MBAN 6110 T
Delina Ivanova
July 23, 2024
Group 10

Data-Driven Insights into Airbnb Pricing for New York City

Alekhya Erikipati Yihua Chen Christine Tang
Narotam Dhaliwal Pui Ching Queenie Sung

Contents

Executive Summary

The report aims to provide actionable insights for Airbnb hosts in New York City (NYC) to optimize the hosts' property portfolio and increase daily revenue. The focus is on increasing the price per day due to 2023 NYC regulations, limiting available Airbnb properties. The problem statement initially focused on predicting Airbnb prices (for regression modelling), however due to poor performance, the problem shifted to classification modelling, determining if a property is classified as Class 0 (low-price per day) or Class 1 (high-price per day).

Leveraging the "Airbnb Open Data", the analytical approach consisted of (1) Preliminary Exploratory Data Analysis (EDA), (2) Data preparation, cleaning and feature engineering, (3) Model Development and Evaluation, and (4) Business insights, (5) Conclusions and Next steps.

After various modelling steps, it was determined that the optimal model was Random Forest Classification for binary classification (Class 0 and Class 1), with a high precision of 0.69, and higher Cross-Validation score versus other models, and a ROC AUC of 0.80, suggesting the model has high capabilities to distinguish between the two classes. From a business perspective, "precision" was emphasized for model tuning as there was a necessity to correctly identify high-price properties for predictions. This model was also selected for interpretability and ability to handle non-linear relationships, especially with the target variable (price).

From feature importance, key recommendations were provided: (1) Focusing on properties with closer proximity to NYC attractions, (2) Improving host behaviours, (3) Engagement metrics, and certain characteristics to avoid, such as private rooms and strict cancellation policies. These insights are necessary to optimize the properties for higher prices per day, improving revenue potential for Airbnb hosts.

Introduction: Dataset Selection and Justification

This report will focus on providing valuable and actionable insights to support property management companies and individual property owners (known as "Airbnb hosts"), with properties listed for Airbnb rentals in New York City (NYC), USA. Due to the pandemic, Airbnb hosts focus on recovering financially due to lockdowns affecting operations (Kolomatsky, 2021). Specifically, the client (Airbnb hosts) would like to discover opportunities regarding how to generate more revenue per day. There are four components for revenue increase: increasing the price per day stay, increasing the number of days stayed, increasing the number of guests for each "stay" (i.e., if the host charges for extra guests), or increasing the number of properties.

The scope will focus on uncovering opportunities to increase price per day. This is determined from two reasons: (1) Starting September 2023, New York City has passed regulations imposing more rules and regulations for Airbnb listings, potentially affecting the total number of properties available (Lung, 2023), and (2) The remaining two factors (number of days and number of guests stayed) do not present revenue increase opportunities for hosts, as they have upper limits (e.g., there are only 365 days in a year). See Appendix A for the KPI Tree.

To address the report scope, the dataset "Airbnb Open Data", last updated in 2022, was utilized. You may find the link to the dataset here. This dataset was selected to address the issue of increasing the price (and finding revenue generation opportunities). This is an appropriate dataset for the report objective, because (1) The dataset contains over 100,000 rows of

information with 24+ columns to begin the analysis, (2) The dataset's columns were relevant to analyzing the issue, such as price, host information, neighbourhood, availability, property construction year, information about reviews and house rules, and (3) The dataset focused on NYC in the USA, which is the geographical focus of the report.

Problem Statement and Hypothesis

The report scope's problem statement is to predict the price of Airbnb listings from the properties in New York City. The problem and hypothesis statements were originally framed for solving a Regression problem, to predict a single value (continuous outcome) based on input features. However, the results were poor from the regression model, likely due to the nature of the data not fitting well with the prediction.

The analytical approach was to switch to a classification problem, where the objective was determining if a property was considered a "low-price" property for Airbnb rentals (Class 0), vs. "high price" property for Airbnb rentals (Class 1).

(REGRESSION PROBLEM) Hypothesis Testing:

- **Null Hypothesis:** The price per Airbnb listing will be the same, regardless of the influence of factors.
- **Alternative Hypothesis**: The price per Airbnb listing is influenced by at least one factor.

(CLASSIFICATION PROBLEM) Hypothesis Testing:

- **Null Hypothesis**: An Airbnb listing being classified under Class 0 or Class 1 is not influenced by any factors.
- **Alternative Hypothesis:** An Airbnb listing being classified under Class 0 or Class 1 is influenced by at least one factor.
    - Note that Class 0 is "Low-Price" properties" for Airbnb rentals, and Class 1 is "High-Price" properties for Airbnb rentals (i.e. price per day).

Exploratory Data Analysis (EDA)

Beginning with summary statistics, "df.describe()" to provide a summary of all statistics, such as count, mean, maximums/minimums, etc. It was determined that there were 102,599 rows from the "id" column, and there were noticeably missing values or an unequal number of non-null counts for each column. This was apparent from "last review", "reviews per month", and "house_rules". Redundant columns, such as "license", were later removed.

