OPTIMIZING AIRBNB PRICING STRATEGY IN NEW YORK CITY FOR HOSTS

Phase – 2 Report

Subject: DATA INTENSIVE COMPUTING (CSE 587B)

NOVEMBER 2023

Naga Venkata Sahithya Alla, MS
alla16@buffalo.edu

Marziye Kouroshli, MS
marziyek@buffalo.edu

University at Buffalo
Graduate School of Education

Model Selection and Analysis Report for NYC Airbnb Dataset

1. Introduction:

Our analysis delves into the NYC Airbnb dataset, aiming to uncover underlying patterns, correlations, and the pivotal factors influencing prices, occupancy rates, and customer satisfaction. We've meticulously chosen a subset of features to examine, with a primary focus on predicting Airbnb listing prices based on these selected attributes. In addition to this predictive analysis, we'll assess how different machine learning models perform on this dataset.

2. Dataset and Features:

2.1 Data Splitting:

To facilitate our analysis, we have undertaken the essential step of dividing the dataset into separate training and testing sets, utilizing an 80-20 split. This division is crucial as it enables us to rigorously evaluate the performance of our predictive models and gain insights into their effectiveness. We have considered data that is not normalized as normalizing the entire dataset before splitting can lead to information leakage from the test set to the training set and avoids target leakage. The mean and standard deviation used for normalization should be calculated only on the training set to ensure that the model generalizes well to unseen data.

3. Machine Learning Algorithms:

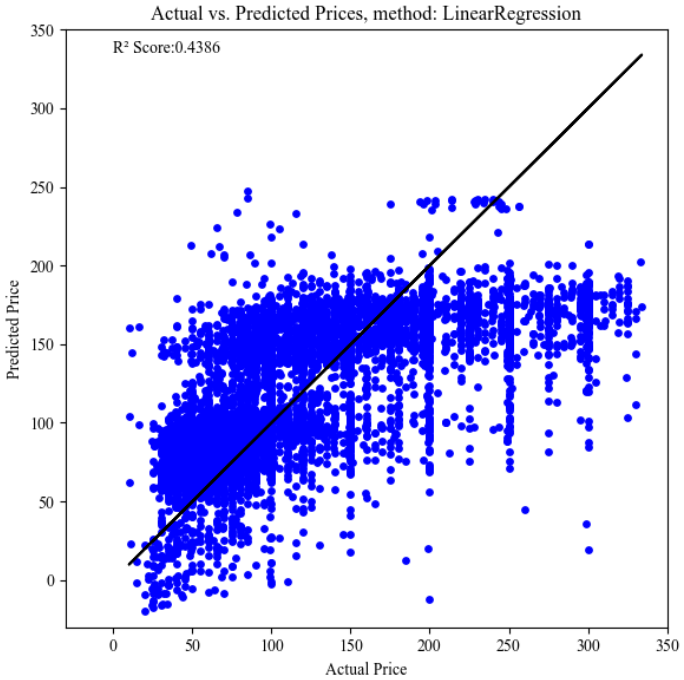
3.1 Linear Regression:

In this approach, we employed a linear regression model to forecast the price of Airbnb listings by considering all available features. Surprisingly, the performance of this linear regression model fell short of our expectations compared to the decision tree algorithm. The coefficient of determination (R-squared) for the linear regression model was

approximately ***0.44***, indicating a weaker predictive capability. Applying this algorithm on test dataset, ***mean percentage error was 17.5%, mean squared error was 2566.5*** **and** ***root mean square error was 50.6603.***

This underperformance can be attributed to the model's inherent assumption of a linear relationship between the input features and the target variable (***price***). In reality, the relationship between these features and the price may not be strictly linear. This linear assumption limited the model's capacity to capture more complex patterns and relationships in the dataset, ultimately resulting in suboptimal predictive accuracy.

