allows us to reduce the vocabulary size while grouping words with similar meanings.

6.2 Gensim Word2Vec

Gensim is an algorithm based on neural networks, using large amounts of unannotated plain text to learn the relationship between words. This gives us a spatial dimension with words of the same meaning close to each other. For example, 'strong' and 'powerful' will be close to each other. We used a lookup window of 4, which allows the neural network to learn to predict words based on 4 words from the left or right of the reference word.

comments	lemmatized	clean
my stay at islam's place was really cool! good...	[stay, islam, place, really, cool, good, locat...	stay islam place really cool good location 5mi...
great location for both airport and city - gre...	[great, location, airport, city, great, amenit...	great location airport city great amenity hous...
we really enjoyed our stay at islams house. fr...	[really, enjoyed, stay, islam, house, outside,...	really enjoyed stay islam house outside house ...
the room was nice and clean and so were the co...	[room, nice, clean, commodity, close, airport,...	room nice clean commodity close airport metro ...
great location. just 5 mins walk from the airp...	[great, location, 5, min, walk, airport, stati...	great location 5 min walk airport station good...

Figure 30: Example lemmatized and clean data retrieved from AirBnB comments

6.3 K-Means Clustering

We used K-Means clustering to perform unsupervised clustering, with n=2 clusters to differentiate clusters of positive comments and negative comments.

6.3.1 Sentiment Scores of Individual Words

Starting off with 50 words as centroids, we assigned sentiment scores based on the other words' proximity to these centroids.

	words	vectors	cluster	label	distance	sentiment
0	great	[-0.14181459, -0.120310664, -0.03921396, 0.145...	1	1	1.314163	1.314163
1	stay	[-0.12412196, 0.085356, 0.10092397, 0.09704713...	1	1	1.291962	1.291962
2	place	[-0.20771533, -0.07208588, -0.076045305, 0.089...	1	1	1.295856	1.295856
3	boston	[-0.10423143, 0.07090261, 0.09206961, 0.111494...	1	1	1.148196	1.148196
4	apartment	[-0.13650987, -0.023392854, -0.093179494, -0.0...	1	1	1.071723	1.071723

Figure 31: Sentiment scores of individual words

6.3.2 Sentiment Scores of Sentences

We start off by defining the term TF-IDF, which stands for Term Frequency (TF) and Inverse Document Frequency (IDF). TF-IDF defines the importance of a term by taking into consideration the importance of that term in a single document, and scaling it by its importance across all documents. Combined, the TF-IDF score indicates the importance of a word relative to the number of times it has appeared in a document. Using the TF-IDF score in our model allows unique words to contribute a higher weight to the sentiment scores.

In order to find the relative importance of each word, we took the dot product of its sentiment score with its TF-IDF score. Subsequently, we added these individual sentiment scores up to

obtain a sentiment rate for each individual AirBnB review. Sentiment rates above zero are positive, while sentiment rates below zero are negative.

sentiment_score	tfidf_score	clean	comments	sentiment_rate
[1.2919618762033729, 1.0521648518560347, 1.295...	[1.8514771938373173, 16.871155479817713, 5.975...	stay islam place really cool good location 5mi...	my stay at islam's place was really cool! good...	79.960931
[1.3141630998290912, 1.1358230663866915, 0.986...	[3.5722321487579984, 2.106513772141474, 4.0421...	great location airport city great amenity hous...	great location for both airport and city - gre...	43.102801
[1.1813289473269477, 1.3216291719430917, 1.291...	[5.769524870355626, 3.53342674378411, 3.702954...	really enjoyed stay islam house outside house ...	we really enjoyed our stay at islams house. fr...	69.959889
[-1.0495620117793203, 1.1137777790720023, 1.10...	[2.3725884415090777, 2.4220556278485077, 2.088...	room nice clean commodity close airport metro ...	the room was nice and clean and so were the co...	69.257423
[1.3141630998290912, 1.1358230663866915, -1.08...	[1.7861160743789992, 2.106513772141474, 4.3037...	great location 5 min walk airport station good...	great location. just 5 mins walk from the airp...	32.345807

Figure 32: Example sentiment rates of sentences

6.4 Manual Labelling of Model

We took out a random sample of 2000 reviews to test the effectiveness of our model in predicting positive and negative sentiments. Besides reading to understand each review's overall sentiment, we judged positivity based on our perceived likelihood that the reviewer would return to the AirBnB on future stays. By manually reading each review and assigning a value of 1 for positive reviews and 0 for negative reviews, we were able to compute the accuracy of our model along with a confusion matrix.

	Predicted: 0	**Predicted: 1**
Actual: 0	56	43
Actual: 1	421	1480

Figure 33: Confusion Matrix of model centered at 0

Using the random sample of 2000 entries, we achieved an accuracy of 0.768.

6.5 Recentering Sentiment Rate

We noticed in our manual labelling of positive and negative sentiments that the model tends to falsely predict a negative sentiment when the overall sentiment was in fact positive. This is substantiated by the high false negative rate of 22.1%. We decided to categorise negative sentiments below -50 instead of 0. Any reviews with sentiment rates above -50 are now considered positive reviews. The recentered axis of -50 was arbitrarily decided.

	Predicted: 0	**Predicted: 1**
Actual: 0	27	72
Actual: 1	201	1700

Figure 34: Confusion Matrix of model centered at -50

This change improved the model's accuracy to 0.8635. We will then save this new sentiment model prediction dataframe as 'final_sentiment.csv'.