This provided important insights, such as outliers, missing values, and data inconsistencies. Notably, it was observed that there were illogical values under the column "minimum nights" and "availability 365", for example, 5645 for "minimum nights". Additionally, columns related to monetary value, such as "price" and "service fee", had an additional symbol ($) included, and that needed removal. This was determined by comparing variable identifications ("dtypes") and noticing inconsistencies. Additionally, it was determined that the "ID" column was not unique, and there were ~1082 rows of duplicated rows, with each row having the same information. As each "ID" row should be unique and refer to a specific Airbnb listing, duplicated rows needed removal. Regarding missing data, it was observed that "last review" and "reviews per month" likely had missing data due to "missing not at random" (MNAR), suggesting a systematic reason

for missing information – that is, the property likely was not available or not booked, resulting in missing information. For “house rules”, the reason is likely MNAR or MAR (missing at random), where hosts likely did not have rules originally, or the hosts did not feel the need to provide them (i.e., their own perspective or leniency). Regardless, all three major sources of missing data will have a difficult time for imputation and therefore, the decision was to drop the rows (listwise deletion) later.

Identifying Numerical and Categorical Variables

The columns in the dataset are divided into numerical and categorical categories. This division helps in applying relevant analysis techniques for each type of variable. Numerical variables included “price”, “service fee”, “minimum_nights”, “number_of_reviews”, “review_rate_number”, “availability 365” and more. Categorical variables included “neighbourhood_group”, “room_type”, “instant_bookable”, “cancellation_policy”, “instant_bookable” and more.

Univariate analysis was conducted for both numerical and categorical variables to determine distribution. For EDA, boxplots and bar plots visualized categorical variables, and histograms and boxplots visualized numerical variables.

Important insights were derived. Firstly, high-demand areas included Manhattan and Brooklyn (with “Bedford-Stuyvesant and Williamsburg” as the most common neighbourhood), and room types were predominantly “entire homes/apartments”. Categorical variables such as host identity verification, instant bookable and cancellation policies, were evenly distributed in terms of count. Secondly, for numerical variables, the mean price per day stay is $625 USD, and an even distribution of all price points, ranging from $50 to $1200 USD/day. The number of reviews and reviews per month is left-skewed, suggesting a presence of outliers on the right side of the distribution, with a median of 7 and 0.74 respectively. The availability needs further analysis, as it’s left-skewed and median of 96 days. The mean construction year is 2012. The mean review rate number is 3 (out of 5), suggesting most reviews are rated as “OK/Average”.

Initial bivariate analysis, versus “price” (the main target variable) provides insights, such as unconfirmed and confirmed listings have the same mean price. The neighbourhood plot vs. prices reveals spelling errors that require fixing, but slight variations for mean. The mean price per cancellation policy is very similar, with price by room type is slightly higher for hotel rooms.

The bivariate analysis for numerical variables versus price reveals from the scatterplot that the relationship between the variables is not linear, as there are often clusters towards one side. This is valuable information as it suggests the presence of non-linear relationships with numerical variables. It was also determined that price and service fee were not normally distributed from later, looking at histograms and QQ-plot.

Data Cleaning

Data Cleaning was conducted concurrently with EDA, with the (1) Removal of the $ symbol from “price” and “service fee”, (2) Duplicated rows, under the “ID” were removed, (3) Fixing the values for minimum nights (i.e. imputing negative values with a 1-day stay minimum), and “availability 365” by imputing the zeros for any value under zero, and also imputing any value over 365 with 365 days (since the metric is the availability of bookings in the next 365 days), (4) “Country”, “Country Code” and “License” were removed, as the dataset already established the data was from NYC in the USA, (5) Fixing the spellings under “neighbourhood group”, (5) Standardizing the column names by applying lower-case and an underscore, (6) Fixing a small

number of rows with future review dates (i.e. beyond 2022 does not make sense, since the data was collected up to 2022). Note that in the process of dropping rows from last_review_NEW, rows with null-values or missing were also dropped as well. As the decision was to drop the 15% of missing rows or null-values from reviews_per_month and last_reviews, this step also supported a subsequent cleaning step.It is important to note that, after cleaning the duplicate rows via "ID", it was determined that each host ID was unique, meaning that each host ID has exactly one only property in the dataset. It was determined to be essential to keep these columns with high amounts of missing data, as imputation would be difficult due to the MNAR nature, but also these columns possessed important information regarding the Airbnb listing performance, and the columns should be retained.

Finally, the notable issue is the high percentage of missing rows, primarily from reviews per month and the last reviews column. This represented roughly 15% of the data, and as there are ~85K rows for analysis and nature (MNAR), it was determined to be the most appropriate action as imputation would be difficult. Do note that the 6th cleaning step (dropping rows that were beyond December 31 2022), also dropped many rows of missing data from both columns as well. Regardless, the decision was to drop rows of missing information from these two columns. Other columns also missing data as well, accounting for <3% of the remaining 85K rows, and thus dropped as well.

Feature Engineering

Firstly, for feature engineering, the first step was imputing "unknown" into missing rows for "house_rules", which accounts for a large percentage of missing rows. This is to ensure that amenities may be extracted from this column, and imputation would be difficult to the MNAR nature, the missing rows were imputed with "unknown" for handling later, especially for amenities extraction (i.e. null-values will not allow for this extraction later).

Next, the "last_review" column was converted into a Python datetime variable, to create year and month, which will be used later to track seasonality. Furthermore, variables were one-hot encoded, namely host_identity_verified, neighbourhood_group, instant_bookable, cancellation_policy and room_type. The encoding was completed sooner for easier identification of feature importance after model development and assessment. Using December 31 2022, a feature called "days since last review from date" was created, calculating the time since the last review to that date, indicating "freshness" or recency of the listing. Seasonality was determined later on using the "last review month" column, encoding it from Spring, Summer, Fall and Winter. This new variable "season" was later encoded.

