Airbnb Pricing and Crime in Seattle

Group members: Chenchen Jiang, Hui Du, Jianjian Liu, Jiayang Liu

Abstract: In this project, we analyzed the relationships between consumer concerned features (accommodates, bedroom number, bathroom number, beds number, review rating score, neighborhood, room type and bathroom type) and lastest Airbnb rental price (2023-2024) in Seattle. We also considered crime data (2008-2024) as a potential factor by merging it into the main Airbnb dataset. Raw dataset from Airbnb and Data.Seattle.gov were cleansed by replacing and dropping missing values, and categorical features were converted into one-hot encoding format. We explored both Airbnb and Crime datasets through visualization to better understand the data distribution before modeling. We utilized three machine learning models—Linear Regression, Random Forest, and Gradient Boosting—for predicting suggested listing prices. Techniques such as Stacking and Bagging were applied respectively to optimize model performance. As a result, the Gradient Boosted model showed the best performance, evaluated using parameters including MSE, MAE, R2, and the residual curve. Furthermore, to provide a user-friendly interface in applying our model, an online application was built by Streamlit, and we also generated a docker image for our application. In summary, this project represents an innovative and beneficial use of data science as it empowers property owners to make data-informed decisions before they list their homes.

Introduction

Airbnb offers alternatives to the traditional hotel. It's an online application which enables people to lease or rent short-term homestays, apartments and hotel rooms. As Airbnb grows into a global phenomenon, it will be useful to provide people with a price-comparison tool in choosing their rentals. In this project, we will explore the key factors which could influence Airbnb rental prices, and our aim is to provide insightful information for the property owners before they enter the Seattle Airbnb market.

The primary objective of this project is to predict the listing distribution and the price fluctuation of Airbnb in Seattle. This exploration will be accomplished by analyzing the relations among rental options (such as area, room type, bedroom number and so on) and price of Airbnb rooms. This analysis and application will provide valuable insights for guiding room selection, devising effective marketing strategies and enhancing homestay sales predictions.

Our analysis will mainly rely on Airbnb listings. To gain a comprehensive understanding of the data and establish relationships between variables, we will employ data mining techniques and exploratory data analysis. This will encompass data cleansing and preprocessing, regression and correlation analysis, the application of data mining models, and model evaluation and validation and more. Figure 1 shows our overall workflow.

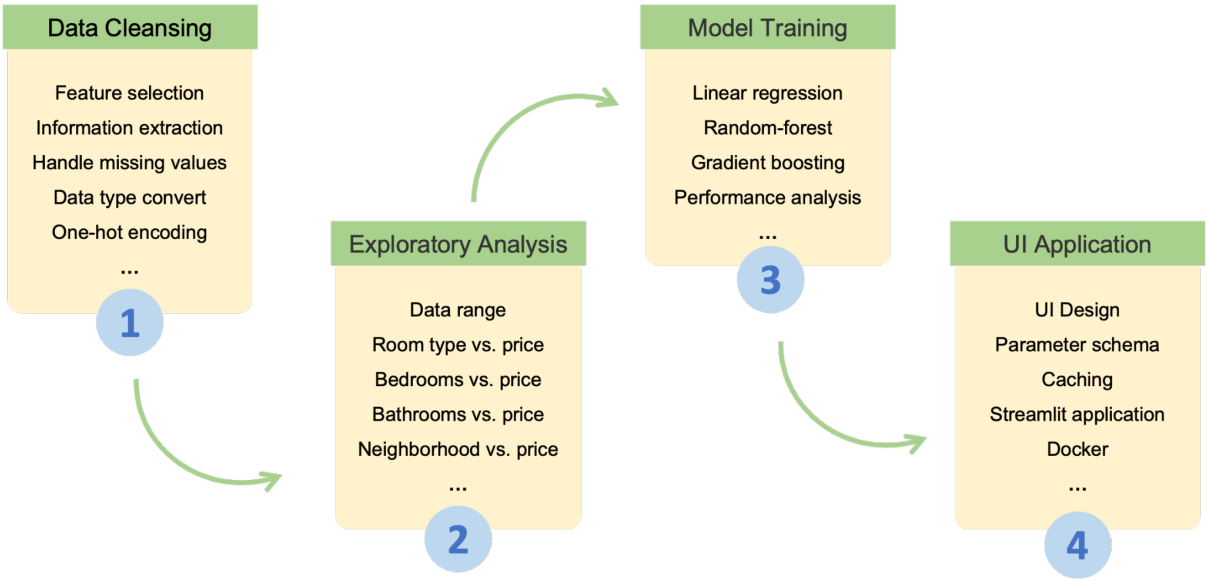


Figure 1. Workflow Chart

In conclusion, we trained three different regression models, which showed acceptable performance in predicting the rental price. In addition, we provided the UI by Streamlit, as well as Docker service to allow users to get access to our model.

Datasets and Features Selection：

1. Airbnb Listing Data:
    - **id**: (numeric) listing ID for the listed property
    - **description**: (text) description of the listed property, including detailed information
    - **neighborhood**: (text) the name of the community listed property
    - **latitude**: (text) latitude of the listed property
    - **longitude**: (numeric) longitude of the listed property
    - **property type**: (text) type of the listed property
    - **room type**: (text) type of the listed room
    - **accommodates**: (numeric) number of people can live
    - **bathroom_text**: (text) number of the bathroom and type of the bathroom
    - **bedroom**: (numeric) number of the bedroom
    - **beds**: (numeric) number of the beds
    - **price**: (numeric) price of the listed property
    - **review_scores_rating**: (numeric) the overall rating scores of the property
    - **review_scores_accuracy**: (numeric) the accuracy rating scores of the property
    - **review_scores_cleanliness**: (numeric) the cleanliness rating scores of the property
    - **review_scores_checkin**: (numeric) the check-in rating scores of the property

- **review_scores_communication**: (numeric) the communication rating scores of the property
- **review_scores_location**: (numeric) the location rating scores of the property
- **review_scores_value**: (numeric) the accuracy rating scores of the property

Note: Due to the complexity of Airbnb property listings, some owners do not list their entire property but only a few rooms (either shared or private) within their property, such as a house, apartment, condo, or boat. The property type indicates the type of the property, while the room type provides information about what is included in the listing.