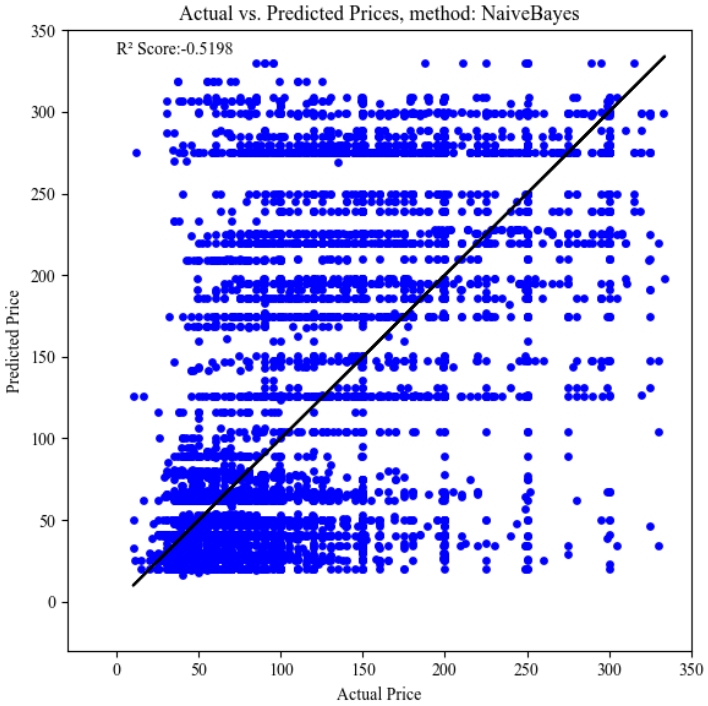
To address this challenge, we experimented with various linear regression techniques such as ***Elastic Net, Lasso, and Ridge regression***. Despite these attempts, none of these alternative methods yielded superior results compared to the **Ordinary Least Squares (OLS)** linear regression. The limitations of linear regression, which presumes linear relationships between variables, hindered its effectiveness in capturing the intricate price dynamics influenced by numerous factors in the Airbnb dataset. As a result, alternative machine learning algorithms like decision trees proved to be more suitable for this specific prediction task.



3.2 Naive Bayes classifier:

We employed the multinomial Naive Bayes classifier for our analysis, which is typically well-suited for classification tasks involving discrete features. However, in our case, where the goal is to tackle a regression problem, the suitability of this classifier is limited. As anticipated, the results fell short of expectations, with an R-squared value of approximately -0.52, highlighting the model's suboptimal performance in a regression context. Applying this algorithm on test dataset, ***mean percentage error was 20.3%, mean squared error was 6947.8*** **and root mean square error was 83.3533.**

It's important to note that the multinomial Naive Bayes classifier is designed for classification tasks, where the target variable represents discrete categories. In regression, our objective is to predict continuous values, and this classifier's inherent assumptions and characteristics may not align well with such tasks. As a result, we must explore alternative regression models to improve our predictive performance.

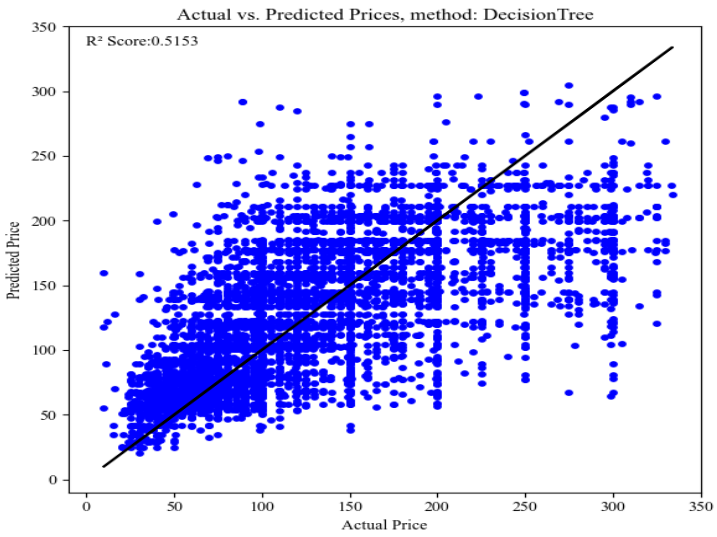


3.3 Decision Tree Regressor:

A Decision Tree Regressor predicts continuous values by recursively splitting data based on features. The algorithm selects features and thresholds to minimize variance, creating a tree structure. Predictions are made by traversing the tree from root to leaf. We examined a decision tree regression model employing the 'squared_error' criterion, which assesses the mean squared error between the actual price (true value) and the model's predicted price (estimated value). Notably, this method was chosen without the application of any pre-processing steps or feature selection for two specific reasons.

