

DATA INTENSIVE COMPUTING

COMPLETE REPORT PHASE 1-3

OPTIMIZING AIRBNB RENTAL STRATEGIES
IN
NEW YORK CITY

HEMANTH SHRINIVASAN BASAVARAJ – HBASAVAR

GOWTHAM SIVARAMAN – GSIVARAM

VISHAL SRINIVAS RAMESH – VISHALSR

1. **Problem Statement:**

With websites like Airbnb providing lodging options outside of traditional hotels, the sharing economy has completely changed the way we travel. In a city as dynamic as New York City, where there are many different areas and a constantly shifting market, Airbnb hosts and potential investors want to get the most out of their investment while giving their guests an excellent experience. In order to support present and potential Airbnb hosts in making strategic decisions, the challenge is to pinpoint the important variables that affect rental rates, demand and comprehend what motivates customers contentment, and forecast market trends. This will allow an aspiring Airbnb host to ensure that his listing is equipped with those important features such that he will be able to charge a higher price without losing customers. Moreover, a traveler will also know the factors to look into to get the lowest price possible while having certain features he prefers.

Background: Especially in large cities like New York City, Airbnb's explosive development has had a tremendous impact on the tourism sector. Given the fiercer competition among hosts, it is critical to comprehend the many factors that go into making a listing successful. The occupancy rate and profitability of a property are significantly influenced by variables like location, cost, kind of lodging, host reputation, and season.

Significance of the Issue: There are several reasons why it's important to be able to assess and forecast market trends in the Airbnb space. In order to sustain high occupancy rates, hosts must offer amenities that appeal to guests and set competitive prices. Better lodging experiences are the result of listings' transparency and optimization, which benefits guests. Local companies and service providers might modify their services to accommodate the increasing number of Airbnb visitors in the area. Policymakers and city planners are able to comprehend how short-term rentals affect the local economy and housing market.

Project Aims:

- Give hosts practical advice on pricing tactics so they can maximize their profits.
- Provide a pricing and demand prediction model so that hosts may choose their listings wisely.
- Recognize trends in guest satisfaction to help hosts provide better service.
- Examine how time and location affect rental demand to help local companies and hosts plan strategically.
- Make a contribution to the wider comprehension of the dynamics of the sharing economy, especially in urban environments.

The project helps in enhancing the economic benefits for the hosts by maintaining affordability and accessibility for the guest, also aids the making informed policy decisions around rentals on housing.

2. **Data Source:**

The dataset is available on Kaggle at the following link.

<https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data/data>

3. Data Cleaning / Processing:

Some preprocessing techniques and cleaning of data to the dataset were done in order to obtain the accurate findings. Data cleaning process involves the correction or removal of inaccurate, improperly formatted, duplicated or incomplete data present in the dataset.

The data cleaning steps are as follows:

3.1 Convert 'last_review' to datetime format from string format – last_review column contains

the date for the reviews. The datatype is in object (string) format and it has been converted to the datetime format.

3.2 Filling missing 'name' and 'host_name' with "Unknown" –

The attributes name and host_name had some missing values. Those missing values were replaced with 'Unknown'.

3.3 Remove listings with unrealistic minimum nights –

Since the minimum nights could not exceed 365 days, so the values that has the minimum nights greater than 365 has been removed.

3.4 Remove outliers for 'price' –

Interquartile method is used to find the potential outliers in the 'price' column of the dataset. The values of first quartile and third quartile were estimated with IQR. A multiplier of 1.5 times the IQR, lower and upper bounds were found to identify

potential outliers. Any data points that falls outside this range were considered outliers and were removed from the dataset.

3.5 Fill missing 'reviews_per_month' with 0 –

Assuming no reviews as 0 reviews per month the attribute 'reviews_per_month' is filled with the value of 0.

3.6 Remove listings with a price of \$0 –

As the price for renting the room can never be zero, the values of zeroes are removed.

3.7 Remove rows where 'last_review' is NaT (Not at Time) after conversion –

Rows with missing or incorrect datetime values (NaT) in the 'last_review' column were eliminated from the dataset following conversion in order to guarantee the accuracy and completeness of the data.

3.8 Converted 'id' and 'host_id' to strings –

Since these attributes do not need numerical operations, they are transformed to strings in order to maintain consistency across data types.

3.9 Reset the index –

To reorganize the data frame for subsequent analysis, a common preprocessing step – resetting the index is performed. It reindexes the df, drops the index and replace it with a new index.

3.10 Normalize 'latitude' and 'longitude' –

To normalize the numerical features 'latitude' and 'longitude' in the DataFrame, a scaling technique called Min-Max scaling technique is used. It ensures to scale the values between 0 and 1.

3.11 Feature Engineering –

To extract additional information for the analysis of the dataset, a new feature called 'price_per_night_per_min_night' is introduced. The value is calculated by divided by 'price' column by the 'minimum_nights' column.

3.12 Extract Year and Month from 'last_review' –

Two new columns, 'review_year' and 'review_month', were created using the 'dt.year' and 'dt.month' attributes respectively. This enables segmentation of the review data by year and month, allowing for deeper insights of review patterns and trends over different time periods.

3.13 Standardize 'number_of_reviews'

3.14 preprocess textual data –

A function is defined to perform some steps like checking for NaN values, removing punctuations, removing extra spaces, ensure text starts with an alphabet or a a number, lowercasing the data, stripping tailing spaces.

3.15 categorical values replaced with numerical values –

The categorical values for the attributes 'neighbourhood_group' and 'room_type' are assigned to numerical values by using map method.

3.16 Removing emojis –

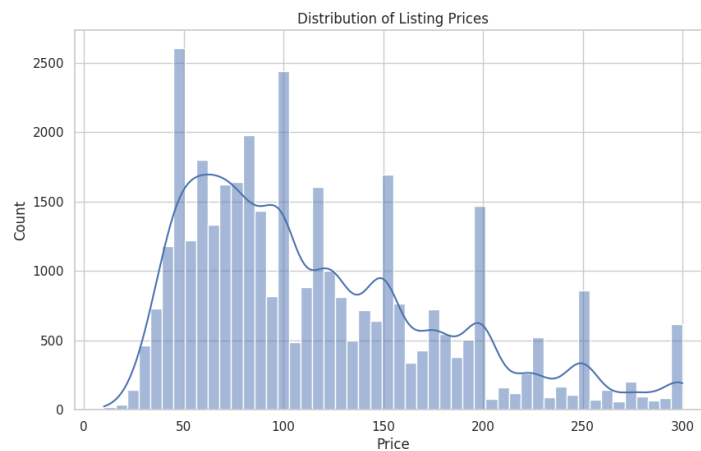
This cleaning process eliminates emojis from textual data using regular expressions.

4. Exploratory Data Analysis:

EDA helps in identifying the trends or general patterns in the dataset. EDA helps to understand the data, discover patterns, spot anomalies, and formulate hypotheses.

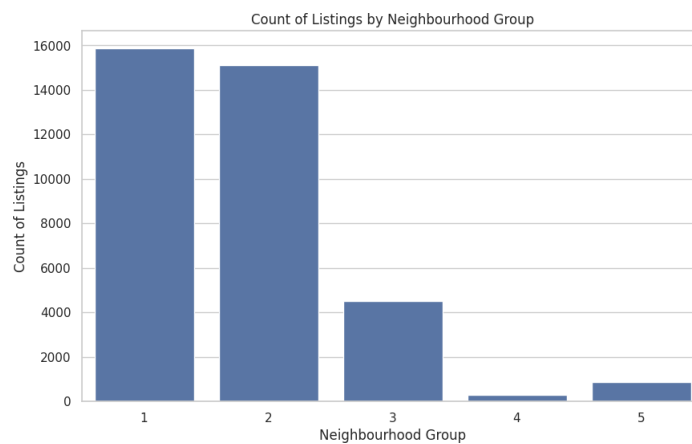
The set of statistical measures is calculated using the describe() method. It displays the values of mean, standard deviation, min and max values.

4.1 Distribution of prices:



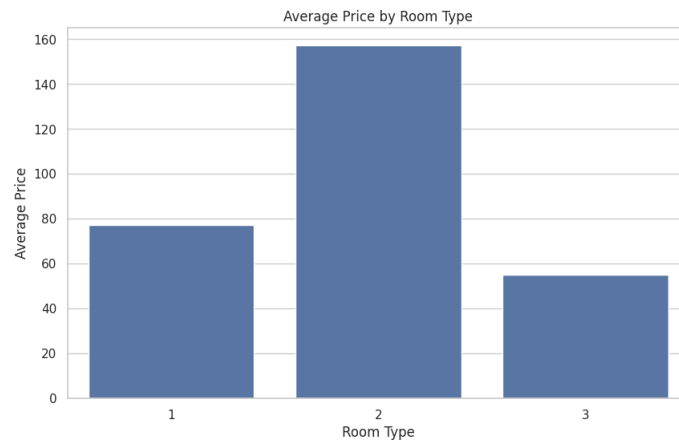
Created a histogram plot to visualize the distribution of listing prices, with Price in X-axis and Count in Y-axis. Most of the hosts booked the listings with prices in the range of 50 to 150, whereas the listings having the prices of more than 250 are booked the least.