2. Crime Data:
   - **Report DateTime:** the report date and time
   - **Crime Against Category**: (categorical) crime category e.g. Society, Property
   - **Offens**e: (categorical) offense category
   - **100 Block Address**: (categorical) the offense location address
   - **Longitude**: (numeric) the longitude of the location
   - **Latitude**: (numeric) the latitude of the location

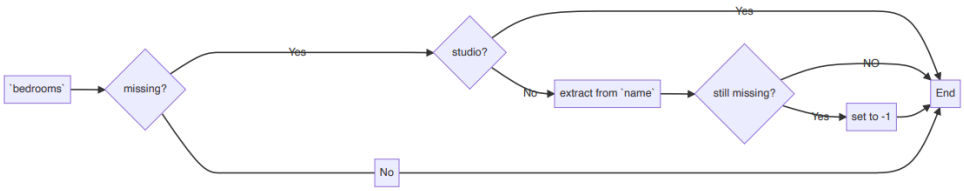
3. Neighborhood_Map_Atlas_Neighborhoods:(GEOJSON file)
   - **L_HOOD**: (text) large neighborhood name
   - **S_HOOD**: (text) small neighborhood name
   - **Geometry**: (polygon) coordinates for the neighborhood

Data Cleansing and Processing

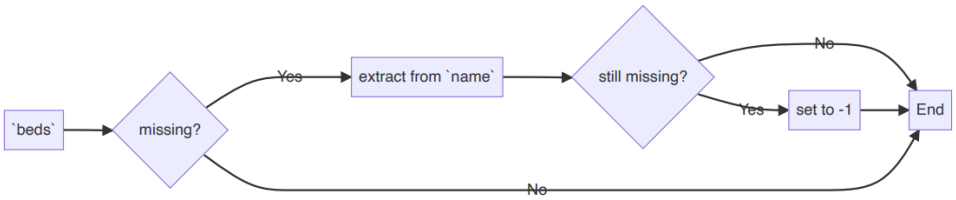
1. Airbnb Listing Data:
   - Split bedroom_text to bedroom_count and bedroom_type features.

bathrooms_text	bathroom_count	bathroom_type
1 bath	1.0	standard
3 shared baths	3.0	shared

   - Bedroom: If it is studio in description, the bedroom number will be set as 0. Otherwise, missing values will be substituted using information extracted from column name



   - Bed: missing values will be substituted using information extracted from column description.



- Percentage of the missing value:

```
missing_percentage = df.isna().sum() / len(df) * 100
print(missing_percentage)
```

```
id                              0.000000
availability in one year        0.000000
name                            0.000000
neighbourhood_cleansed          0.000000
latitude                        0.000000
longitude                       0.000000
property_type                   0.000000
room_type                       0.000000
accommodates                    0.000000
bathrooms_text                  0.000000
bedrooms                        0.000000
beds                            0.000000
price                           0.000000
number_of_reviews               0.000000
review_scores_rating           24.312485
review_scores_accuracy         24.361158
review_scores_cleanliness      24.361158
review_scores_checkin          24.361158
review_scores_communication    24.361158
review_scores_location         24.361158
review_scores_value            24.361158
bathroom_count                  0.000000
bathroom_type                   0.000000
dtype: float64
```

2. Crime Data:

- Missing value: we retained several key features for further use: report datetime, crime category, offense, address, longitude, and latitude, which are all categorical features. Some address entries are missing (NA), but as these constitute only 4.67% of the data, we drop the rows with missing addresses.

```
na_counts = df_crime_filtered.isna().sum()
na_rate = na_counts / len(df_crime_filtered) *100
na_rate
```

✓ 0.1s

```
Report DateTime           0.00000
Crime Against Category    0.00000
Offense                   0.00000
100 Block Address         4.46231
Longitude                 0.00000
Latitude                  0.00000
dtype: float64
```

- Add neighborhood name: we created columns of the geometry point by using coordinates of crime cases, and we used geopandas.join() to connect crime data and neighborhood map data to assign the neighborhood name to each crime case.

- Crime case count: we counted the crime case for each neighborhood.

3. Final data:

   - We merged the airbnb listing data and crime case count data by using the neighborhood name.

Exploratory Analysis

1. Data Range

   Before moving into training the regression model using a cleaned dataset, we first applied some intuitive analysis of the data, so we made a boxplot as Figure 2. Interestingly, we observe some outliers in feature price: a few rooms seem to have incredibly high rental prices. Further analysis showed that the upper quartile of price is $187, while the max price is $10,000. Notably, only 6 rooms had prices higher than $4,000. Therefore, we identified rooms with prices higher than $4,000 as outliers and removed them for model training. After removing the outliers, the highest price of rooms is $3,071. However, it's reasonable because the house is prepared for up to 15 people.