Firstly, decision trees are classified as non-parametric models, implying that they are not reliant on feature scaling. As such, normalizing the features is not deemed essential when working with decision trees. Secondly, the decision tree method inherently encompasses feature selection within its operation.

When utilizing this approach, the model demonstrates favorable performance, achieving an R-squared value of approximately ***0.52***. It is crucial to recognize that decision trees have a susceptibility to overfitting. To mitigate this, we made adjustments to two primary parameters: (1) the depth of the tree and (2) the minimum number of samples per split. These modifications were made to manage the tree's size and curtail overfitting on the training dataset. Applying this algorithm on test dataset, ***mean percentage error was 13.7%, mean squared error was 2215.0 and root mean square error was 47.1153.***

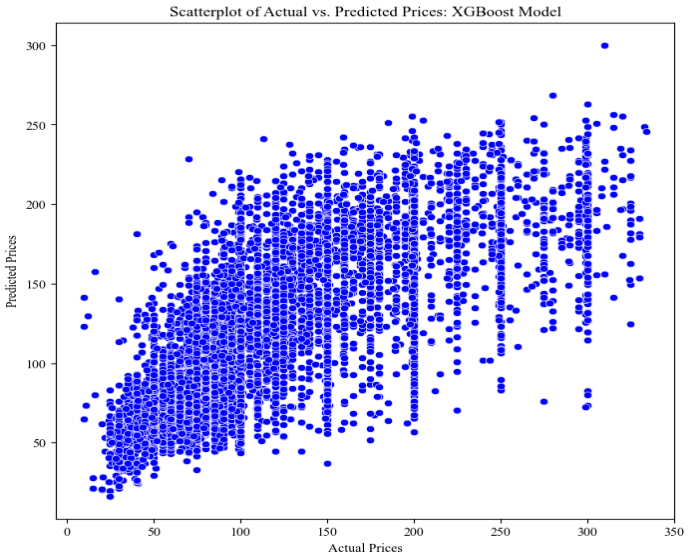


3.4 XGBoost Regressor:

In addressing the price prediction of this task, we have selected the XGBoost (Extreme Gradient Boosting) algorithm due to its proven effectiveness in regression problems, especially when dealing with complex relationships within datasets. The algorithm's capability to handle non-linear patterns, regularization features, and efficient tree pruning made it a strong candidate for capturing the intricate relationships inherent in price prediction scenarios. This is an ensemble learning algorithm designed for both classification and regression tasks.

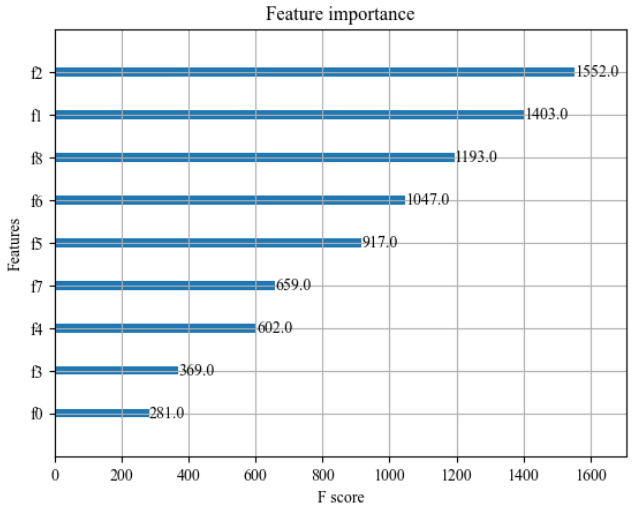
To fine-tune the XGBoost model, we experimented with different hyperparameter configurations. Specifically, we tested variations in the number of estimators (n_estimators), the maximum depth of trees (max_depth), the learning rate (learning_rate), and other parameters like 'colsample_bytree' and 'gamma'.
After several iterations, the configuration with parameters 'n_estimators=145, learning_rate=0.1, colsample_bytree=0.8, gamma=0.1' demonstrated good performance.