4.2 Count of Listings by Neighborhood Group:



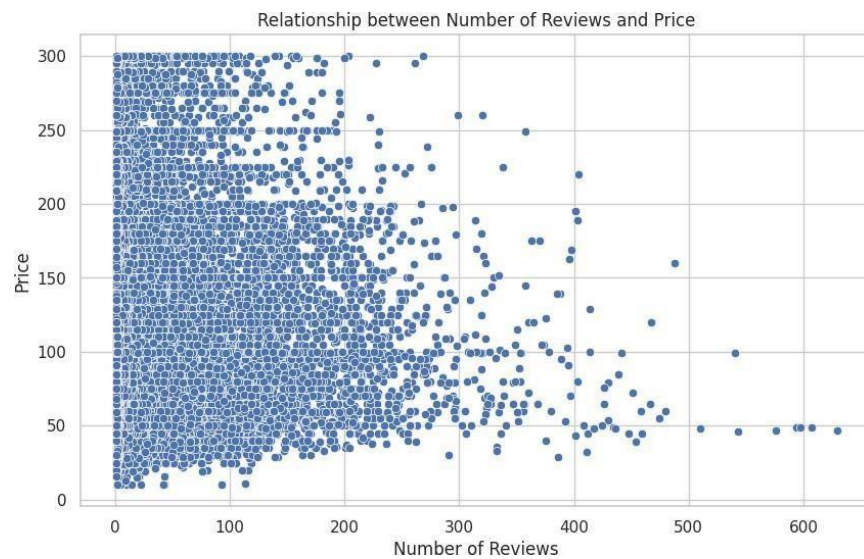
Created a count plot to visualize the count of listings by Neighborhood group. From the graph, it is seen that most of the hosts listed belongs to the 'Brooklyn' neighborhood, while the least listed are from 'Staten Island' Neighborhood.

4.3 Average Price by Room Type:



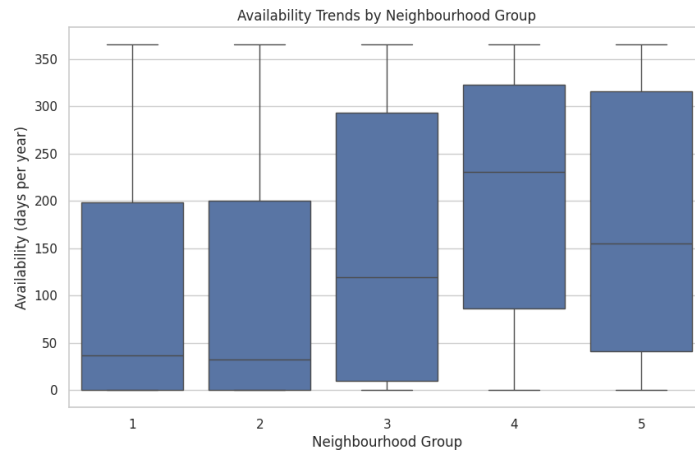
Average prices of 'Entire home/apt' is close to \$160 (highest) and 'shared room' is around to \$55 (lowest).

4.4 Relation between Number of reviews and Price:



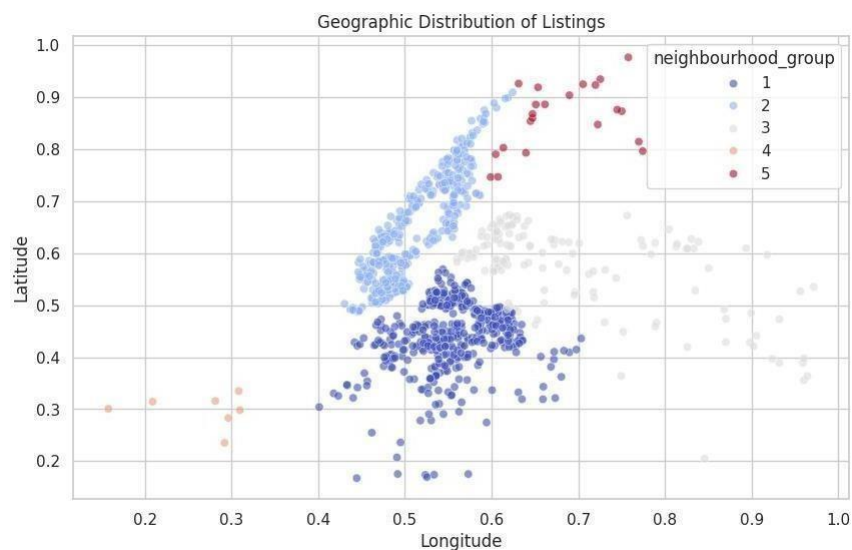
A scatterplot is plotted between Number of reviews on x-axis and price on y-axis. It can be seen that most of the hosts gave their reviews for the listings with the prices up to \$200 and as the price increases above that, the number of reviews decreases.

4.5 Availability Trends by Neighborhood Group:



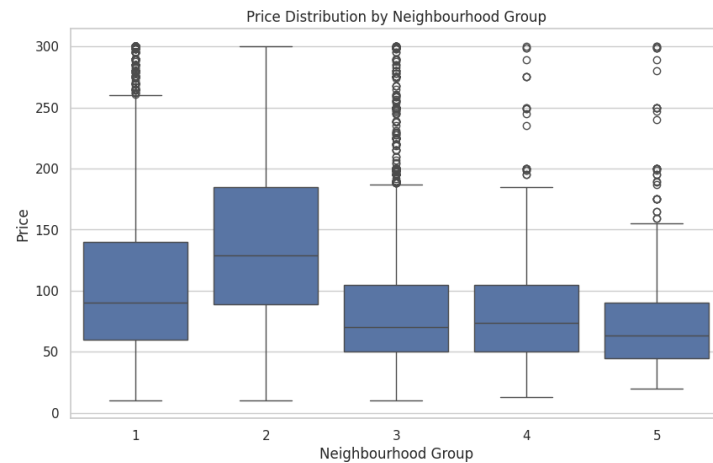
Boxplots of Group 1 and 2 are similar which shows that Brooklyn and Manhattan neighborhood have less availability. Since group4 has higher median and 75th percentile value, Staten Island neighborhood is more available when compared to others.

4.6 Geographic Heatmap of Listings:



A visualization of geographic distribution of listings based on the neighborhood groups is created. It is seen that Brooklyn group is compact/closer. Manhattan and Queens group are larger in size and are widely distributed, whereas the listings of Staten Island less.

4.7 Price Distribution by Neighborhood Group:



From the boxplot of the price distribution graph, the Manhattan group listings are expensive and Bronx group listings have less price.

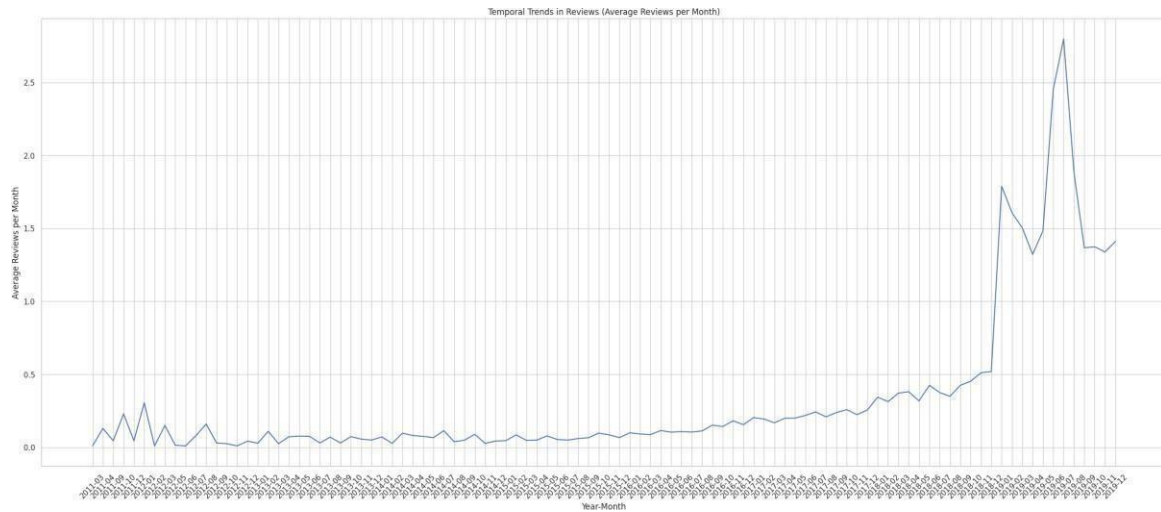
4.8 Impact of Minimum Nights on Price:



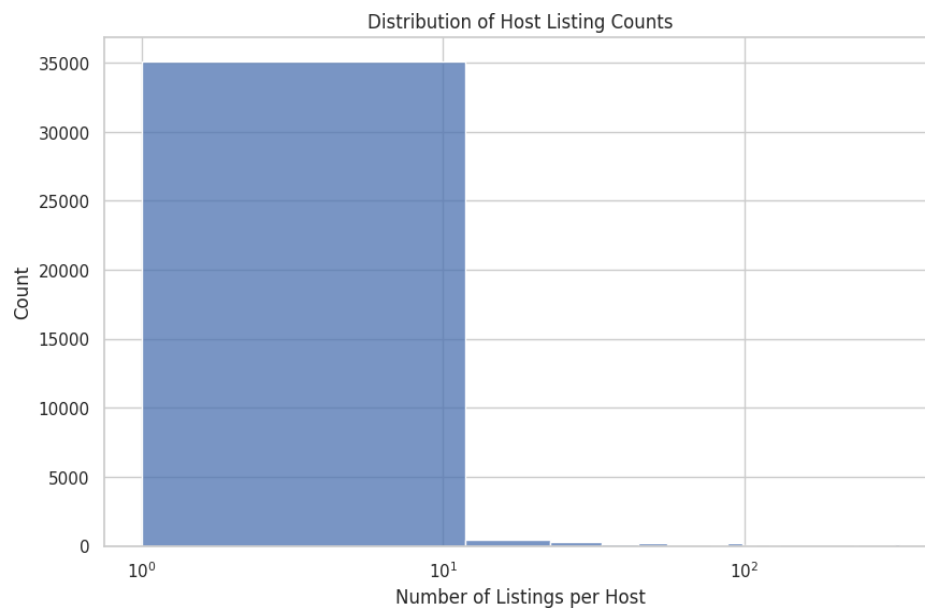
This scatter plot provides details how the price of listings is influenced by the minimum nights with each point representing a listing and its corresponding minimum nights and price values.

4.9 Temporal Trends in Reviews:

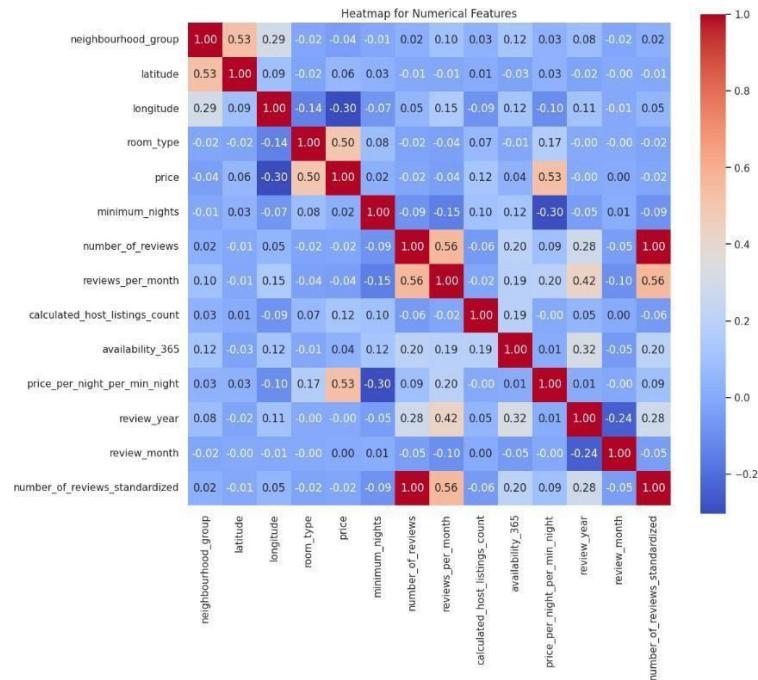
Average number of reviews oscillates between 0.2 to 0.3 for the years from 2011 to 2016 and it increases gradually from 2017 and reaches maximum number of reviews in 2019.



4.10 Host Listing Count Distribution:

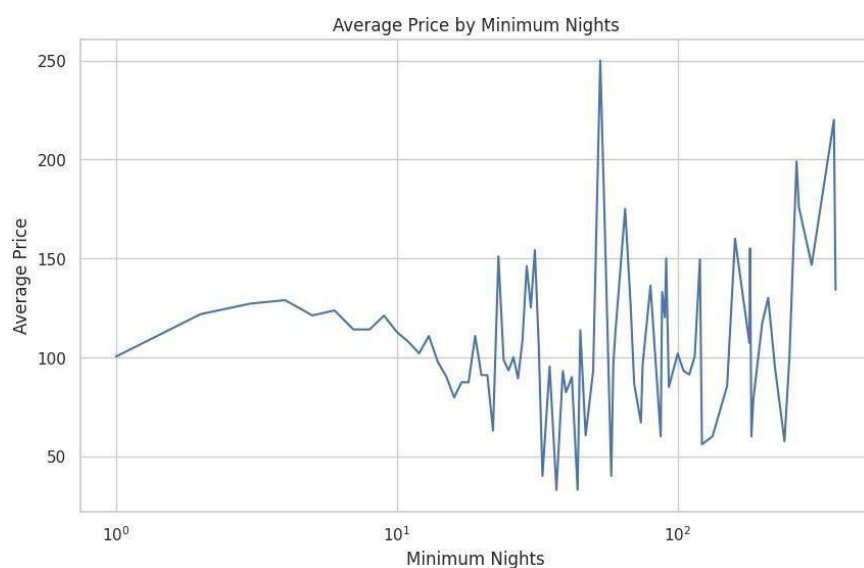


4.11 Heat map of the numerical features of the dataset:



Review year and Review month have very less correlation. Number of reviews and reviews per month are highly correlated.

4.12 Relation between Length of Stay vs. Price:



The plot basically illustrates how the price varies based on price for minimum nights. This helps to visualize the trends and patterns of price based on the days of night stay in an effective manner.

PHASE-2

DATA PREPROCESSING:

Feature Selection: The dataset has many features related to Airbnb listings. Features such as latitude, longitude, neighborhood group, room type, minimum nights, number of reviews, reviews per month, calculated host listings count and availability throughout the year were selected for analysis.

Here, we set the feature “Price” as a target variable (dependent variable) and the remaining features as independent variable.

Feature Scaling: To ensure that all the features contribute equally to the model fitting process, numerical features are standardized using the StandardScaler. It does not allow the features with larger scales to have high effect on the model training process.

Data Splitting: The dataset is split into training and testing sets using the `train_test_split` function from the `sklearn.model_selection` module. The dataset is divided into 80% for training and 20% for testing.

LINEAR REGRESSION:

In general, Linear Regression describes the relationship between the independent and dependent variables. In the context of Airbnb rental strategies, we can assume that the features such as neighbourhood group, minimum nights, location (latitude, longitude), type of listing (room type) have a linear impact on price.

It provides coefficients for each relationship between features and the price, so that it is easier to interpret the influence of each feature on the predicted price. It is easy to implement and computationally efficient.

The effectiveness of the Linear Regression model was assessed using two key metrics:

Root Mean Square Error (RMSE) – measures the average deviation of the predicted prices from the actual price. A lower RMSE indicates better model performance.

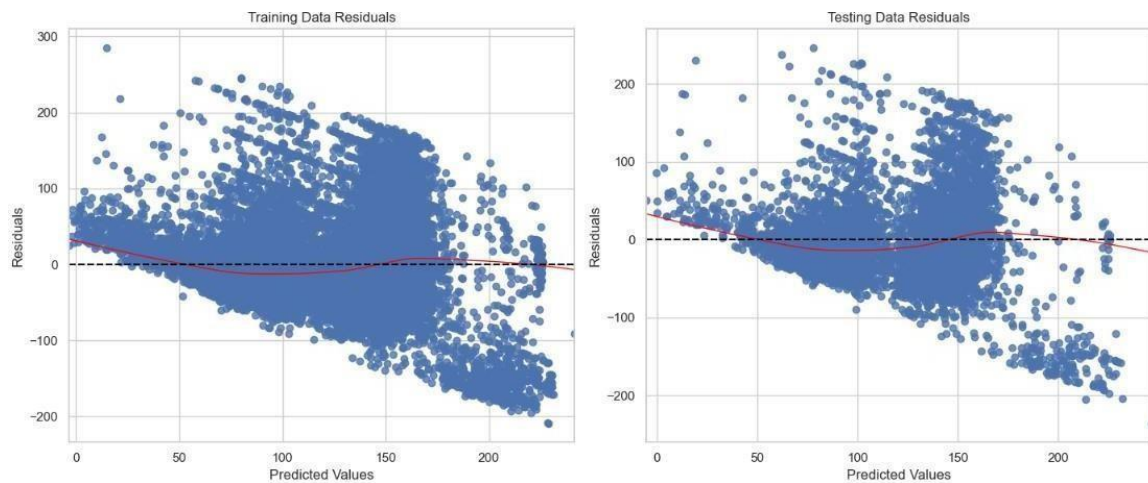
R-squared (R²) score – gives the proportion of variance in the listing prices. A higher R² score indicates better fit of the model to the data.

```
linear regression RMSE:55.5675689819255  
linear regression R2:0.31569962456377365
```

The metrics values provided by the model are-

RMSE: 55.57. It suggests that the model's predictions have a considerable margin of error, which could be further improved.