Leveraging Natural Language Processing, word-counting was conducted using the house_rules column, where major amenities were determined first by the 30 most common English word via "CountVectorizer". Next, major amenities words were also accounted for, such as "subway, park, stadium, park, zoo, library, etc". These were then applied as columns as encoded variables (1 indicating the word was mentioned, 0 was not). Another feature created was the allowance and prohibition of parties, which was encoded. Another important feature was house_rules sentiment (house_rules_sentiment), using the TextBlob package. These meaningful variables were created, as they captured information regarding amenities offered nearby and some input about host behaviour.

Finally, an important feature created was the "proximity_score" which leveraged the Google Maps Platform API. Using the API, the longitude and latitude of each property's listing were compared against the top six NYC attractions (Times Square, Central Park, Status of Liberty,

Empire State Building, Brooklyn Bridge, and Metropolitan Museum of Art). Then, a proximity score was calculated per row by determining the distance of each property to the six attractions, and summing these values. Therefore, a property that is close to the six locations will have a lower proximity score, and properties that are farther away will have a higher proximity score. Please note that the results were extracted for proximity key, merged with the dataframe at this point, and then the CSV was utilized in Part 5 to load this dataset for simplicity, and as the retrieval process of the Google Maps API information was computationally high, and time-intensive.

Prior to model development, multi-collinearity analysis was conducted to determine which columns to consider for the model development aspect. This is important to not introduce bias, having highly correlated variables will make the model unreliable and artificially inflate the performance metrics for the model, and make it harder to assess the feature importance. Using the Correlation Matrix (Pearson R) and Variance Inflation Factor (VIF), each feature was assessed, and it was determined that construction_year and last_review variables (year and month) had high multicollinearity. They were removed, along with “service fee”, leveraging domain knowledge of service fee being a percentage of the price, resulting in high collinearity.

Finally, the predictor variables were utilized for model development: minimum nights, number of reviews, reviews per month, review rate number, calculated host listings count, availability 365, days since last review, variables from counting (Natural Language Processing), seasons variable, house rules sentiment, and proximity score. Special attention was conducted throughout model development ensuring “service fee” was not included due to high collinearity.

Model Development

The first model development focused on **regression modelling.** Beginning model development, the predictor variables were determined and “price” was the target variable (for prediction). Please note: the encoding steps were completed prior to the model development aspect.

Firstly, data preparation and preprocessing steps were required, where (1) The Regression models were determined, (2) The X and y splits were conducted (so train and test splitting) based on the target and predictor variables, with the random state 42 and test size default of 20% (80% for training), (3) Data Scaling and Normalization was conducted, using StandardScaler and MinMax Scaler (MinMax mostly for regression problems, Standard Scaler for classification problems). This was necessary as unscaled and varying degrees of numerical values would affect model performance. To simplify this process, some functions were created for model evaluation in the regression aspect, which created the pipeline (inserting the steps for preprocessor steps along with the model name). This resulted in simple tables displaying evaluation metrics.

For **regression**, the following eight models were assessed: linear regression, lasso and ridge Regression, k-nearest neighbour, decision tree, random forest, gradient boosting, and support vector machine regressors. These models were selected due to their potential for interpretability to understand pricing. From each model, it’s expected to derive insights for pricing.

- **Linear, Lasso and Ridge Regression:** The models are simple, and provide straightforward interpretations (i.e. coefficients, and feature selection for Lasso/Ridge)
- **K-Nearest Neighbours (KNN) Regressor:** Selected to potentially capture neighbouring points, with similar properties for price prediction (continuous value) in this case.

- **Decision Tree and Random Forest Regressor:** This is ideal for capturing non-linear relationships, which are observed from several variables with “price”.
- **Gradient Boosting Regressor**: The model can improve prediction accuracy, and as the model has many columns (~69 columns after encoding steps), iteratively finding improvements is a good approach.
- **Support Vector Machine (SVM) Regressor**: This model is effective for exploring higher dimensions, especially as there are many columns created.

Afterwards, hyperparameter tuning was applied with various parameters. GridSearchCV, which exhaustively searches for the optimal hyperparameters for each model, was employed initially for regression modelling. Due to the high computational cost of searching, along with the K-Fold Cross Validation (k iterations for validation set and k-1 folds remainder utilized for training set), it was determined to utilize RandomSearchCV for the remaining classification modelling. See Appendix C for a list of the modelling parameters.

Unfortunately, after evaluating metrics (which will be discussed in Model Evaluation), it was determined that the models fit poorly. One underlying concern was the presence of outliers, so Linear/Lasso/Ridge regression was conducted at a 2nd attempt with winsorization of 95% to numerical variables (i.e., Minimum nights, number of reviews, reviews per month, calculated host listings count, availability 365), however, after running the models with the newly winsorized data, the model evaluation was still quite poor.

A final regression modelling attempt required applying Boxcox transformation to the “price” variable, especially as the price was not normally distributed, and from the Linear, Lasso and Ridge models, it appeared that the residuals were not normally distributed across the fitted values, along with the QQ plot providing similar insights. Despite this, the model results were not satisfactory. This was likely due to the non-linear relationships between the features and the target variable, “price”.

Overall, despite careful cleaning and data preparation steps, from observing data distribution, it was determined that the non-linearity relationships affected the performance of models such as linear modelling, which are high “bias” and assume a linear relationship. Additionally, there may be high variance, as seen from the high RMSE across all models, poor Cross-Validation scores across all models, and a near-zero adjusted R-Squared value (which adjusts a model if there are too many features that might explain the variability of dependent variables).