The model's effectiveness was evaluated using key regression metrics. The model achieved a ***Mean Percentage Error of 13.73%, a Root Mean Square Error of 44.3149, a Mean Squared Error of 1963.8137, and an R^2 of 0.5704***, Percentage Error and Root Mean Squared Error values being desirable, and the R^2 score suggesting a reasonable fit to the data. This configuration demonstrated a good balance between the model complexity and predictive accuracy for the given price prediction task.
The insights gained from feature importance analysis facilitated a better understanding of the factors influencing price predictions. This intelligence can be valuable for developers/ stakeholders, helping them to make informed decisions based on the identified significant features. Overall, the XGBoost algorithm, coupled with careful hyperparameter tuning, demonstrated its efficacy in addressing the specific challenges posed by the price prediction problem.



Feature Importance: The 'plot_importance' function in XGBoost is used to visualize the feature importance of your XGBoost model. This is a great approach to gain insights into which features are contributing the most to your model's predictions.

f1-f8 - 'neighbourhood_group','latitude','longitude','room_type', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'availability_365'. As we can see from the below plot, the location and availability of rooms features are contributing the most.



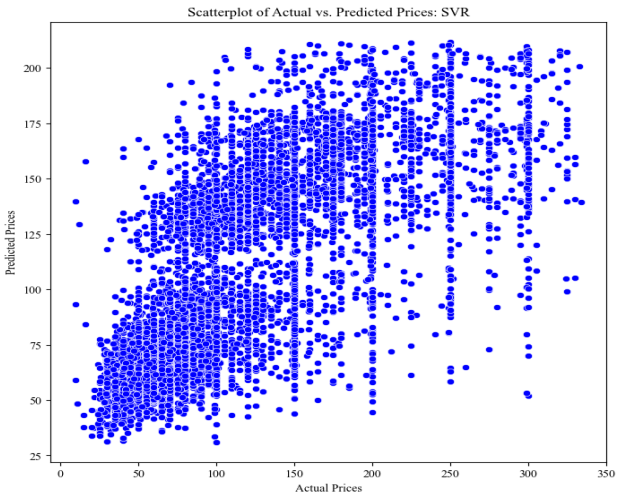
3.5 Support Vector Regressor:

The Support Vector Regression (SVR) model was chosen for price prediction due to its flexibility with various kernels, including the capacity to capture non-linear relationships, robustness to outliers, and effectiveness in high-dimensional spaces.

Among the SVR configurations tested, the model with radial basis function (RBF) kernel, C=1.0 and degree=3 performed optimally than the one we tested with linear kernel. SVR with RBF kernel is effective in capturing non-linear relationships in the data, making it suitable for complex regression tasks.

To fine-tune the SVR model, we tested different kernel and other parameter values like different kernels, different regularization parameter (C value), epsilon value, and kernel coefficients. But the model with RBF kernel, C=1.0 and degree=3 performed well than others.

The SVR model, with the specified configuration, has produced a ***Mean Percentage Error of 6.30%, a Root Mean Square Error of 48.496, a Mean Squared Error of 2351.8651, and an R^2 of 0.4855***. These metrics provided a high-level insight into the model's performance. The low Mean Percentage Error indicates that, on average the predictions are relatively close to the actual values. The Root Mean Squared Error and Mean Squared Error signify the accuracy of predictions, with lower values being desirable. The R^2 score of 0.4855 indicates the proportion of variance in the target variable explained by the model, with a higher score suggesting a better fit to the data.