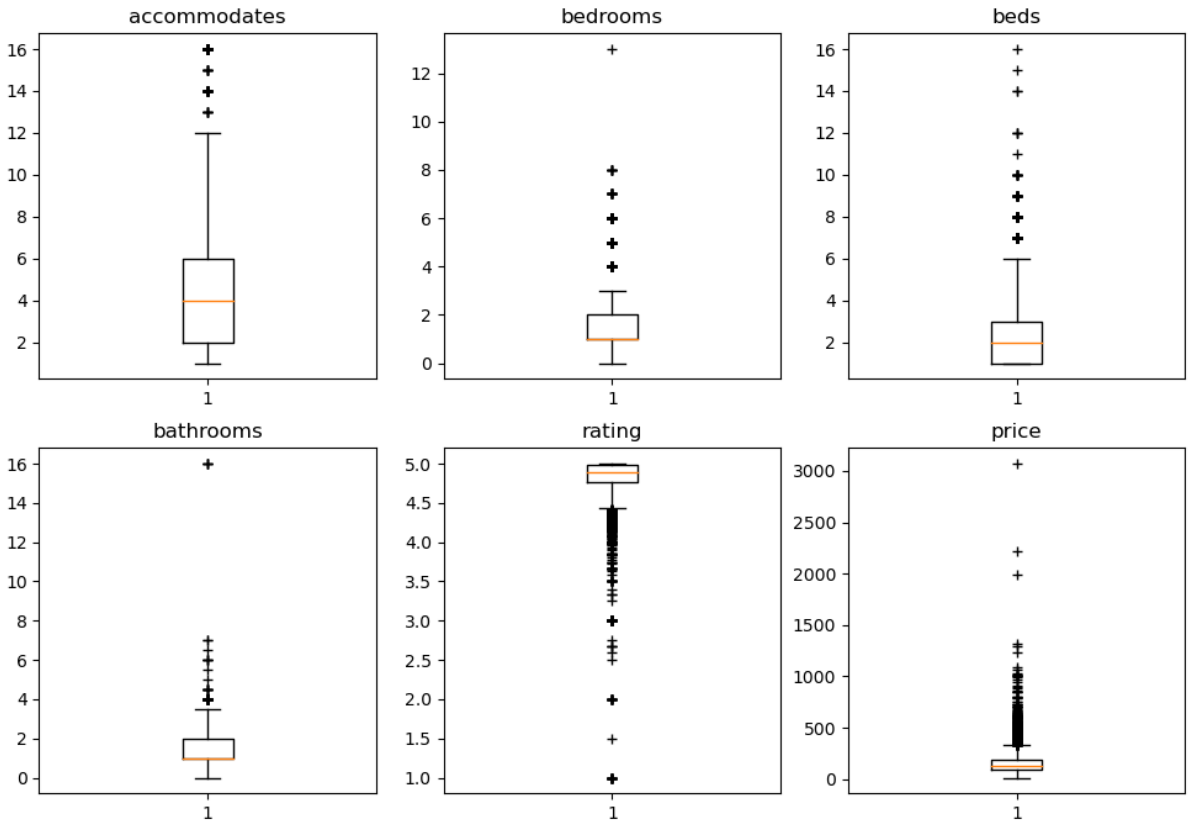


Figure 2. Boxplot of Features

2. Relation to Price

We next analyzed the possible relations between our selected features (both numerical and categorical features) and the price. As for numerical features (accommodates, bedrooms, beds, bathrooms, rating), though not so obvious, it seems that these features do have some relations to the price. However, it's hard to tell whether the price is influenced by the area of the house from the chart we have here. As for room and bath types, it seems that shared bedrooms or bathrooms are a little more inexpensive than the standard type.