In the second modelling set, **binary classification modelling** was conducted through the following models: Logistic Regression, K-Nearest Neighbours Classification, Random Forest, Gradient Boosting, XGB Classifier and LGBM Classification. Regarding binary classes (Class 0 = low-price per day and Class 1 = high-price per day), the class division was determined by dividing the classes using price median – as the price distribution was equally, uniformly distributed across all price points, this resulted in equal counts for each class (avoiding the issue of data imbalance).

Regarding classification modelling and connecting to the business problem, the models were developed with a focus on “***precision***” when conducting RandomSearch CV and GridSearchCV, with the scoring parameter assessing performance based on precision. This approach helps Airbnb hosts in identifying high-price properties (Class 1) by setting a higher threshold for classification, ensuring that time and investments are made to the right properties. While this will reduce false positives, there is a trade-off of missing other high-price properties.

The following models were selected for binary classification modelling:

- **Logistic Regression**: Simple model for binary classification that offers efficiency and potential interpretability, indicating a relationship between each feature and the probability of Class 0 and Class 1 (outcomes)
- **K-Nearest Neighbours Classification:** Captures non-linear relationships and offers an opportunity for uncovering patterns with nearby points (clustering) to determine which point belongs to which class
- **Random Forest Classification**: Captures complex, non-linear relationships, providing insights on feature importance that helped shape the model, handling a variety of data well
- **Gradient Boosting Classification**: Higher predictive accuracy, building on sequential models for better performance to determine interactions between features.
- **XGBoost Classification:** Optimized version of Gradient Boosting.
- **LightGBM Classification:** Faster version of the traditional Gradient Boosting model, handling larger data to find patterns with Class 0 and Class 1.

After performing binary classification modelling, it was determined that the models were performing better with evaluation metrics, please see Appendix E. From there, the top three most promising models were selected for hyperparameter tuning: Logistic Regression, Random Forest Classifier, and Neighbours. Finally, **multi-class classification modelling** was subsequently conducted. There were three specific classes – Class 0 was “low-price per stay”, Class 1 was “medium” price per stay, and Class 2 was “high” price per stay. After setting up the pipeline to assess the same six models as binary-class classification. Similar to binary classification, there was no class imbalance (equal counts).

Model Evaluation

Beginning with **Regression Modelling**, the following metrics were utilized to assess model performance: Root Mean Squared Error (RMSE), adjusted R-squared and Cross-Validation Scores (CV). RMSE was necessary to understand how much error there was in the predictions vs. test results, with lower RMSE being desired in models. Adjusted R-Squared was important to understand the proportion of variance explained (factoring in the additions of features), and CV scores assisted with assessing with several folds for performance, conducting train/test splits several times. The test split was 20%, and the training split was 80%.

As seen in Appendix E under Regression Modelling, almost every model had a near zero adjusted R2 and high RMSE between 280-330, and negative Cross-Validation Scores. Essentially, the regression models were ineffective for predicting Airbnb prices. The model’s predictions had high deviations from actual prices, suggesting poor predictive performance, and there is a poor explanation of the variation in the dependent variable (price). Critically, the negative CV score suggests the models are unable to generalize data well, suggesting an inability to understand the underlying patterns to predict price. While Random Forest Regressor did perform better, the overall poor performance of regression models highly suggests a poor fit to the data.

Regarding **Binary Classification Modelling**, the following metrics were utilized to assess model performance: Precision, Recall, and Mean CV-precision score. Additionally, the split was now “train/test/validation”, to ensure any higher precision results were consistent and robust, reducing model overfitting and allowing hyperparameter tuning. This was also conducted later

for multi-class classification modelling. From the business problem, the priority was “precision”, focusing on the accurate identification of Class 1 (high-price) for hosts to invest correctly for properties. Precision and Recall were necessary to evaluate, as Precision indicated the ability of the model to correctly identify Class 1 and minimize the risk of false positives, and Recall for capturing as many high-value properties as possible. The Mean CV-Precision score provided insights into the average precision across the multiple cross-validation folds.

There was a similar approach for Multi-class Classification modelling, where Precision, Recall and Mean CV-precision scores were evaluated. From Appendix E, the model performance metrics were similar to binary classification modelling prior to tuning, and as “Random Forest Classifier” in binary classification had better metrics, the focus was on binary classification.

From the model evaluation, **the optimal model was the Random Forest Classification Model from binary classification**, with the following parameters: {'n_estimators': 300, 'min_samples_split': 10, 'min_samples_leaf': 1, 'max_depth': None, 'bootstrap': False}. As seen from Appendix E, this model performs the highest in terms of precision, recall and mean-CV precision score, with an emphasis on high precision. Additionally, the ROC Curve and AUC score was high, at 0.80. This suggests the model has an 80% chance of being able to distinguish between the two classes (low and high priced Airbnb properties).