R²: 0.315. Though this explains the variance moderately, it can be improved to capture more variance in the data.



The residual plot is plotted. It represents the difference between actual value and predicted values produced by the linear regression model.

K NEAREST NEIGHBORS:

KNN algorithm is capable to capture nonlinear relationships between the attributes. If there is any relationship between features and price is nonlinear, KNN can provide accurate predictions. As we have localized patterns on the dataset i.e nearby data points have similar price values, applying the KNN model could be beneficial.

Since KNN algorithm works based on the distance metrics. Tuning is performed before training the model.

GridSearchCV is used for hyperparameter tuning to identify the optimal configuration of the K-NN algorithm.

n_neighbors: The number of nearest neighbors. A range of 1 to 5 was given. **Weights:** (Uniform, Distance), it weights neighbors equally (uniform) or by the inverse of their distance (distance), giving closer neighbors.

metric: The distance metric used to measure similarity, with Euclidean and Manhattan metrics considered to accommodate different aspects of spatial and attribute-based proximity.

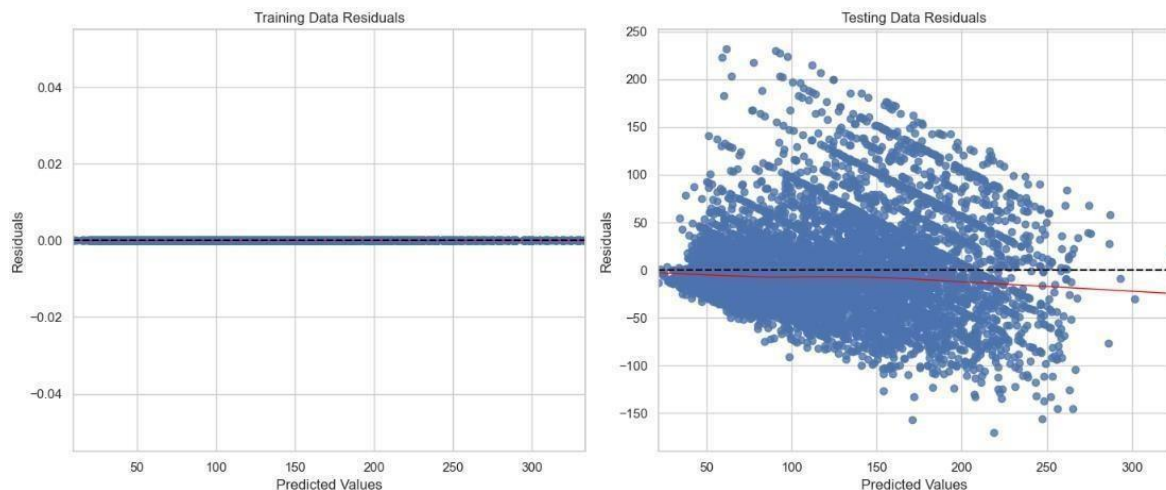
GridSearchCV performs the fitting across the specified parameter values, using 5-fold Cross Validation to find the model's best combination ensuring performance.

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best parameters found: {'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}
K-NN RMSE: 46.86631241605377
K-NN R2: 0.5132280433996699
```

The model produces RMSE value of 46.86 suggesting that model prediction is off by 46.86 from the actual value. The R2 score of 0.51 indicates that the model explains around 51% of the variance in the target variable which is moderate.

It can be observed that KNN performed better well than the linear regression model as the RMSE value is reduced and R2 score is increased. However, if the dataset has high dimensionality or if there

are non-linear relationships that KNN struggles to capture effectively. Further experimentation with different algorithms and feature engineering techniques may be necessary to improve model performance.



The residual plot is plotted. It represents the difference between actual value and predicted values produced by the linear regression model.

GAUSSIAN NB CLASSIFIER:

Gaussian NB model is used here for classification, predicting whether the price of a property is above or below the median price. It assumes that the features follow a Gaussian distribution, which is suitable for numeric data like property prices.

Training/Tuning: The median price is used as a threshold to classify properties as above or below the median. This preprocessing step simplifies the problem into a binary classification task. The dataset is split into training and testing sets to evaluate the model's performance on unseen data.

The model is fitted on the training data, where it is trained on the patterns in the features and their relationship with the target variable.

For the Hyperparameter Optimization, the parameters such as `var_smoothing` – adds a smoothing factor to variance, values from 1 to $1e-9$ was given for tuning the model's sensitivity. It fitted multiple models across the hyperparameter space using 5- fold CV to identify the parameters that gives the best accuracy.

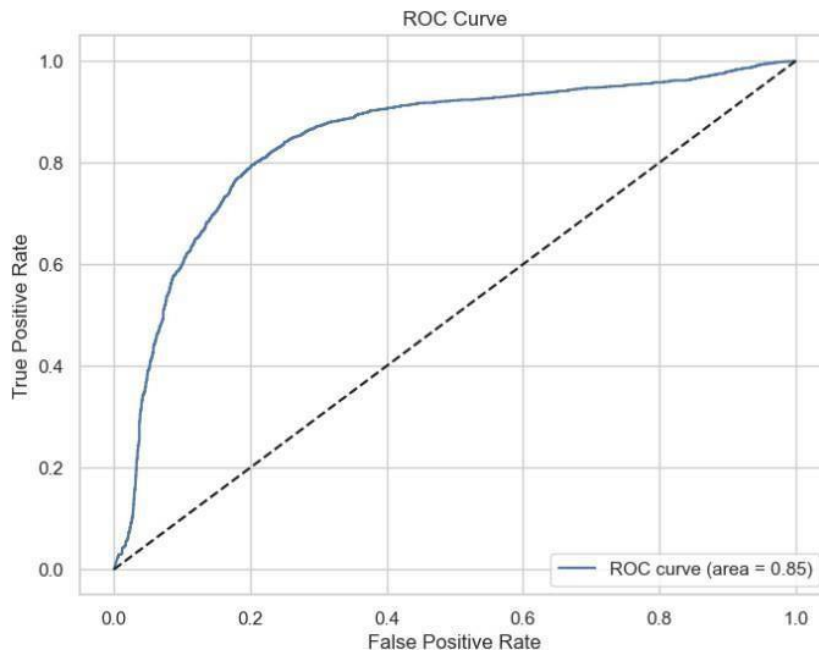
The trained model is evaluated using accuracy and F1 score metrics on the test set. The accuracy score measures the proportion of correctly classified instances out of the total instances. The F1 score is a harmonic mean of precision and recall. It gives a measure between the two, where precision represents the proportion of correctly predicted positive cases out of all predicted positive cases, and recall represents the proportion of correctly predicted positive cases out of all actual positive cases.

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best parameters found: {'var_smoothing': 0.001}
Naive Bayes Accuracy: 0.731941997560645
Naive Bayes F1: 0.6517605633802817
```

Accuracy of 73.19% is achieved, that means 73.1% of the properties were correctly classified as being above or below the median price. The F1 score of approximately 0.651 shows a reasonable balance between precision and recall for the model.

The model is reasonably good at both minimizing false positives and false negatives.

It's important to analyze misclassified instances to understand where the model struggles the most. Examining features associated with misclassifications can provide insights into areas for improvement, such as feature selection or data preprocessing. It is important to check the misclassification of the model and to analyse misclassified parts to understand the model to understand the problem.



Based on this ROC curve obtained, the gaussianNB model performs well.

Logistic regression:

Logistic regression model is used for binary classification tasks, where the target variable has two classes. In this case, we are predicting whether the price of a property is above or below the median. It provides coefficients for each feature, allowing for interpretation of the impact of each feature on the probability of the target class.

An instance of logistic regression is created, with a parameter `max_iter` set to 1000 to ensure convergence.

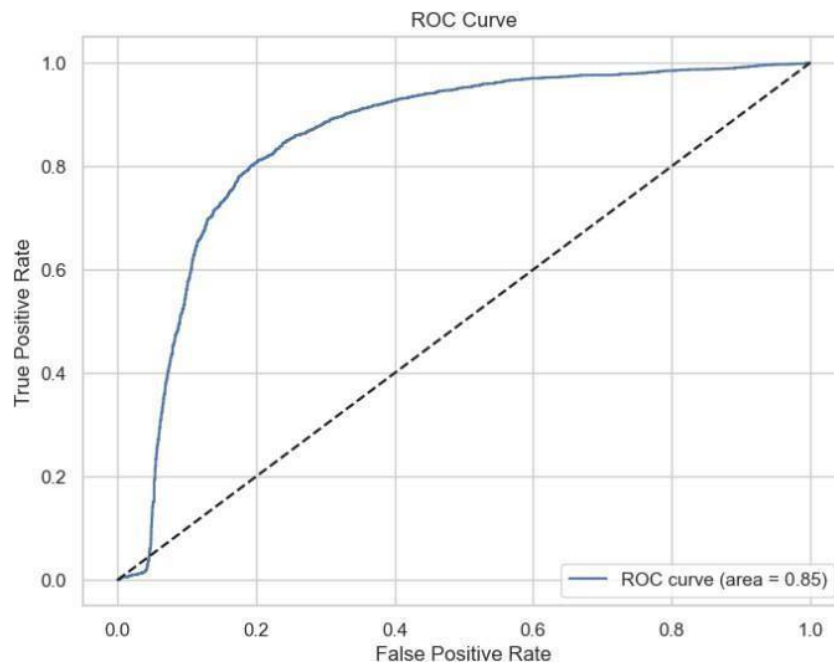
A comprehensive grid search over key hyperparameters (`C` for regularization strength, penalty for the norm used in the penalization, and solver for the algorithm used in the optimization problem) was performed. It ensures the model is neither overfits nor underfits balancing model complexity.