7 Combined Analysis

Given our model price prediction and sentiment analysis from the previous 2 parts, we now merge both dataframes ‘predicted_prices.csv’ and ‘final_sentiment.csv’ together using listing id as the key. Each AirBnB has multiple reviews, hence we took the mean of the sentiment scores of each AirBnB listing.

7.1 Calculating Difference between Predicted and Actual Price

With the actual listed prices and our predicted prices, we computed the price difference between these 2 values. A positive difference indicates that a listing is underpriced, while a negative difference indicates that a listing is overpriced.

	id	property_type	room_type	neighbourhood	accommodates	bedrooms	beds	comments	price	predicted_price	sentiment_rate	price_difference
0	5506	8.0	1.0	23.0	2.0	1.0	1.0	terry's hotel alterntv in boston was a perfect...	145.0	85.833479	60.281931	-59.166521
1	9765	0.0	0.0	26.0	2.0	0.0	1.0	everything was great. i wish they had advertis...	229.0	168.954652	-70.625988	-60.045348
2	9824	0.0	0.0	1.0	2.0	0.0	1.0	the location for the back bay studio (on heref...	209.0	168.048695	-30.228172	-40.951305
3	9857	0.0	0.0	1.0	4.0	2.0	2.0	the apartment was very nice and comfortable, a...	342.0	283.462564	-43.799172	-58.537436
4	9860	0.0	0.0	1.0	2.0	1.0	1.0	had no problem getting this plave last minute,...	251.0	215.438168	-67.088560	-35.561832

Figure 35: sample dataframe showing price differences

7.2 Grading Each AirBnB

Finally, we graded each AirBnB listing with grades A, B or C. An ‘A’ grade indicates positive sentiment and underpriced, which implies that the listing is a steal for both quality and value. A ‘B’ grade indicates that the AirBnB does not warrant its listing price despite guests having positive experiences during their stay. Alternatively, a ‘B’ grade could also indicate an AirBnB that is no frills but is value for money. Lastly, a ‘C’ grade indicates an AirBnB listing that is both overpriced and is unable to satisfy guests. We will save this finalised dataframe as ‘airbnb_prediction’, which will allow travellers to easily access our new AirBnB grading system.

	id	property_type	room_type	neighbourhood	accommodates	bedrooms	beds	comments	price	predicted_price	sentiment_rate	price_difference	grade
0	5506	8.0	1.0	23.0	2.0	1.0	1.0	terry's hotel alterntv in boston was a perfect...	145.0	85.833479	60.281931	-59.166521	B
1	9765	0.0	0.0	26.0	2.0	0.0	1.0	everything was great. i wish they had advertis...	229.0	168.954652	-70.625988	-60.045348	C
2	9824	0.0	0.0	1.0	2.0	0.0	1.0	the location for the back bay studio (on heref...	209.0	168.048695	-30.228172	-40.951305	C
3	9857	0.0	0.0	1.0	4.0	2.0	2.0	the apartment was very nice and comfortable, a...	342.0	283.462564	-43.799172	-58.537436	C
4	9860	0.0	0.0	1.0	2.0	1.0	1.0	had no problem getting this plave last minute,...	251.0	215.438168	-67.088560	-35.561832	C

Figure 36: sample dataframe showing AirBnB grades

8 Conclusion

Having created an AirBnB grading system that is more well-rounded than an arbitrary, non-standardized star grading system, we can determine quantitatively how well each listing is rated. This allows us to better understand if an AirBnB is worth renting, without the hassle of reading comments and deciphering each reviewer’s rationale for star rating. We believe that this project is scalable to more recent AirBnB listings and will be useful for future travels.

9 Appendix

9.1 Pycaret

PyCaret is an open-source, low-code machine learning library in Python that automates machine learning workflows. It is an end-to-end machine learning and model management tool that speeds up the dataset training cycle exponentially.

In comparison to the other open-source machine learning libraries, PyCaret is an alternate low-code library that can be used to replace hundreds of lines of code with few lines only. This makes model selection and training exponentially fast and efficient. PyCaret is essentially a Python wrapper around several machine learning libraries and frameworks such as scikit-learn, LightGBM, RidgeRegression, Lasso, Adaboost and many more.

It comes with a lot of functionality to interact with the model and analyze the performance and the results of the dataset. All the standard plots like confusion matrix, AUC, residuals, and feature importance are available for all models.

9.2 Keras

Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models.

Deep Learning Neural Networks have 3 main layers, which are the input layer, the hidden layer(s), and the output layer. In each layer, there are activation functions that have weights assigned to them to feed the input of the next layer of the Neural Network with the output of the previous layer, which will tune the model.

- Activation functions are functions used to determine the output of each node in the Neural network. We will use the Rectified Linear Unit (ReLU) Activation function in our input and hidden layers, which gives a positive output if the input is a positive, and a 0 if the input is negative. These functions are weighted and biased before passing through to the next layer in the network.

In the final layer, we will use the Linear Activation Function to output both positive and negative values for our model prediction. We will use the Adam (Adaptive Moment estimation) optimizer, to implement gradient descent on our Keras model, and then calculate the MSE of our tuned Keras model.

9.3 TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency) vectorizes/scores a word by multiplying the word's Term Frequency (TF) with the Inverse Document Frequency (IDF). Term Frequency(TF) of a term or word is the number of times the term appears in a document compared to the total number of words in the document. Inverse Document Frequency (IDF) looks at how common (or uncommon) a word is amongst the corpus of words. TF gives us information on how often a term appears in a document and IDF gives us information about the relative rarity of a term in the collection of documents. By multiplying these values together we can get our final TF-IDF value. The higher the TF-IDF score the more important or relevant the term is; as a term gets less relevant, its TF-IDF score will approach 0.