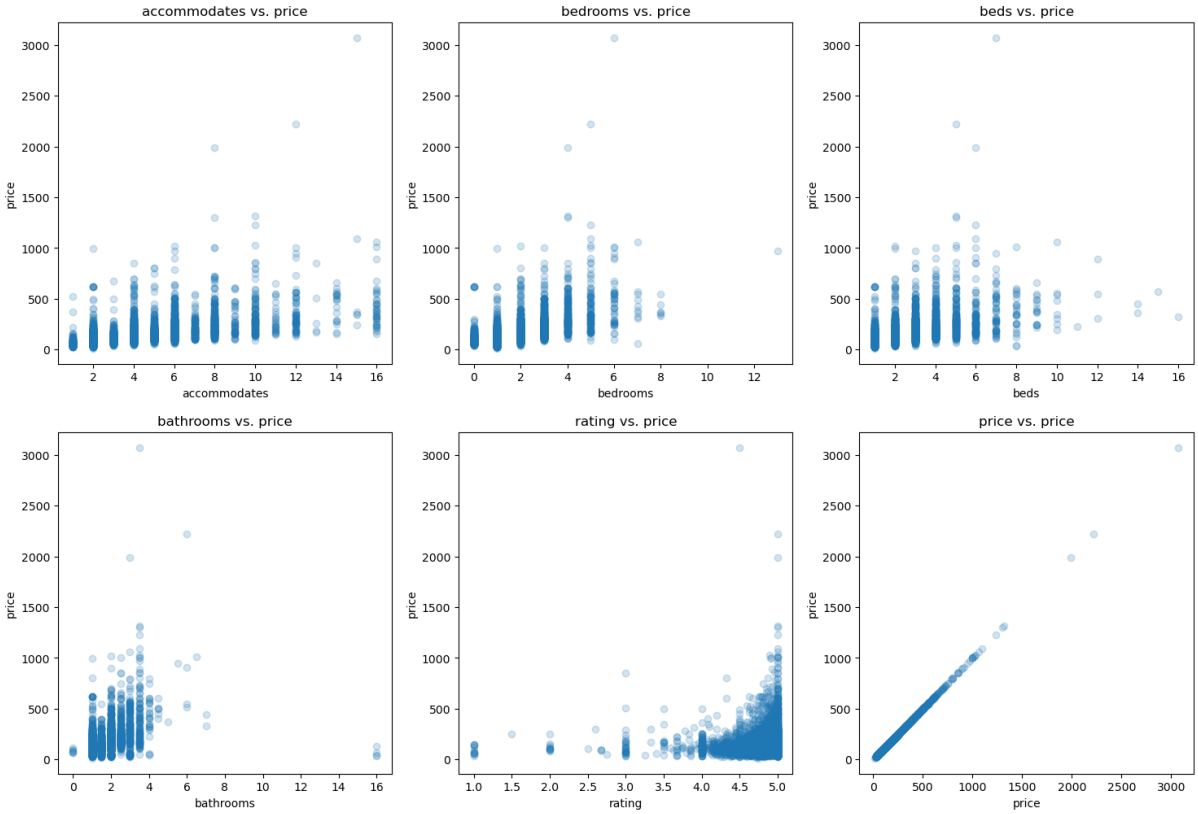


Figure 3. Scatter Plot Price and Dependent Variables

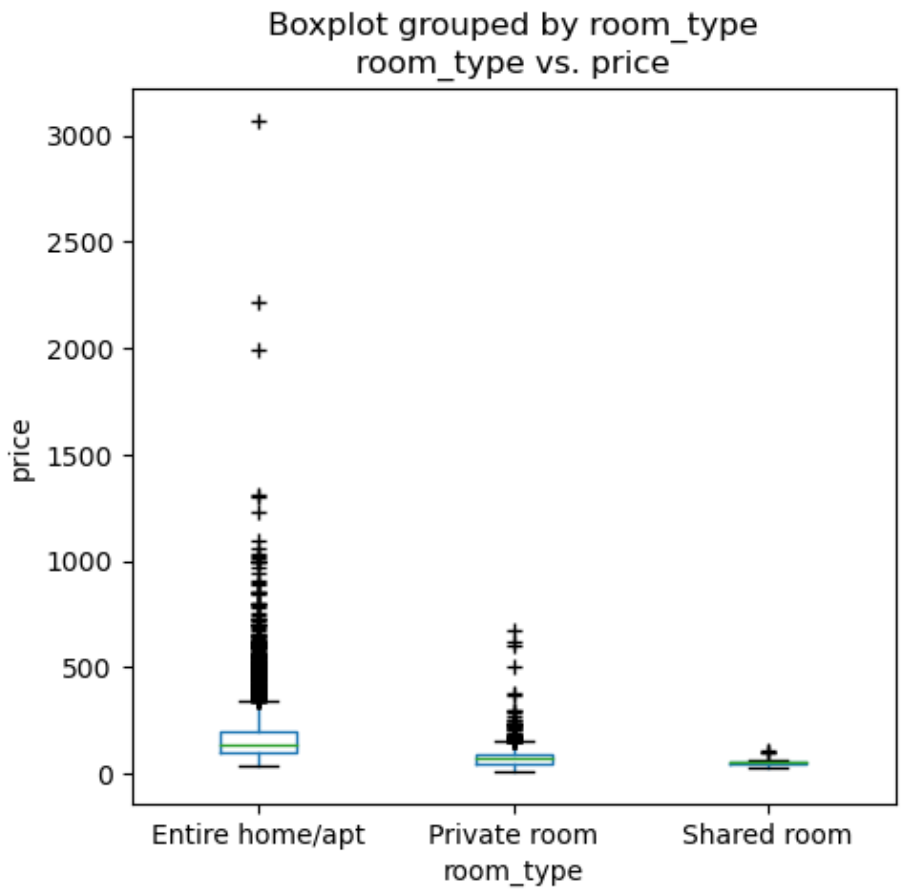


Figure 4. Boxplot of Price and Property Type

3. Area Distribution

We are examining the distribution of Airbnb prices and total crime rates across different neighborhoods in Seattle, using a choropleth map for visual representation.

Figure 5 indicates that the Industrial District is highlighted as the most expensive Airbnb neighborhood, marked in dark red, with an average price of $495. This is followed by the BriarCliff ($225) and Central Business District ($200) neighborhoods.

Figure 6 shows the neighborhoods with the highest crime rates are depicted in the darkest shades: Central Business District (66k reported incidents), Broadway (55k), and Belltown (32.5k), which are primarily located in the center of Seattle. The interactive map can be viewed in our Streamlit Application.