Then the model is trained and is evaluated using F1 and accuracy metrics.

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
Best parameters found: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
Logistic Regression Accuracy: 0.8033608890093509
F1-score: 0.7976007811410238
```


This model gives an accuracy of around 80% and F1 score of 79.7%, which is comparatively higher than the Gaussian NaïveBayes model. logistic regression outperformed Gaussian Naive Bayes, suggesting that it captures the patterns in the data more effectively.

It proves to be a suitable algorithm for the classification task providing good accuracy and F1 score. Its interpretability makes it valuable for understanding the factors influencing property prices, and its efficiency makes it scalable to larger datasets.



Based on this ROC curve obtained, the logistic regression model performs well.

Decision Tree Classifier:

Decision Tree classifier is used for the classification task of predicting whether a property's price is above or below the median. Decision Trees provide a feature importance score, which indicates the most critical features in making classification decisions. Decision trees work well for both categorical and numerical data, making them versatile for various Airbnb listing features like location, amenities, room type, and prices.

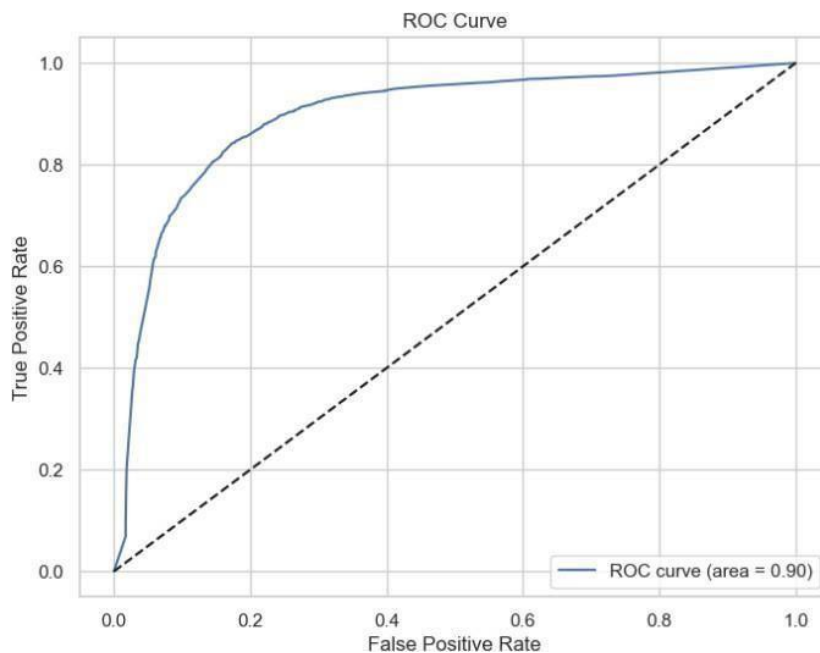
The Decision Tree model is trained on the training data, where it recursively partitions the feature space based on the target variable.

A grid search over the `max_depth` (to control the complexity of the tree and prevent overfitting) and `criterion` (to choose the best split) parameters was performed. This step maximizes accuracy by avoiding creation of overly complex trees that do not generalize well.

The trained model is evaluated using accuracy and F1 score metrics on the test set to measure its performance.

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best parameters found: {'criterion': 'entropy', 'max_depth': 10}
Decision Tree Accuracy: 0.8341238650223608
F1-score: 0.8293838862559243
```

This model gives an accuracy of around 83.4% and F1 score of 82.9%. Though this model gives a good performance, their interpretability and ability to capture nonlinear relationships make them a valuable tool for understanding and predicting property prices.



This model gives the best ROC curve as the area under the curve is 0.90.

Gradient Boosting Regressor:

GRADIENT BOOSTING:

Gradient Boosting Regressor is a powerful ensemble learning technique that can effectively capture complex patterns in the data by combining multiple weak learners. It handles missing data during training and as well as outliers and helps to alter the weights of the features in order find the data patterns. Given the project aims to optimize Airbnb rental strategies in New York City, GBR's ability to model such complexities makes it an excellent choice for predicting pricing and demand, analyzing guest satisfaction patterns, and understanding the impact of time and location on rental demand.

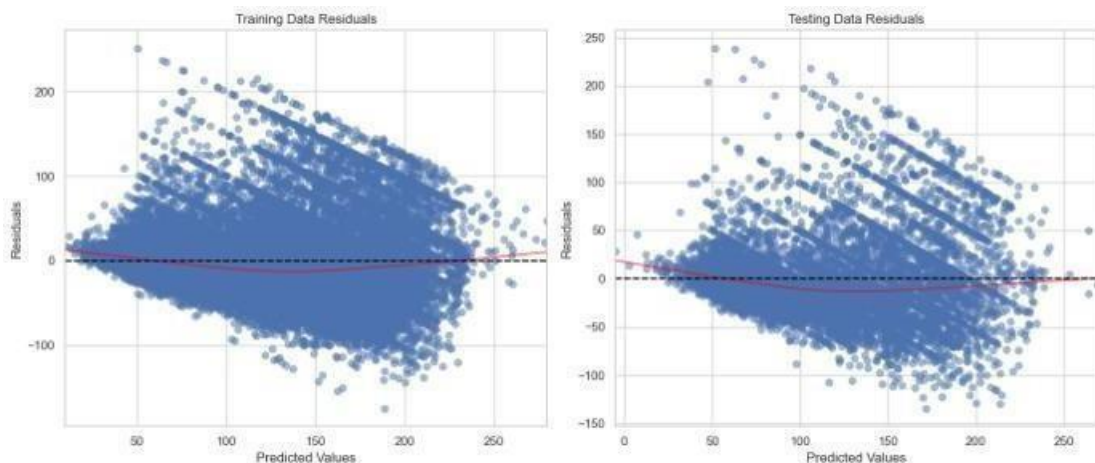
Training: An instance of Gradient Boosting Regressor is created with specified hyperparameters such as the number of estimators (`n_estimators`), learning rate (controls the contribution of each tree to the final outcome), fitting is done using 5- fold CV to evaluate model's performance.