Business Case Insights from Modelling

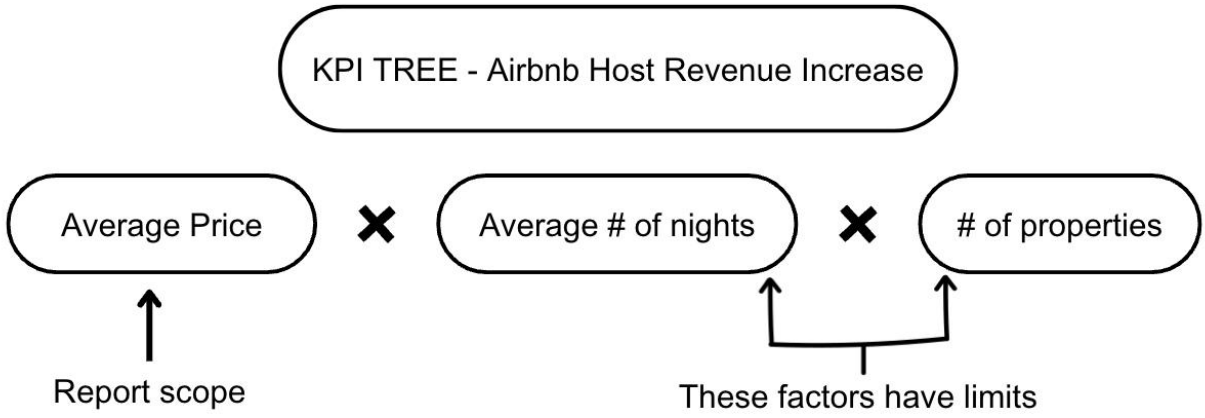
Per Appendix E, the analysis of feature importance from the Random Forest Classifier provides essential business insights for Airbnb hosts, who desire identifying and optimizing properties to justify higher property prices. From the top three features (proximity score, reviews per month, days since last review), and other features like house_rules_sentiment, suggest that properties with higher proximity scores to major attractions in NYC may influence pricing classification as “high price” (over $625 USD/night). For example, from feature importance and Spearman’s Correlation Matrix, some features such as neighbourhoods in Brooklyn may not contribute to the classification of high-price, suggesting location is also a factor for optimizing property to be high-price. Additionally, keeping properties active on Airbnb’s platform, as well as host behaviours (i.e., friendly and positive mindsets, allowing flexible cancellation policies) may also support properties having higher price justification, highlighted by the model for feature importance. Notably, having stricter cancellation policies and having a private room type does not support classification for high-price (Class 1).

Through the hypothesis testing (determining if the factors affect the classification of properties under high or low-price per day), the null hypothesis is rejected, and the findings suggest an Airbnb listing’s price classification is affected by at least one factor, such as proximity to attractions, engagement and host behaviours.

Conclusion and Next Steps

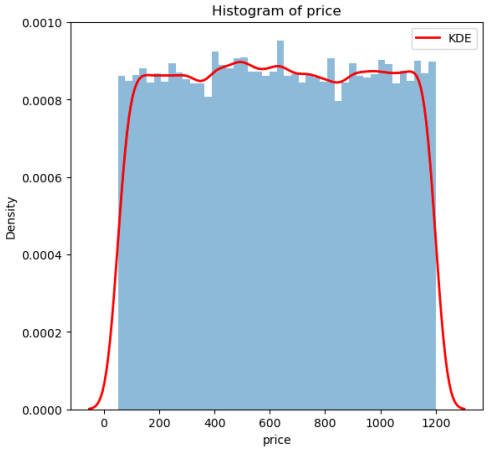
This report provides significant, valuable insights for Airbnb hosts to improve revenue generation, with the primary focus on understanding Airbnb listing prices. It is highly recommended that hosts leverage the model findings, to begin optimizing property portfolios to increase listing prices and to remain competitive. Regulatory changes in NYC will significantly reduce the number of short-term rentals, stressing the importance of optimizing a host’s portfolio of properties to increase revenue. By focusing on these business insights, hosts can enhance listing visibility and attractiveness, and improve profitability through higher prices per day.

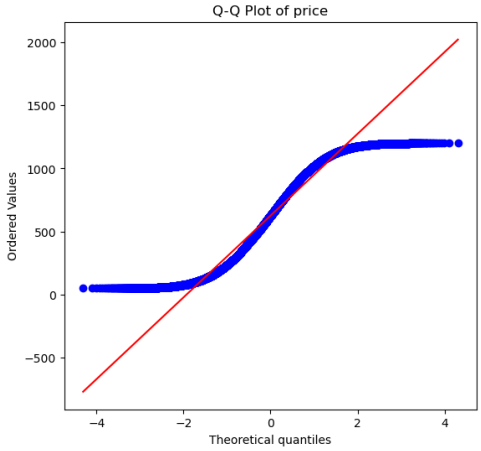
Appendix A: KPI Tree – Revenue Increase for Airbnb Hosts

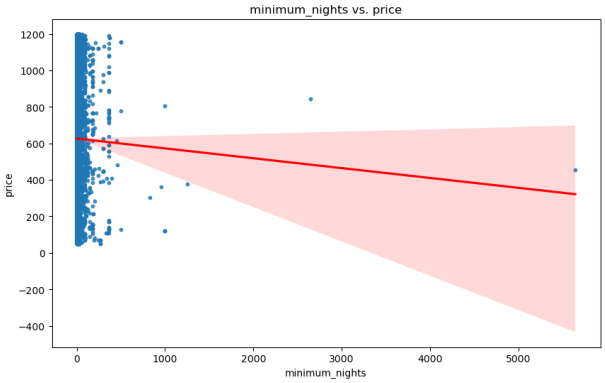


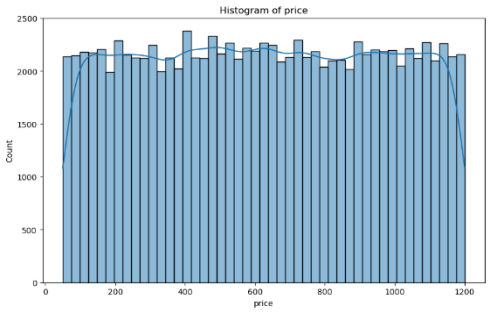
- The optimal factor to investigate is “Average price”, as the number of properties and average of nights will have limited opportunities for revenue increase opportunities for Airbnb Hosts.