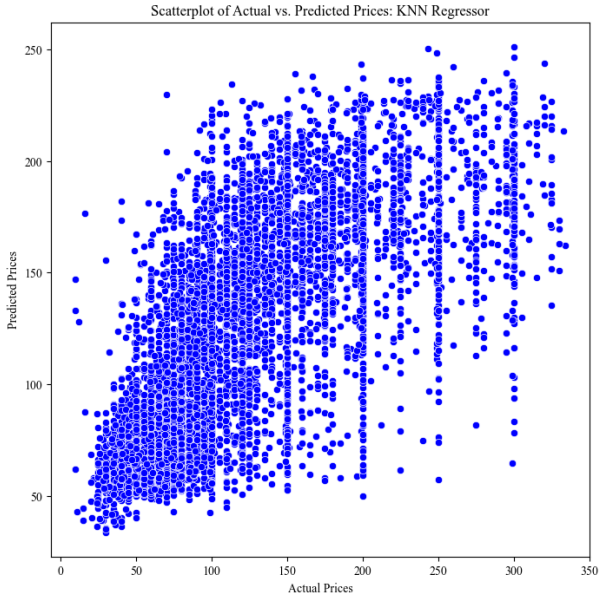
3.6 K-Nearest Neighbor Regressor:

We chose K-Nearest Neighbor Regressor (KNNR) model for this task, employing a value k=30. We experimented with various k values derived from cross validation results, the square root number of samples, and randomly chosen values. Interestingly, our randomly assumed k values, guided by data analysis, demonstrated slightly better performance compared to other approaches. KNNR is a non-parametric, instance-based learning algorithm that does not assume a specific functional form for the relationship between features and the target variable. We chose this model because of its simplicity, flexibility, and ease of implementation. It is particularly suitable when the underlying relationships between the data are complex and cannot be easily captured by parametric models. It does not make strong assumptions about the data distribution, making it versatile for various types of problems.

The hyperparameter k, representing the number of neighbors to consider, was set to 30. The choice of k is crucial as too low values may lead to overfitting, while too high values may result in over smoothing. The selection of 30 neighbors represents a balance, aiming to capture the local structure of the data without being overly sensitive to noise.

The model has produced a ***Mean Percentage Error of 15.891%, a Root Mean Square Error of 46.4826, a Mean Squared Error of 2160.6313, and an R^2 of 0.5274***.

The KNNR model, while providing simplicity and flexibility, seems to have a higher Mean Percentage Error compared to the SVR model and XGBoost. This suggests that, on average, the SVR model might be providing more accurate predictions for our dataset when compared to this model. This poor performance might be because of its sensitivity to outliers and noise, potential issues with the curse of dimensionality, the impact of choosing an appropriate k value. In our task, where relationships are a little complex and non-linear, other models could be more effective. Also, the scale of features and the inherent characteristics of the dataset might not align with the KNNR's strengths, contributing to less accurate predictions.



More algorithms:

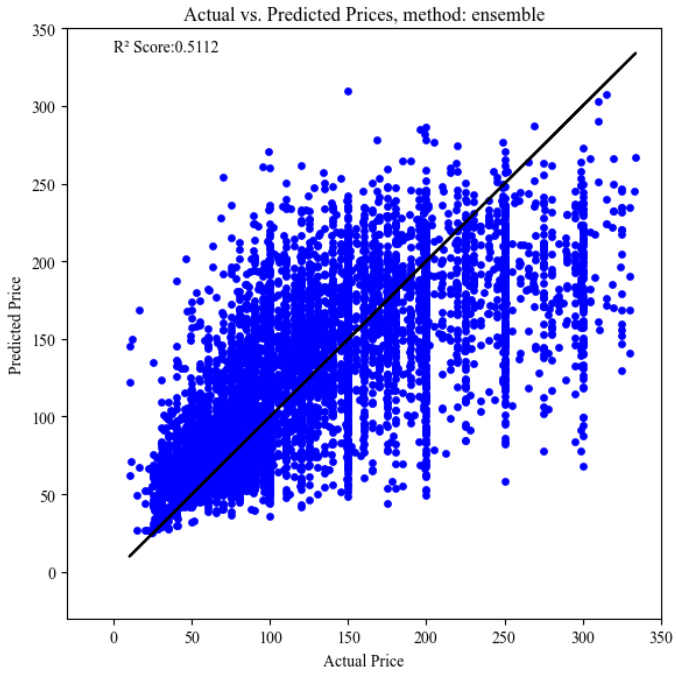
3.7 Ensemble Random Forest:

Similar to the Decision Tree algorithm, we refrained from implementing feature selection or normalization when using the Random Forest method. The reason behind this approach was to maintain model simplicity and mitigate the risk of overfitting. By employing a relatively conservative setting with only 10 estimators and a limited tree depth of 20 layers, we aimed to strike a balance between model complexity and predictive performance.