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best parameters found: {'learning_rate': 0.2, 'n_estimators': 150}
Gradient Boosting RMSE: 43.67854189735724
R-squared: 0.5771948702824106
```

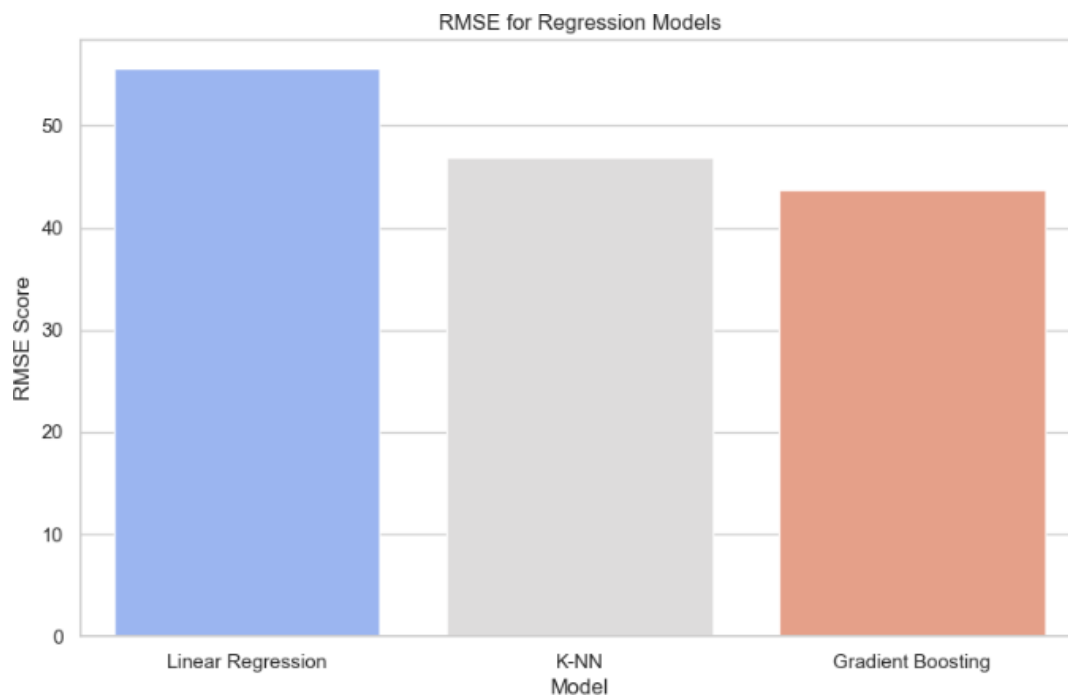
The RMSE value of approximately 43.67 indicates the average magnitude of the errors made by the model in its predictions. Lower RMSE values are desirable, as they indicate better agreement between predicted and actual values.

R-squared values range from 0 to 1, where higher values indicate better fit. The R-squared value of approximately 0.577 shows that it has captured more than 50% of the data patterns of the dataset.

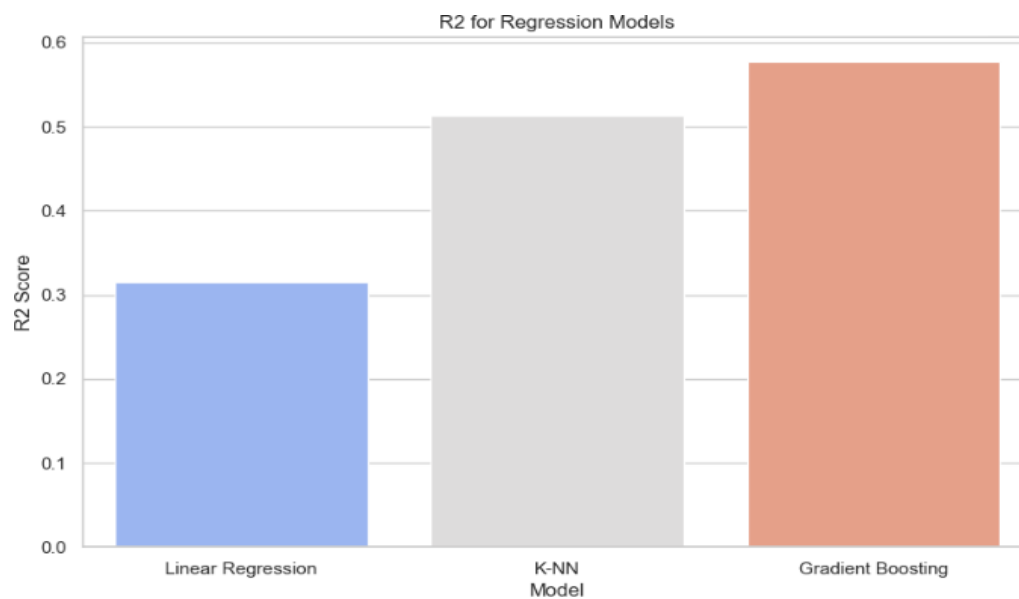
Gradient boosting regressor shows moderate effectiveness in prediction. While model has a low accuracy it also provides some useful insights of the underlying patterns which can be used in analysis.



Comparison of Regression Models:



RMSE scores are compared and a bar graph is plotted for the regression models. It can be seen that Gradient Boosting has less RMSE score meaning that it performs well than Linear and KNN model.

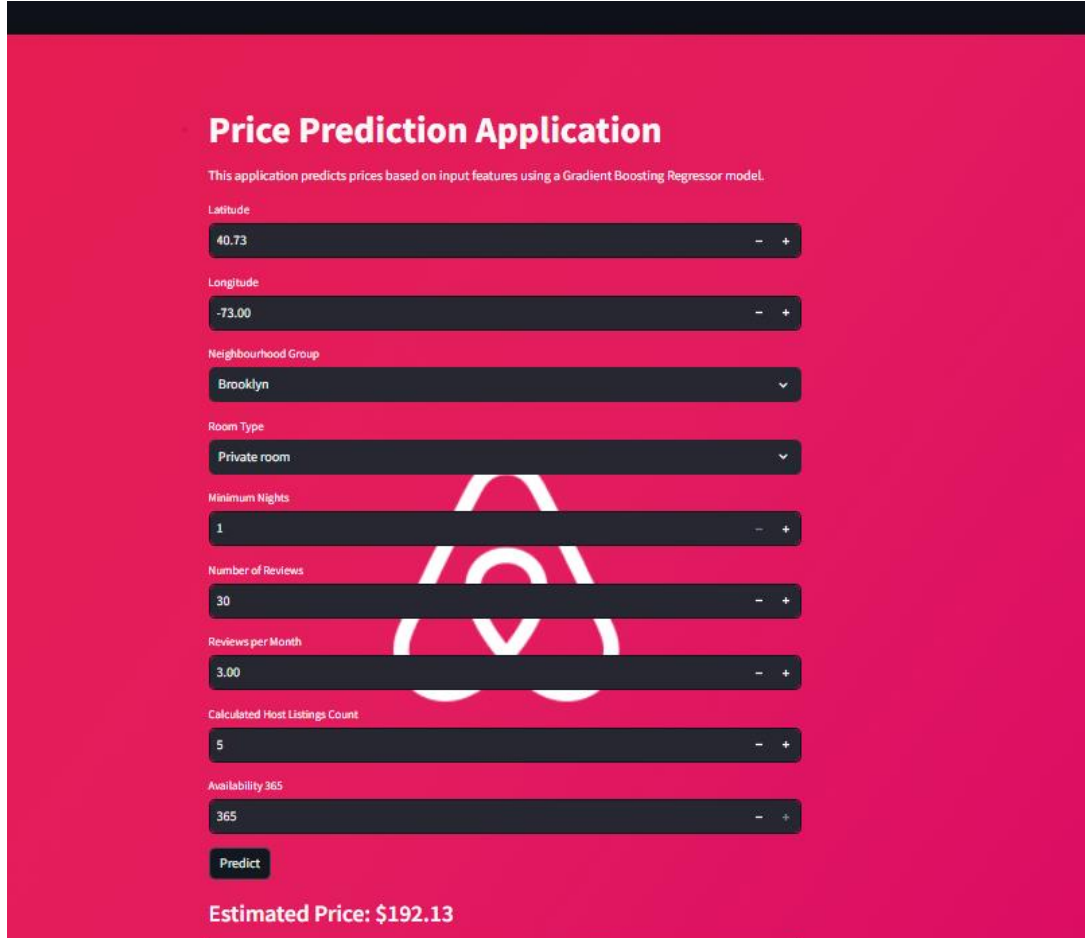


R2 scores are compared and a bar graph is plotted for the regression models. It can be seen that Gradient boosting scored higher meaning that it performs well than Linear and KNN model.

PHASE-3

WEBPAGE:

Below is a sample prediction that was made using a set of input parameters in the webpage.



The screenshot shows a web application titled "Price Prediction Application" on a dark blue background. Below the title, a subtitle reads: "This application predicts prices based on input features using a Gradient Boosting Regressor model." The interface contains several input fields, each with a label, a value, and a minus/plus control:

- Latitude: 40.73
- Longitude: -73.00
- Neighbourhood Group: Brooklyn (dropdown menu)
- Room Type: Private room (dropdown menu)
- Minimum Nights: 1
- Number of Reviews: 30
- Reviews per Month: 3.00
- Calculated Host Listings Count: 5
- Availability 365: 365

At the bottom left is a "Predict" button. At the bottom right, the "Estimated Price: \$192.13" is displayed. A large, faint white Airbnb logo is visible in the background of the form area.

1) A) Working Instructions of the product

For running this product, first the dependencies such as streamlit, pandas, numpy, scikit-learn and matplotlib are imported/installed. Streamlit is used to build this web application for the Airbnb price prediction.

Scaler is used to scale the user-input features for the trained machine learning model.

Gradient Boosting Regressor model is converted to a pickle file, Standard scalar is converted to a pickle file and loaded into streamlit code (saved as app.py). The files required for the website such as app.py, requirements.txt, gradient_boosting_regressor.pkl, scalar.pkl are uploaded to a new github repository and the streamlit website is launched by linking it to the Github repository that takes in the required files for the website to be deployed.

In the app.py python file, a function is used to collect user input for various features related to an Airbnb listing. Users can input values for features such as latitude, longitude, neighbourhood_group, room_type, minimum nights, number of reviews, reviews per month, calculated host listings count, and availability 365. The function returns a DataFrame containing these features. A transform function is used to transform these features and finally prediction is performed to predict the price using the trained model on the user input data frame.

Website link: <https://gxvw3jcjtjfejvsg6ztj2x.streamlit.app/>

In case if the website is not opening on the Chrome browser, try to open on the incognito mode or use the other browser.

Launch the webpage using this link. In the app interface, you'll find nine input fields:

- Latitude: Input for the latitude coordinate of the listing. Minimum and Maximum values are given as -90° and 90° respectively. Default value is the minimum value i.e -90°
- Longitude: Input for the longitude coordinate of the listing. Minimum and Maximum values are given as -180° and 180° respectively. Default value is the minimum value i.e -180°