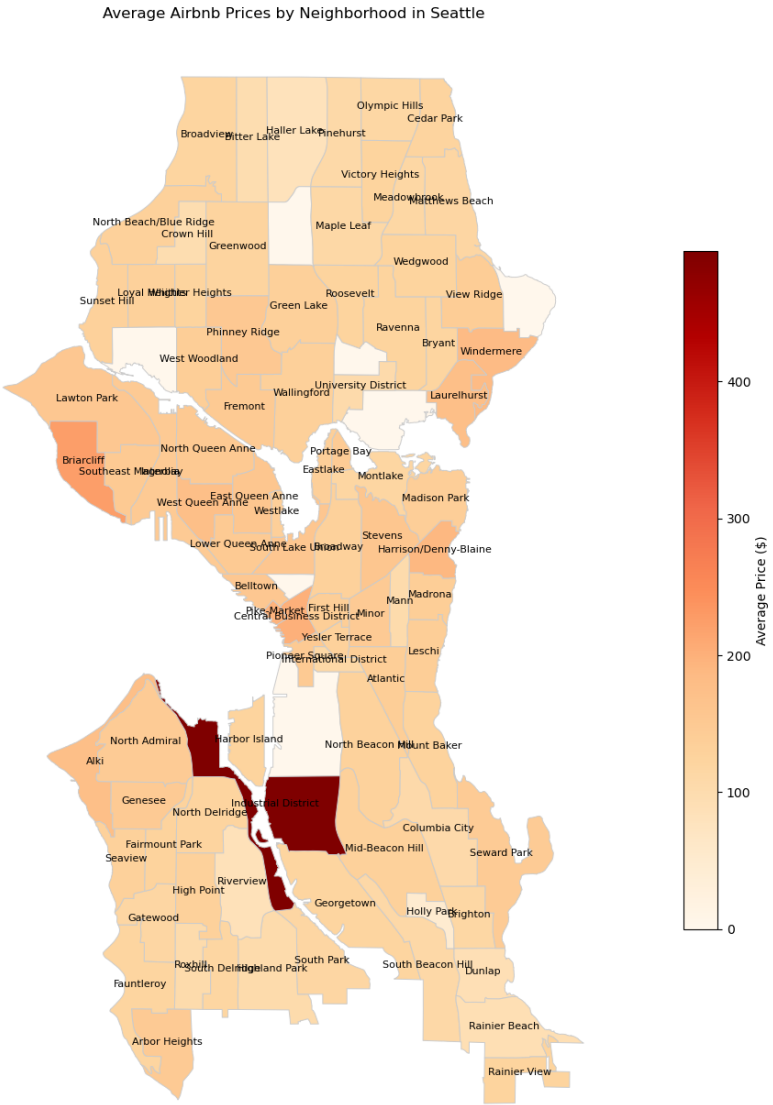


Figure 5. Choropleth Map of Average Airbnb Prices by Neighborhood in Seattle

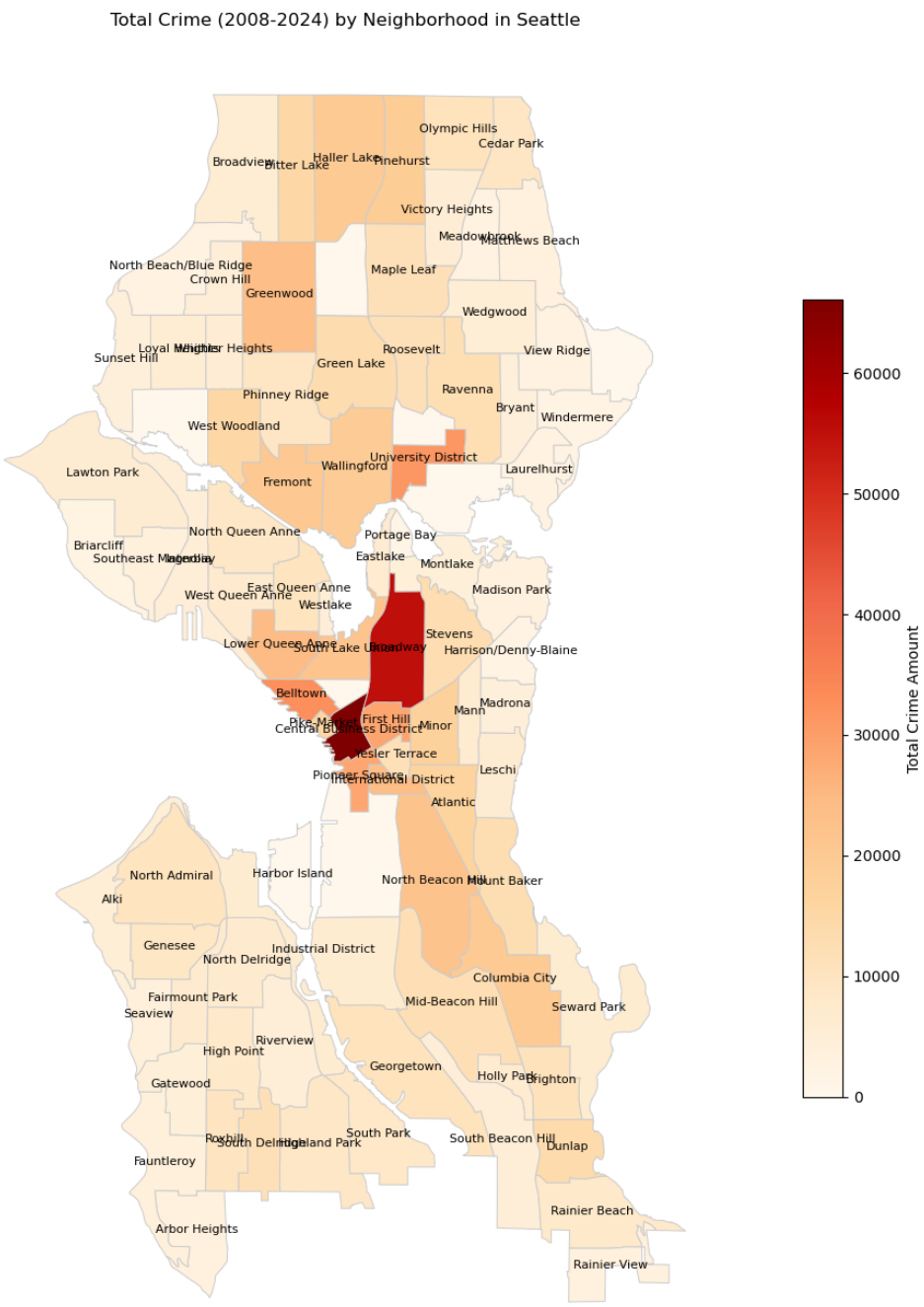


Figure 6. Choropleth Map of Total Crime (2008-2024) by Neighborhood in Seattle

Modeling

1. Feature Selection and additional process:
    - Independent variables: area, property type, room type, accommodates, bedrooms, beds, total crime.
    - Dependent variables: price
    - Remove outliers by using z-scores

2. Model Initialization

In training the regression model, the dataset was splitted into a training set (70%) and testing set (30%), and three different regression models were trained for selecting the best.
For our model, it takes independent variables (neighborhood, property type, room type, accommodates, bedrooms, beds, total crime) to predict the dependent variable (price) of each night for different properties.
Initially, we apply Multi-linear Regression, Random Forest and Gradient Boosting Regression models on the datasets and evaluate these models to see their performances respectively.

Linear Regression: in the exploratory analysis part, we do observe some (though not obvious) relation trends among some numeric features and the price. As this model assumes a linear relationship among dependent and independent features, we trained this model to find the best-fitting straight line to predict the rental price.

Random Forest Regression: an ensemble learning method that builds multiple decision trees and averages their predictions to improve accuracy and reduce overfitting. It's one of the most widely used methods and we thus trained the model to handle the high dimensionality of our dataset.

Gradient Boosted Regression: an ensemble learning method that builds a predictive model in a step-by-step manner. Different from the random-forest method, this model belongs to a stacking ensemble model. Therefore, we will use this model to make comparisons between different ensemble methods.

In evaluating our models, we analyzed the Prediction Error Curve and Residual Error among the three models (Figure 7- Figure9).

- As for the linear regression model, the fitting level seemed not as well as the other two models in the training set but it performed well in the testing set. However, the linear regression model seemed a little bit more "conservative" than the other two, since the range of the predicted price was more narrow as shown in the prediction error curve.
- Intuitively, the random-forest model showed best fit in the training set, however, its performance in the testing set was compromised, compared to the other two models. In this way, we think that an overfitting problem might occur when training with a random-forest regression model.

- Gradient boosted models performed very similarly in predicting in both training and testing sets. It's not as "conservative" as the linear regression model in predicting the rentals with potentially higher prices. And its performance seemed to be more consistent between training and testing sets than the random-forest model.