However, despite these measures, the results, while improved compared to linear regression, still fell short of the performance achieved with the Decision Tree algorithm. The coefficient of determination (R-squared) remained at approximately ***0.51***. This disparity in performance can be attributed to the fundamental nature of the Random Forest algorithm.

Random Forest is a more complex ensemble learning method that combines multiple decision trees to make predictions. Although we limited the depth and number of estimators, it still exhibits a higher degree of complexity compared to individual decision trees. This added complexity allows Random Forest to capture more intricate relationships

within the data. Still, the method's limitations, such as potential overfitting or suboptimal model parameters, could have contributed to its performance not surpassing that of the Decision Tree algorithm. Applying this algorithm on test dataset, ***mean percentage error was 14.7%, mean squared error was 2234.59 and root mean square error was 47.2715.***



3.8 Neural Network

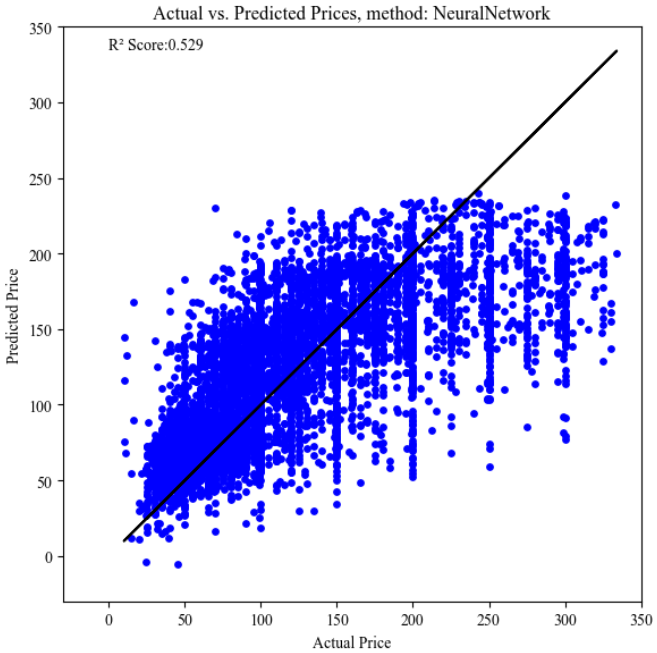
In this approach, we embarked on a comprehensive hyperparameter tuning process to fine-tune our neural network model for predicting Airbnb listing prices based on various numerical features. The aim was to leverage the full potential of the neural network by optimizing hyperparameters to achieve superior predictive performance. Before diving into model architecture and hyperparameters, we first applied feature scaling to normalize the input data. We didn’t apply feature selection and add L2 regularization in the fitting procedure.

For the neural network architecture, we explored multiple hyperparameters, including the number of units in hidden layers, the number of hidden layers, activation functions,

learning rates, learning rate schedulers, batch sizes, and solver methods. After extensive experimentation, we converged upon the following configuration:

- Default optimizer: ADAM
- one hidden layer with 100 units
- ReLU activation function
- Initial learning rate of 0.001, with an exponential decay schedule of 0.5
- Batch size of 16
- Maximum iterations set to 500 with early stopping

The results of this neural network model surpassed those of the previously mentioned algorithms, including Decision Trees, Linear Regression, and Random Forest. We achieved a notably improved fitting score, with an R-squared value of **0.52**, signifying the model's enhanced predictive accuracy. This optimized neural network configuration, along with data normalization, has proven to be a robust solution for predicting Airbnb listing prices. Applying this algorithm on test dataset, ***mean percentage error was 15.23%, mean squared error was 2153.05 and root mean square error was 46.4011.***



Summary:

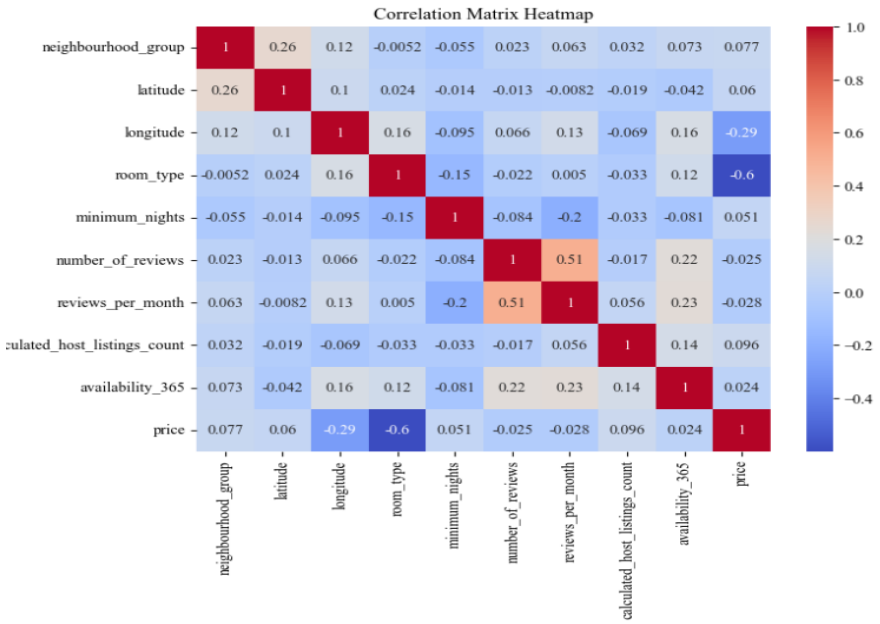
Algorithm	R-squared	Mean Percentage Error	Mean Squared Error	Root Mean Squared Error
Linear Regression	0.4386	17.5%	2256.5	50.66
Naïve Bayes	-0.5198	20.33%	6947.8	83.35
Decision Tree	0.5155	13.7%	2215.0	47.11
XGBoost	0.5704	13.7%	1963.8	44.31
Support Vector Regressor	0.4855	6.3%	2351.8	48.49
K-Nearest Neighbor Regressor	0.5274	15.8%	2160.6	46.48
Random Forest	0.5112	14.7%	2234.5	47.27
Neural Networks	0.529	15.23%	2153.0	46.40

Algorithms used that are taught in class: Linear Regression, Naïve Bayes, SVM, KNN

Other Algorithms used: Decision Tree, XGBoost, Random Forest, Neural Networks

Conclusion:

From the presented models and their outcomes, it is evident that XGBoost outperforms the others. While XGBoost demonstrates good performance, the potential for even better results exists with a more robust dataset. Given that regression tasks performance evaluation cannot be relied on accuracy metric, our focus primarily lied on improving the Root Mean Squared Error (RMSE) and R^2 values. This improvement can be achieved by augmenting the dataset significantly and incorporating business insights to better understand the factors impacting prices. For better performance, we have also tried to preprocess data in various ways to see if it will perform better, changed the features that we considered for training but the results did not seem to change much and got even worse in few cases. The correlation matrix for the current dataset reveals weak correlations between the features and the target variable (as illustrated below), indicating potential issues with the existing dataset.



To optimize the pricing to help hosts and guests, we can show the hosts a table of prices in the same neighbourhood area offered by different hosts along with our model's price prediction. While the prices also depend on many factors like amenities in the Airbnb, location of property, property size, and unique features offered by the hosts, they can decide based on the special services they are offering along with considering our recommendations and set the price accordingly.

References:

1. https://scikit-learn.org/stable/
2. https://matplotlib.org/stable/
3. https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article
4. https://www.datacamp.com/tutorial/xgboost-in-python
5. https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b

Contribution Form

Group member 1: Naga Venkata Sahithya Alla

Group member 2: Marziye Kouroshli

(5 – being highest and 1- being lowest)

Evaluation Criteria	Group member 1	Group member 2
How effectively did your group mate work with you?	**5**	**5**
Contribution in writing the report	**5**	**5**
Demonstrates a cooperative and supportive attitude.	**5**	**5**
Contributes significantly to the success of the project.	**5**	**5**
TOTAL	**5**	**5**

Overall Contribution:

Group member 1: 50%

Group member 2: 50%