```
latitude = st.number_input('Latitude', min_value=-90.0, max_value=90.0)
longitude = st.number_input('Longitude', min_value=-180.0, max_value=180.0)
```

- Neighborhood Group: Neighborhood within where the listing is located. This is a dropdown field, the user can select the group from the five groups such as Brooklyn, Manhattan, Queens, Staten Island, Bronx.
- Room Type: This is a dropdown field, the user can select the group from three room types such as Private room, Entire home/apt, Shared room.
- Minimum Nights: Enter the minimum number of nights for staying.
- Number of Reviews: Input the total number of reviews received by the listing.
- Reviews per Month: Enter the average number of reviews received per month.
- Calculated Host Listings Count: Input the total number of listings the host has in the dataset.
- Availability 365: Enter how many days in a year the listing is available for booking. The value can range from 0 to 365 days.

Once you've entered the necessary data, click the "Predict" button. The machine learning model will process the input features and provide an estimated price based on the user's input values.

These potential features help in more accurate price prediction for the particular listings. Each feature has individual weights such that based on the user entered value the pricing is produced.

Below is the snip of how the pickle files were generated for the purpose of predicting the prices:

```
In [70]: import pickle
model_filename = 'gradient_boosting_regressor.pkl'
with open(model_filename, 'wb') as file:
    pickle.dump(grid_search.best_estimator_, file)
```

```
In [77]: from sklearn.preprocessing import StandardScaler
import pickle

scaler = StandardScaler()
scaler.fit(X_train)

# Save the scaler
with open('scaler.pkl', 'wb') as file:
    pickle.dump(scaler, file)
```

Latitude
-10.00

Longitude
-60.00

Neighbourhood Group
Manhattan

Room Type
Private room

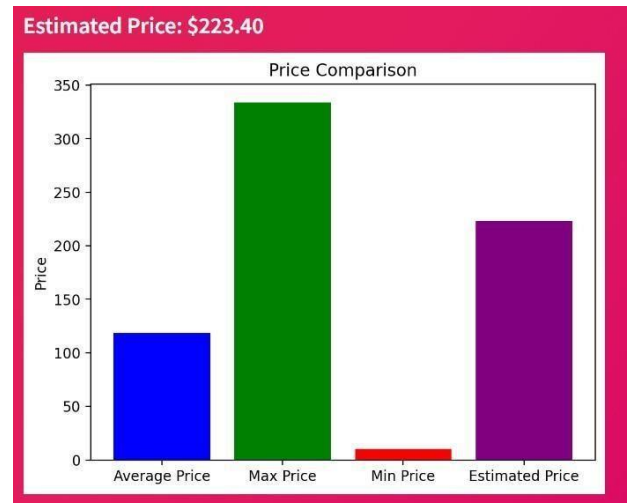
Minimum Nights
2

Number of Reviews
5

Reviews per Month
0.40

Calculated Host Listings Count
0

Availability 365
10



In the above scenario, the user is trying to book the listings, the values given by the user are:

- Latitude: -10°
- Longitude: -60°
- Neighborhood Group: Manhattan
- Room Type: Private Room
- Minimum nights: 2
- Number of Reviews: 5 reviews
- Reviews per month: 0.4
- review/month Calculated Host listings
- count: 0
- Availability 365: 10 days

The listing price based on the user's inputs is predicted as \$223.40 as shown in the plot.

The estimated-price of the listing is visualized in a bar-plot that compares it with the average, maximum, and minimum prices and estimated price for the listing. This visual representation aids the hosts to quickly interpret the estimated price with respect to average, maximum and minimum prices.

1) B) Implementation of the model on the Webpage

In phase 2, different machine learning models are run on the New York city Airbnb dataset. It is found that Gradient Boosting Regressor is the best model for our dataset giving the best results. Gradient Boosting Regressor is a powerful ensemble learning technique that can effectively capture complex patterns in the data by combining multiple weak learners. It handles missing data during training and helps to alter the weights of the features in order to find the data patterns. It finds any nonlinear relationship between the model target and features and has great usability that can deal with missing values, outliers, and high cardinality categorical values on the features.

Training/Tuning: We performed hyperparameter tuning using GridSearchCV. An instance of Gradient Boosting Regressor is created with specified hyperparameters such as the

- number of estimators (n_estimators) i.e, the number of boosting stages to be used. We have given the values of 100 and 150.
- Learning rate: The step size at which the model adapts during training. We have values of 0.1 and 0.2.

Fitting is done using 5-fold CV to evaluate model's performance. GridSearchCV searches through the specified hyperparameter combinations to find the best set of parameters.

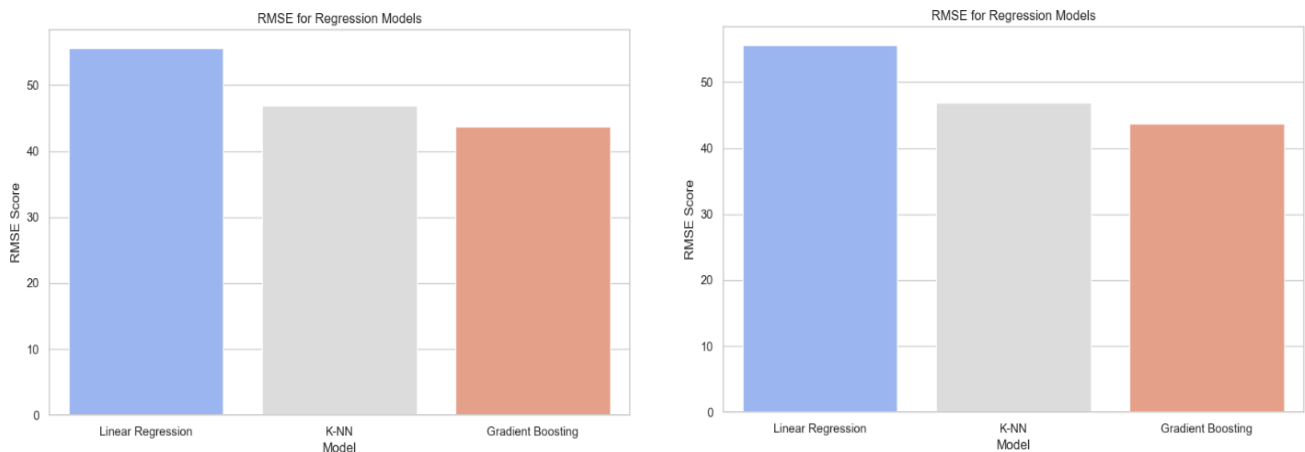
Best parameters found by the model were:

n_estimators: 150

learning_rate: 0.1

The model gives the evaluation metrics as: RMSE: 43.67 and R-Squared: 0.57

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best parameters found: {'learning_rate': 0.2, 'n_estimators': 150}
Gradient Boosting RMSE: 43.67854189735724
R-squared: 0.5771948702824106
```



The RMSE value of approximately 43.67. RMSE value gives the average magnitude of the errors made by the model in its predictions. This model gives the least RMSE value when compared to all the models.

R-squared values range from 0 to 1, where higher values indicate better fit. The R-squared value of approximately 0.577 shows that it has captured more than 50% of the data patterns of the dataset.

Given the project aims to optimize Airbnb rental strategies in New York City, GBR's ability to model such complexities and also higher R-squared value and lower RMSE value makes it an excellent choice for predicting pricing and demand, analyzing guest satisfaction patterns, and understanding the impact of time and location on rental demand.

1.C.

The project aims on providing the hosts with tools and insights to optimize their rental strategies.

Predicting the estimated price for an Airbnb listing based on features like location, room type, minimum nights, and other attributes can be valuable to hosts in several ways:

Optimal Pricing Strategy: By knowing the estimated market price for a listing similar to theirs, hosts will be able to adjust their own pricing to stay competitive. This helps in setting a price that is neither too high, which might put hosts in a situation to lose potential guests, nor too low, which could mean losing out on potential revenue.

Understanding Demand: This predictive model might also help hosts understand how demand varies with changes in price and other factors like location. This insight allows hosts to adjust their prices dynamically based on expected demand and they can vary their prices on and off season.

Increased Revenue: Hosts can optimize their pricing strategy to maximize their income. By pricing appropriately, they can improve their chances of booking and reduce vacant periods, and potentially attract more guests by offering competitive rates.

Strategic Decision Making: Insights from the model can aid in making strategic decisions such as when to list the property, how to market and whether investing in upgrades or additional amenities could justify a higher pricing tier based on similar listings.

Improvements: By comparing their properties to similar ones in the provided dataset, hosts can identify areas for improvement that could enhance guest satisfaction and allow them to command a higher price or increase their occupancy rate.

On the whole, the predictive pricing using gradient boosting regressor model acts as a decision support tool, helping hosts fine tune their operations based on data driven insights, leading to better financial outcomes and improved guest experiences.

Further Extension/ Analysis of the project:

Purpose of our model is to educate the hosts with the pricing details for the neighbourhood based on the features they enter. This helps the hosts to understand the pricing based on the location of the listing and maximizing the accuracy of the price using other features.