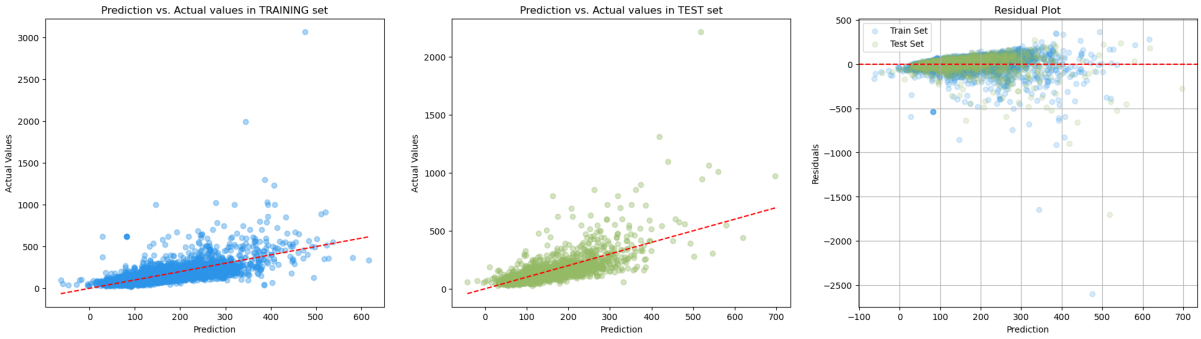


Figure 7. Prediction Error Curve and Residual Error Analysis for Linear Regression

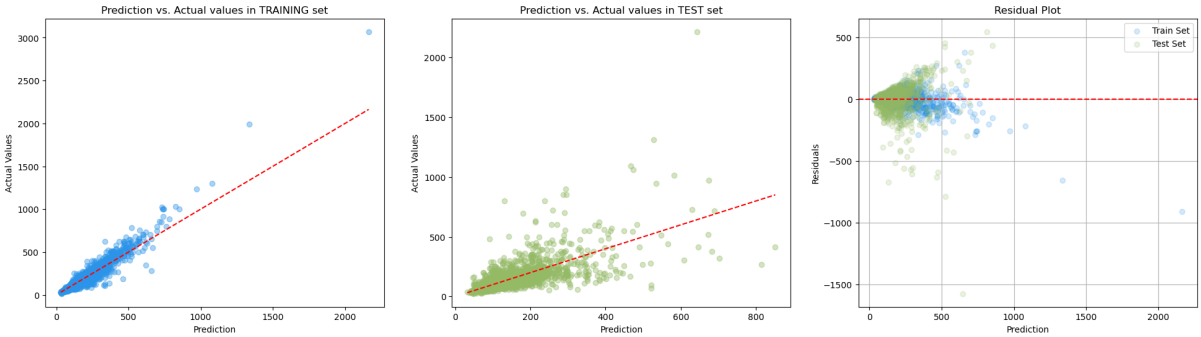


Figure 8. Prediction Error Curve and Residual Error Analysis for Random Forest Regression

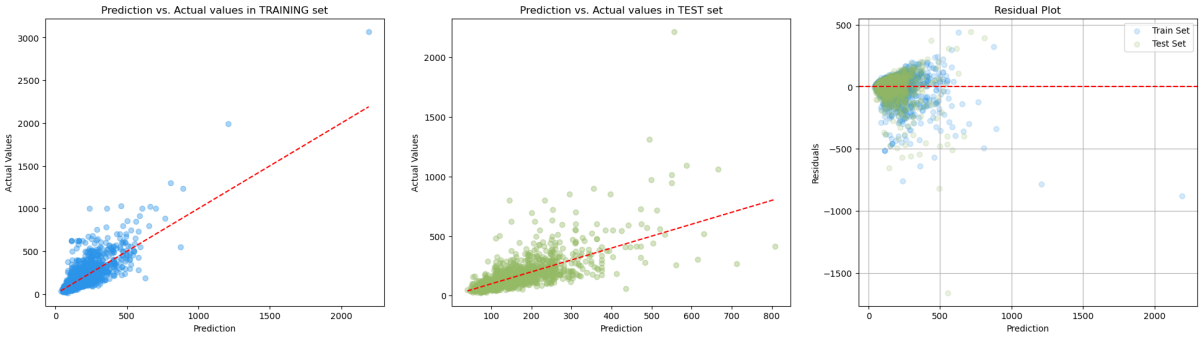


Figure 9. Prediction Error Curve and Residual Error Analysis for Gradient Boosting Regression

Also, mean squared error (MSE), mean absolute error (MAE) and R-squared scores were used to further evaluate these regression models. It turned out that the Gradient Boosted Regression model had the minimum MAE and the largest R-square. Random Forest model has the highest MAE value.

	Model	RMSE	MAE	R2
0	Linear Regression	68.688701	46.481987	0.533314
1	Random Forest	70.291394	44.894421	0.511282
2	Gradient Boosting	68.072180	45.279054	0.541654

3. Model Optimization

Mutual Information (MI): We attempted to enhance the prediction accuracy by altering the dataset, so we utilized MI to identify the most significant features. Subsequently, we applied the three models to the columns with high MI scores. However, we found that the performance did not improve as expected, so we decided to continue using the original datasets.

	Model	RMSE	MAE	R2
0	Linear Regression	73.185889	49.468502	0.470204
1	Random Forest	73.076608	47.598107	0.471785
2	Gradient Boosting	68.983334	46.367501	0.529302

Random Search & 5-fold Cross-Validation: First, we define the range of the parameters for random forest (number of estimators, max of depth and minimum number of samples required to split a node) and gradient boosting (number of estimators, max of depth, learning rate, min of split node and min of split leaf). And we use random search to pick a set of parameters from the range for each iteration. Then we use 5-fold cross-validation to train and test the model five times to find the best parameter of the prediction model.