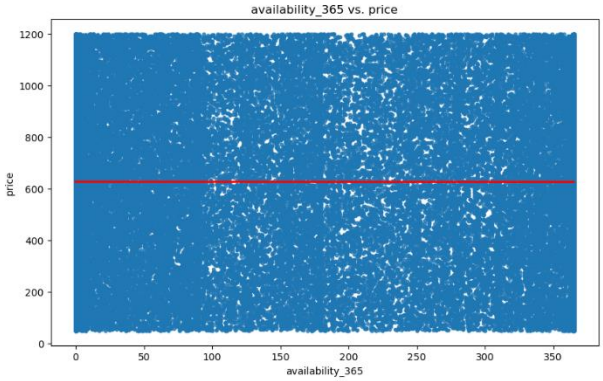
Appendix B: Relevant Charts from EDA

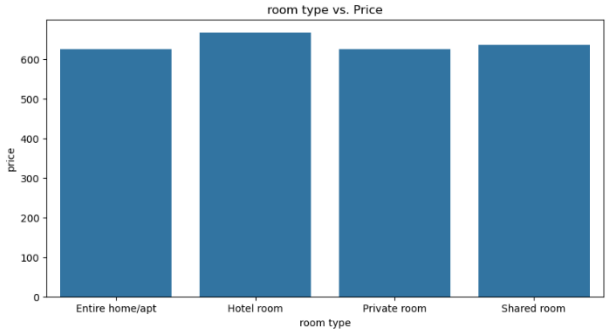












- Overall, the “price” variable, even after data cleaning is not normally distributed, as seen from the QQ plot. As an example, with some numerical variables, there is no meaningful indication from scatterplots of any linear or unique relationships vs. price. This suggests non-linearity.
- For categorical variables vs. price, each category had almost the same mean price except for “room-type” – where hotel-rooms had a higher mean price. This is generally expected (as hotels have higher prices), but also, they account for <1% of all data.

Appendix C: Parameters Grid and Classification Bins

Model	Hyperparameter	Values
KNN Regressor	knnregressor__n_neighbors	[1000, 5000, 8000]
Decision Tree Regressor	decisiontreeregressor__max_depth	[None, 10, 20, 30]
Random Forest Regressor	randomforestregressor__n_estimators	[50, 100, 200]
Random Forest Regressor	randomforestregressor__max_depth	[None, 10, 20, 30]
Gradient Boosting Regressor	gradientboostingregressor__n_estimators	[100, 200]
Gradient Boosting Regressor	gradientboostingregressor__learning_rate	[0.01, 0.1]
Gradient Boosting Regressor	gradientboostingregressor__max_depth	[3, 5, 7]
Support Vector Regressor	supportvectorregressor__C	[0.1, 1, 10]
Support Vector Regressor	supportvectorregressor__gamma	[0.001, 0.01, 0.1]
Support Vector Regressor	supportvectorregressor__epsilon	[0.1, 0.2, 0.5]
Lasso Regressor	regressor__alpha	[0.01, 0.1, 1, 10]
Ridge Regressor	regressor__alpha	[0.01, 0.1, 1, 10]
Linear Regressor	regressor__fit_intercept	[True, False]
Logistic Regression	C	np.logspace(-4, 4, 20)
Logistic Regression	penalty	['l1', 'l2']
Logistic Regression	solver	['liblinear']
KNN Classifier	n_neighbors	[1000, 5000, 8000]
KNN Classifier	weights	['uniform', 'distance']
KNN Classifier	metric	['euclidean', 'manhattan', 'minkowski']
Random Forest Classifier	n_estimators	[100, 200, 300]
Random Forest Classifier	max_depth	[10, 20, None]
Random Forest Classifier	min_samples_split	[2, 5, 10]
Random Forest Classifier	min_samples_leaf	[1, 2, 4]
Random Forest Classifier	bootstrap	[True, False]
Logistic Regression	penalty	['l1', 'l2']
Logistic Regression	C	np.logspace(-4, 4, 3)
Logistic Regression	solver	['liblinear']
Random Forest Classifier	n_estimators	[100, 200]
Random Forest Classifier	max_depth	[10, 20]
Random Forest Classifier	min_samples_split	[2, 5]
Random Forest Classifier	min_samples_leaf	[1, 2]
Random Forest Classifier	bootstrap	[True, False]
Gradient Boosting Classifier	n_estimators	[100, 200]
Gradient Boosting Classifier	learning_rate	[0.01, 0.1]
Gradient Boosting Classifier	max_depth	[3, 4]
Gradient Boosting Classifier	subsample	[0.8, 1.0]
XGBoost Classifier	n_estimators	[100, 200]

XGBoost Classifier	learning_rate	[0.01, 0.1]
XGBoost Classifier	max_depth	[3, 4]
XGBoost Classifier	subsample	[0.8, 1.0]
XGBoost Classifier	colsample_bytree	[0.8, 1.0]
LightGBM Classifier	n_estimators	[100, 200]
LightGBM Classifier	learning_rate	[0.01, 0.1]
LightGBM Classifier	max_depth	[3, 4]
LightGBM Classifier	num_leaves	[20, 30]
LightGBM Classifier	subsample	[0.8, 1.0]
KNN Classifier	n_neighbors	[10, 20, 30]
KNN Classifier	weights	['uniform', 'distance']
KNN Classifier	metric	['euclidean', 'manhattan']

Binary Classification – Class Bins:

	Count	Range (price wise)
Class 0 (low price per day property)	41,962	Under $625 USD per night
Class 1 (high price per day property)	41885	Over $625 USD per night

Multi-Class Classification – Class Bins:

	Count	Range (price wise)
Class 0 (low price per day property)	28,023	Between $50 to $437 USD per night (not including $437)
Class 1 (medium price per day property)	27,929	Between $437 to $816 USD per night (not including $816)
Class 2 (high price per day property)	27,895	Over $816 USD

Appendix D: All Model Results (Regression, Classification)

Regression Classification – Hyperparameter Tuning Included:

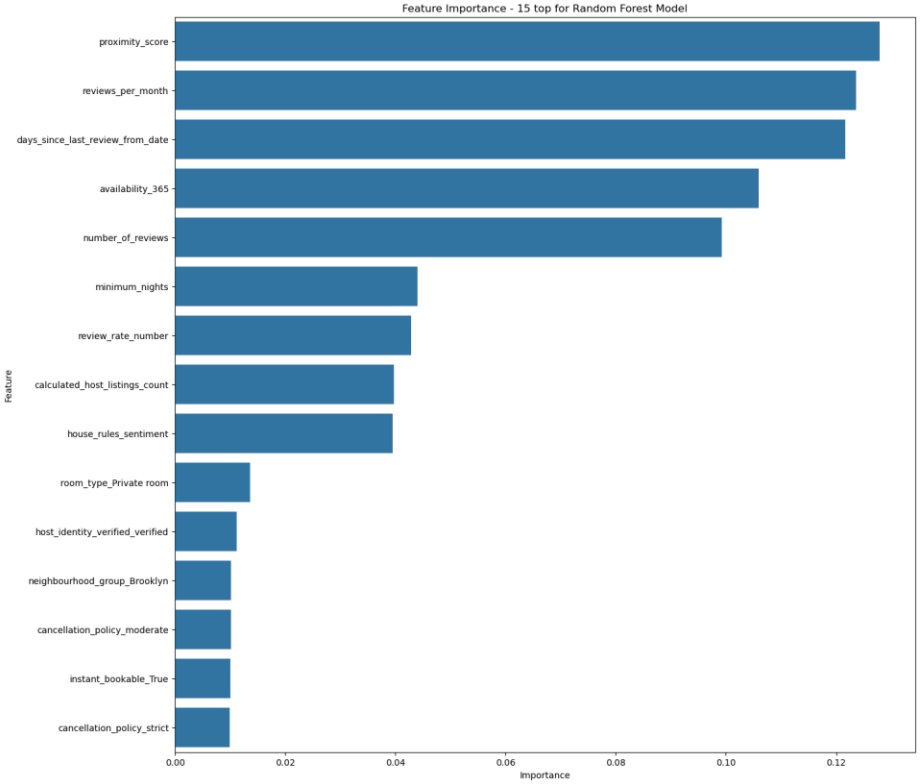
Model_Name	RMSE	Adjusted R2	CrossVal Score
KNN Regressor	329.716246	-0.004281	-110008.785631
Decision Tree Regressor	330.957689	-0.011858	-110717.250579
Random Forest Regressor	281.108772	0.269999	-73830.177492
Gradient Boosting Regressor	322.420030	0.039674	-103228.946077
Support Vector Regressor	329.696237	-0.004159	-110015.996528
Linear Regression	329.660730	-0.003943	-109997.411706
Lasso Regression	329.652263	-0.003953	-109988.452564
Ridge Regression	329.660730	-0.003891	-109995.850385

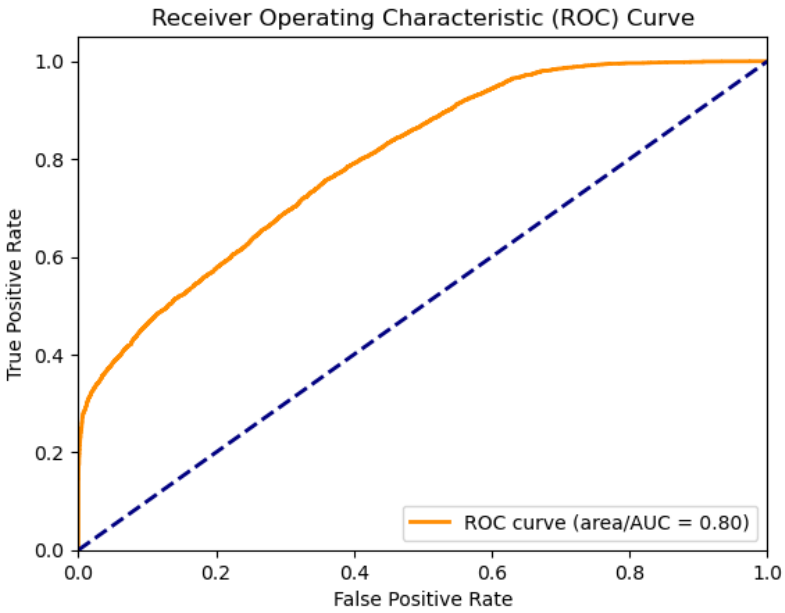
Classification Modelling:

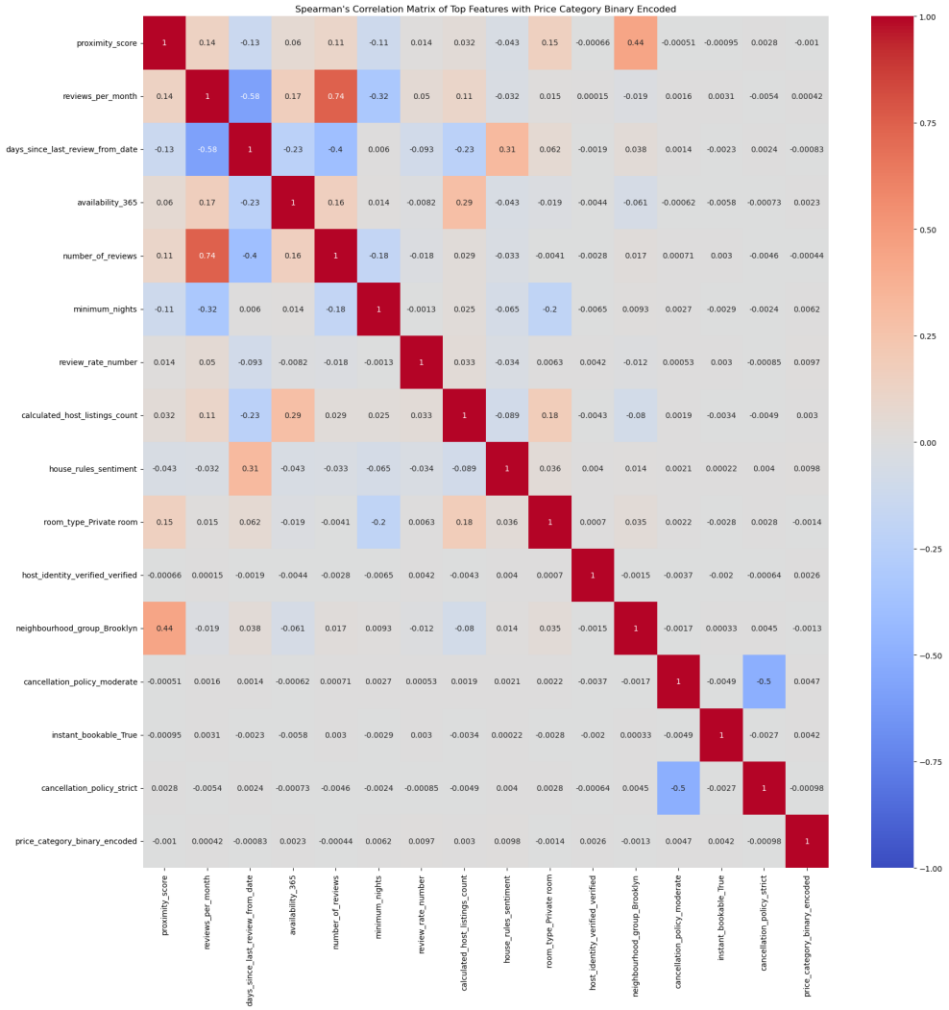
Type	Hyperparameter Tuning Applied?	Model	Precision	Recall	Mean CV Precision Score
Binary	No	LogisticRegression	0.501220	0.516770	0.502557
Binary	No	KNeighborsClassifier	0.544929	0.541926	0.544736
Binary	No	RandomForestClassifier	0.559872	0.588644	0.557698
Binary	No	GradientBoostingClassifier	0.513946	0.538572	0.517544
Binary	No	XGBClassifier	0.510073	0.533781	0.513782
Binary	No	LGBMClassifier	0.511203	0.532942	0.512424
Binary	Yes	LogisticRegression	0.501321	0.664173	0.505259
Binary	Yes	RandomForestClassifier	0.697279	0.685182	0.660636
Binary	Yes	KNeighborsClassifier	0.502639	0.939838	0.533827
Multi-class	Yes	LogisticRegression	0.34	0.34	0.3423
Multi-class	Yes	KNeighborsClassifier	0.40	0.34	0.3910
Multi-class	Yes	RandomForestClassifier	0.56	0.56	0.5270
Multi-class	Yes	GradientBoostingClassifier	0.40	0.40	0.3847
Multi-class	Yes	XGBoost Classifier	0.39	0.39	0.3785
Multi-class	Yes	Light GBM Classifier	0.38	0.38	0.3764

GREEN = best model given the focus on precision, and highest mean CV precision score.

Appendix E: Random Forest Classifier - Results, Feature Importance, Other Charts







Notes:

- The most important takeaway: the bottom row (price category binary encoded) shows a negative relationship and positive relationship, which backs up the feature importance uncovered from Random Forest Classification Model.
- **Positive Relationship** (increase of feature would contribute to classification of Class 1, which is “High price per day” property):
    - House rules sentiment, host identity verified, reviews per month – suggesting good behaviours and great host experience to guess
- **Negative Relationship** (inverse relationship of feature and Class 1 which is “High price per day” property):
    - Proximity score (the higher the score, the less likely it will be high price). The lower the proximity score, the closer that a property is to the top six NYC attractions.
    - Days since last review from date – if the last review date is old, it suggests the listing is inactive, potentially not be attractive
    - Private room type
    - Brooklyn neighbourhood group
    - Strict Cancellation policy

Works Cited

Kolomatsky, Michael. “What Happened to Airbnb during the Pandemic?” *The New York Times*, The New York Times, 15 July 2021, www.nytimes.com/2021/07/15/realestate/what-happened-to-airbnb-during-the-pandemic.html.

Lung, Natalie. “NYC Airbnb’s, Short-Term Rentals Just Got a Lot Harder to Find.” *Bloomberg.Com*, Bloomberg, 5 Sept. 2023, www.bloomberg.com/news/articles/2023-09-05/airbnb-s-new-nyc-regulations-what-renters-and-hosts-need-to-know.