Information about safety and nearby amenities can also be provided to the guests. Travelers can choose neighbourhoods that align with their preferences. The option to collaborate with local businesses who offer guided tours and assistance for travelling can be implemented on the webpage. We can also think about extending the app to predict maintenance needs (e.g., HVAC servicing, appliance replacements) based on usage patterns.

Property Investment Analysis helps potential investors evaluate the profitability of purchasing a property for Airbnb hosting. Sustainability Metrics may be performed to incorporate eco-friendly features like solar panels, energy-efficient appliances and educate hosts and travellers about sustainable practices.

2.a.

Demo video has been uploaded.

Website deployment using python in streamlit:

Files app.py requirements.txt , scaler.pkl and gradient_boosting_regressor were created in a directory streamlit_app and pushed into a new github repository.

Pickle.dump() is used to generate both the pickle files scaler.pkl and gradient_boosting_regressor.pkl.

app.py - is the code file for the streamlit website to be deployed.

scaler.pkl-pickle file for standard scaler processing.

gradient_boosting_regressor.pkl-pickle file of the gradient boosting regressor model that is used to predict the price.

requirements.txt-text file that has all the necessary libraries that need to be imported.

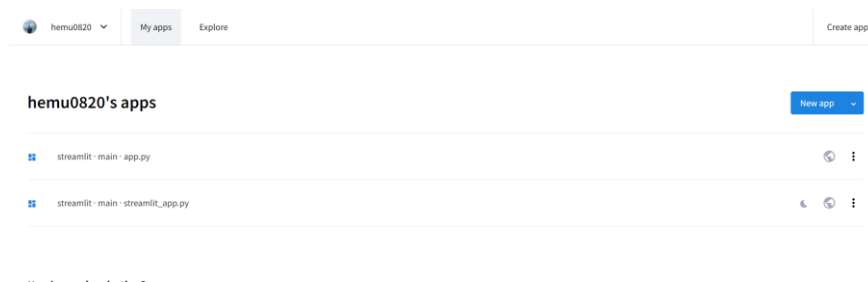
The link to the Github repository was used in the streamlit creation along with the required available name for the website.

Below is the Github repository link:

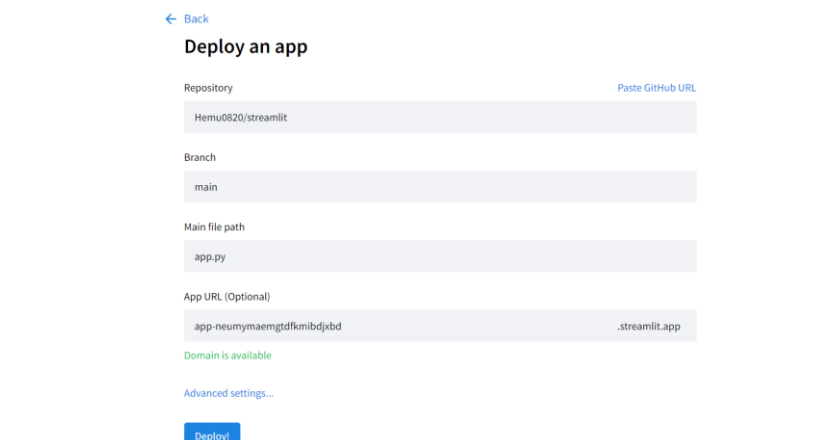
<https://github.com/Hemu0820/streamlit>

Procedure to deploy the website using streamlit:

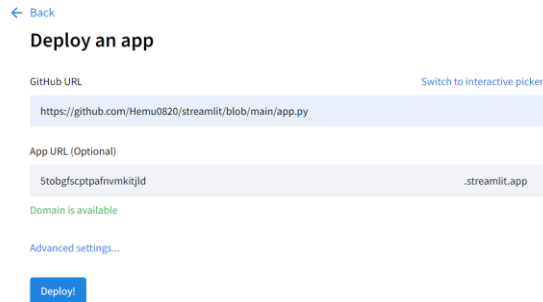
A new app is created by clicking on the new app button on the top right corner and below is the screenshot for the same.



The github repository ,branch and main file path are mentioned and an app url is chosen according to our convenience.



Github repository Url that redirects to app.py is specified and upon clicking the deploy button our website is deployed.



2.b.

Below is the sample set of inputs that user can provide to predict the price and get a visualization of the same.

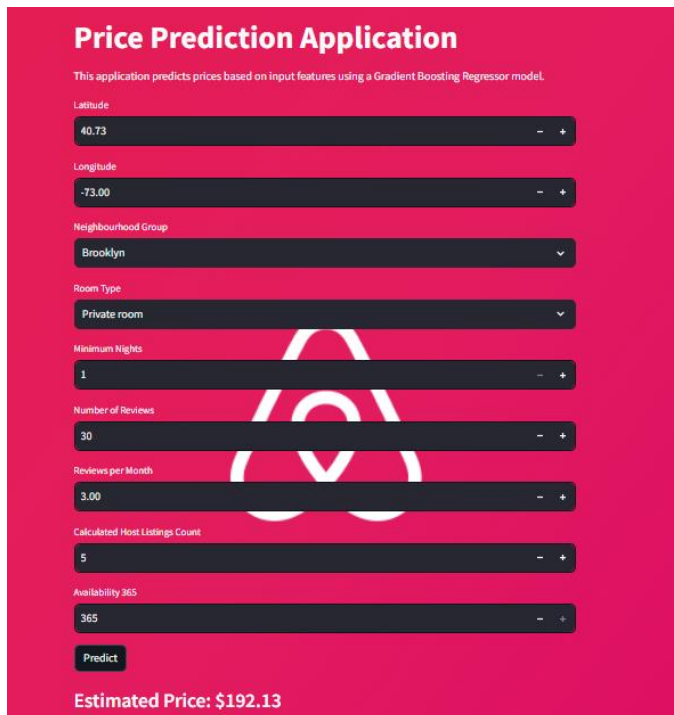
Combination Set 1:

Latitude: 40.7128
Longitude: -74.0060
Neighbourhood Group: Manhattan
Room Type: Entire home/apt
Minimum Nights: 3
Number of Reviews: 45
Reviews per Month: 0.5
Calculated Host Listings Count: 2
Availability 365: 180

Combination Set 2:

Latitude: 40.7306
Longitude: -73.9352
Neighbourhood Group: Brooklyn
Room Type: Private room
Minimum Nights: 1
Number of Reviews: 30
Reviews per Month: 3.0
Calculated Host Listings Count: 5
Availability 365: 365

Below is the predicted price using the combination set 2 after inputting the parameters.



Price Prediction Application

This application predicts prices based on input features using a Gradient Boosting Regressor model.

Latitude: 40.73

Longitude: -73.00

Neighbourhood Group: Brooklyn

Room Type: Private room

Minimum Nights: 1

Number of Reviews: 30

Reviews per Month: 3.00

Calculated Host Listings Count: 5

Availability 365: 365

Predict

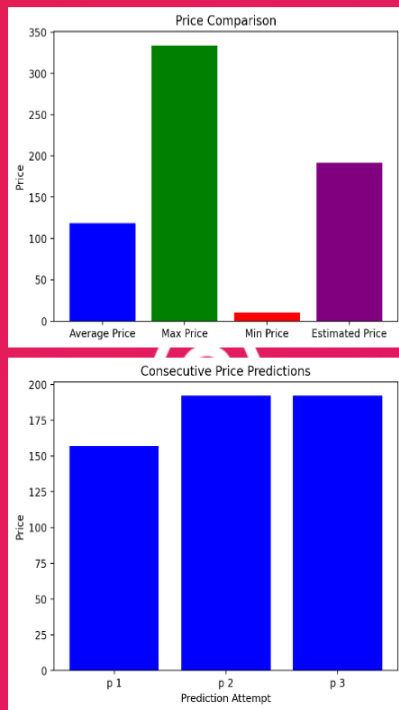
Estimated Price: \$192.13

2.c.

The below bar plot represents the price interpretation with respect to min ,max and average price.

The consecutive price predictions are the result of consecutive set of inputs with different combination sets that would help the hosts to compare the prices range as their choice of input parameters.

We have 3 consecutive bar plots here which were generated as a result of 3 different combination set of inputs.



We have written the app.py code in such a way that it generates 10 different predictions results for 10 different set of combinations and once the limit of 10 is exceeded we will pop the first prediction.

Below is the code logic for the same:

```
if 'predictions' not in st.session_state:
    st.session_state['predictions'] = []
if len(st.session_state['predictions']) >= 10:
    st.session_state['predictions'].pop(0)
st.session_state['predictions'].append(predicted_price)
```

The feedback that our product gives is the comparisons of predicted prices, it means to understand the prices ranges for a particular location and other combination of inputs parameters.

Hence the visualization obtained above would help the hosts understand and compare the prices for different set of neighborhoods, room_types and other input parameters.

References:

<https://docs.python.org/3/library/pickle.html>

<https://github.com/streamlit/streamlit>

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>