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best parameters: {'max_depth': 10, 'min_samples_split': 13, 'n_estimators': 444}
RMSE: 68.72959052576944
MAE: 45.34864180618583
R-squared: 0.5327583825577196
```

	Model	RMSE	MAE	R2
0	Random Forest Before Optimization	70.291394	44.894421	0.511282
1	Random Forest After Optimization	68.729591	45.348642	0.532758

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best parameters: {'learning_rate': 0.05, 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 8, 'n_estimators': 343}
RMSE: 67.43080705405146
MAE: 44.17175857869453
R-squared: 0.550250467180026
```

	Model	RMSE	MAE	R2
0	Gradient Boosting Before Optimization	68.072180	45.279054	0.541654
1	Gradient Boosting After Optimization	67.430807	44.171759	0.550250

Stacking: We stack all three models from above (linear regression, optimized random forest, optimized gradient boosting) to get a stacking model for better prediction performance. In the stacking model, the predictions from the base models will be input to a meta-model, which learns how to best combine these predictions.

```
Stacking Model Evaluation Metrics:
RMSE: 67.27960349946956
MAE: 44.13780615812178
R-squared: 0.5522651986322878
```

Bagging: And we use the bagging method to train multiple instances of the stacking model on different data subsets and get their predictions together. After this final optimization we got a better prediction performance model.

```
Bagging Stacked Model Evaluation Metrics:
RMSE: 66.38752719905041
MAE: 44.00577740079706
R-squared: 0.5640597284581361
```

Observation and Conclusion

1. **Crime Affects Pricing**: based on the MI scores, we found that crime rate is an important factor influencing airbnb price. Consequently, we strongly advise listing owners or potential listing owners to prioritize safety in their area.
2. **Listing Accuracy Matters**: a correlation matrix analysis (Table 1.) is conducted to examine the relationship between overall rating scores and individual review criteria, aiming to provide informative advice to listing owners. The results indicate that customers highly value accuracy and good value. Therefore, we recommend that

owners provide detailed property descriptions and photographs on the listing, and ensure the pricing to be reasonable.

3. **Cleanliness and Communication are Highly-valued**: cleanliness and communication are also significant factors to customers. Therefore, we advise listing owners to maintain their properties tidy and clean, and respond promptly to enquiries and feedback from their guests.

	Rating	accuracy	cleanliness	checkin	communication	location	value
Rating	1.00	0.85	0.77	0.67	0.77	0.62	0.84
accuracy	0.85	1.00	0.70	0.60	0.69	0.58	0.78
cleanliness	0.77	0.70	1.00	0.50	0.59	0.47	0.66
checkin	0.67	0.60	0.50	1.00	0.68	0.49	0.61
communication	0.77	0.69	0.59	0.68	1.00	0.49	0.69
location	0.62	0.58	0.47	0.49	0.49	1.00	0.60
value	0.84	0.78	0.66	0.61	0.69	0.60	1.00

Table 1. Correlation Matrix

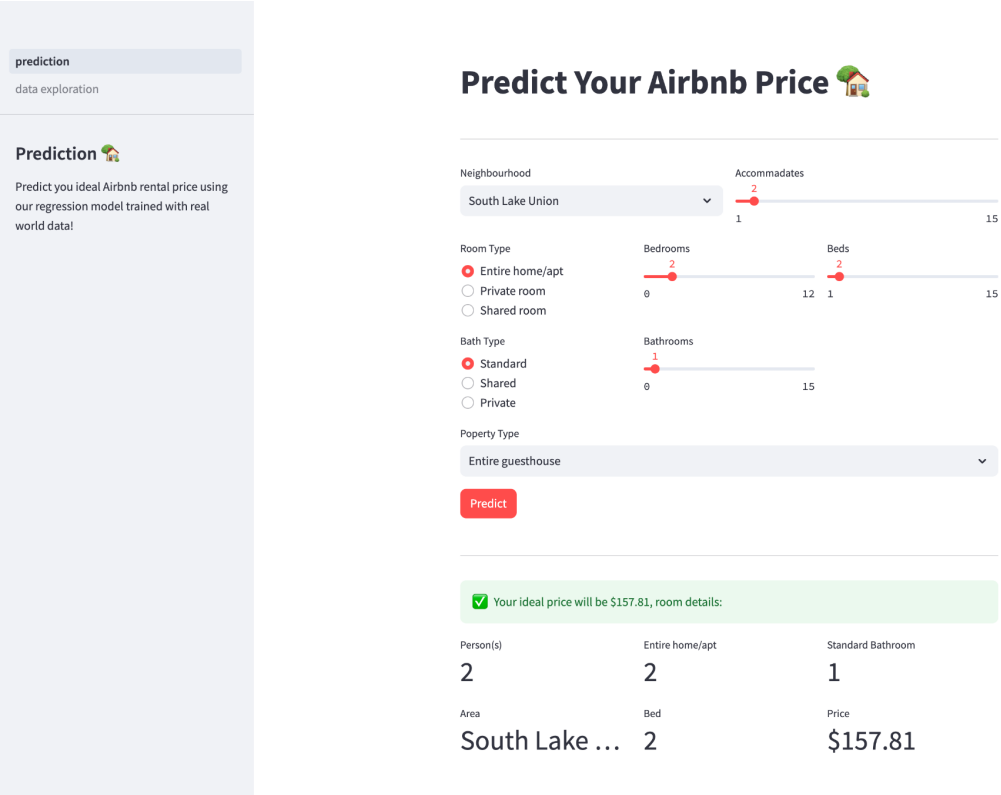
Online Application and Docker Service

For better user experience, we provide an online UI application using Streamlit. In our online application, users will be asked to give information about their preferences on 9 features (including neighborhood, room type, property type, bathroom type, accommodates, bedroom number, bathroom number, beds number, and review rating scores). Here, "property type" refers to the type of property, such as a house, apartment, condo, or boat. "Room type" indicates the type of room included in the listing. Our model will take user inputs and predict the ideal price of selected homestay. We have optimized data processing and running models. All these processes are running online and will be done within a second. Furthermore, we also provide a dockerized service to give a universal access to our model and service, regardless of the OS and Python environment. The details of our service are described below.

Streamlit Server

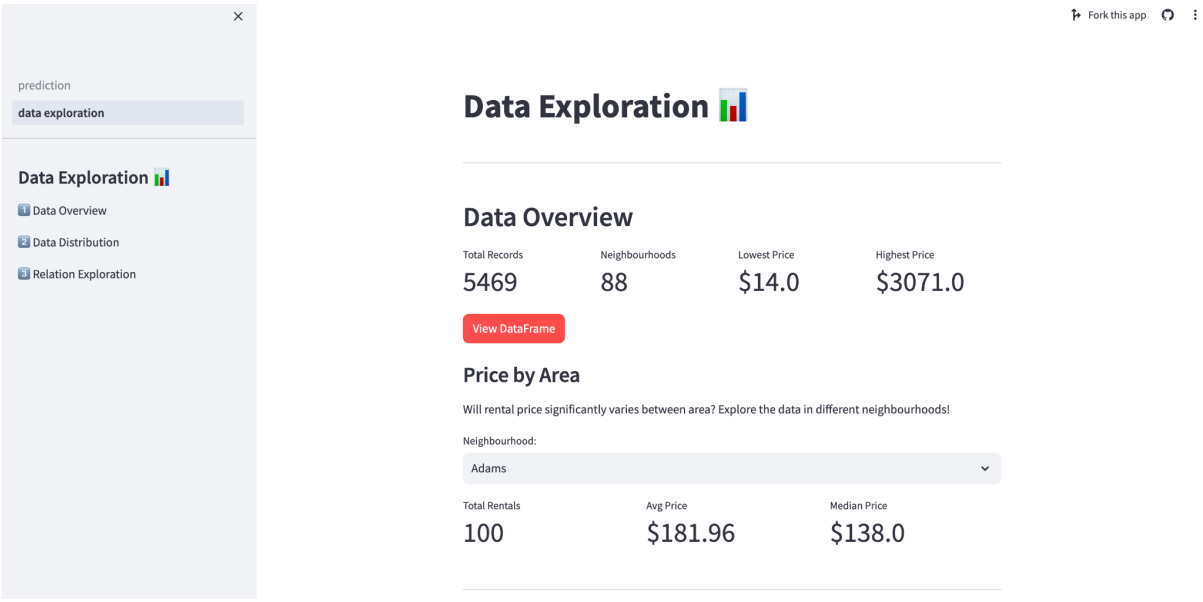
In the online Streamlit server we provided, a prediction page and data exploration page are provided to predict the price and look through the raw data we used in training our model. In the prediction page, users will provide all 9 features in predicting the ideal price of the homestay they are looking for. Categorical features are presented using either Selectbox or Radio components provided by Streamlit API. While numeric features are presented using Slider for better user experience. The Slider component limits the values selected by users, thus saving the server from data validation. This design ensures a user-friendly experience and data validation at the front-end level. By clicking the Predict button, Streamlit will collect

the user input values and the model predicts the price based on user inputs, which are encoded and matched to the model's expected data structure. Finally, the predicted price and a detailed breakdown of the user-provided data are displayed dynamically using Streamlit's Metrics component, offering users immediate and clear insights into the potential rental price.



Overview of the prediction page can be viewed in our online Streamlit application.

In the data exploration page of our Streamlit app, we show a full view of Airbnb listing data, including total records and price ranges. Users can look at detailed stats for specific neighborhoods like average and median rental prices, and see the spread of property features (beds, rooms) through bar charts. The page has interactive scatter plots to check how different features impact rental prices, helping users understand market trends. Also, we provide maps showing average Airbnb prices and crime rates across Seattle neighborhoods, letting users check trends and safety by location. This rich data display and interactive tools help users deeply understand the Airbnb market in Seattle.



Overview of the data exploration page can be viewed in our online Streamlit application.

Docker Service

A docker image is provided for better compatibility and convenience in applying our prediction model. A step-by-step instruction is provided here to build the docker image as well as run docker container from the image.

1. Make sure the docker daemon is running
2. Build docker image

- Navigate to the working directory of the docker project (where the `Dockerfile` locates)
- The orgnization of the dokcer service folder looks like:

  ◊ Dockerfile - contains the instructions for building the docker image.
  ◊ data folder - contains dataset used for real-time statistical analysis and request generation.
  ◊ model folder - includes the trained model to provide prediction function.
  ◊ pages folder - contains extra Streamlit pages for the online application.
  ◊ prediction.py - is the main page of the online Streamlit application.
  ◊ requirements.txt - lists the required Python packages in building the docker image.

```
./docker-project
├── Dockerfile
├── data
│   ├── default_input.csv
│   └── listing_primary.csv
├── model
│   └── model.joblib
├── pages
│   └── data_exploration.py
├── prediction.py
└── requirements.txt

3 directories, 7 files
```

- Run the following instruction to build the docker image:

```
docker build . -t cs6220/docker-project
```

- It will take a while for the docker image to be built. The name of our docker

```
docker run -d -p 8501:8501 --name airbnb-prediction cs6220/docker-project:latest
```

3. Run docker container from the image we just built using the following command

```
docker run -d -p 8501:8501 --name airbnb-prediction cs6220/docker-project:latest
```

- Navigate to `localhost:8501` to access the Streamlit application of the docker service.
- Log trace example when running docker container for the applicaiton may look like:

```
✗ docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
cs6220/docker-project   latest    ce65b277909d   20 seconds ago   1.06GB

✗ docker run -d -p 8501:8501 --name airbnb-prediction cs6220/docker-project:latest
2b023d5c36f1618bc6dc84f2f158979846a163bb65adfdd96fe2b578a7524679

✗ docker ps
CONTAINER ID   IMAGE                           COMMAND                  CREATED         STATUS         PORTS
2b023d5c36f1   cs6220/docker-project:latest   "streamlit run predi…"   4 seconds ago   Up 3 seconds   0.0.0.0:850

✗ docker stop airbnb-prediction
airbnb-prediction
```

Reference

- Dataset for training the model was dirived from Seattle Airbnb Activity
- scikit-learn:
    - LinearRegression
    - RandomForestRegressor
    - GradientBoostingRegressor
- Streamlit
- Docker
- GitHub repository of our project
- Online Streamlit application