Predicting Nightly Rates for Airbnb Listings in Boston

MGMT 68300

Group 5

PURDUE UNIVERSITY

Our Team

Prashanth Chari

Yukti Sanjay Jain

Ashvin Raj

Akshita Sharma

Business Problem Overview

Dynamic Market Challenges

Hosts in Boston might face difficulties setting optimal nightly rates due to fluctuating demand and competitive market conditions, risking revenue loss or low occupancy

Data Overwhelm

Hosts often struggle to effectively utilize vast amounts of market and customer behavior data, leading to suboptimal pricing decisions

Evolving Consumer

Expectations: Rapidly changing guest preferences in the post-pandemic era require hosts to adapt quickly, a challenge without sophisticated analytical tools

Objectives

Optimizing Revenue through Accurate Pricing

Competitive Edge in a Dynamic Market

Guest Satisfaction and Reputation Management

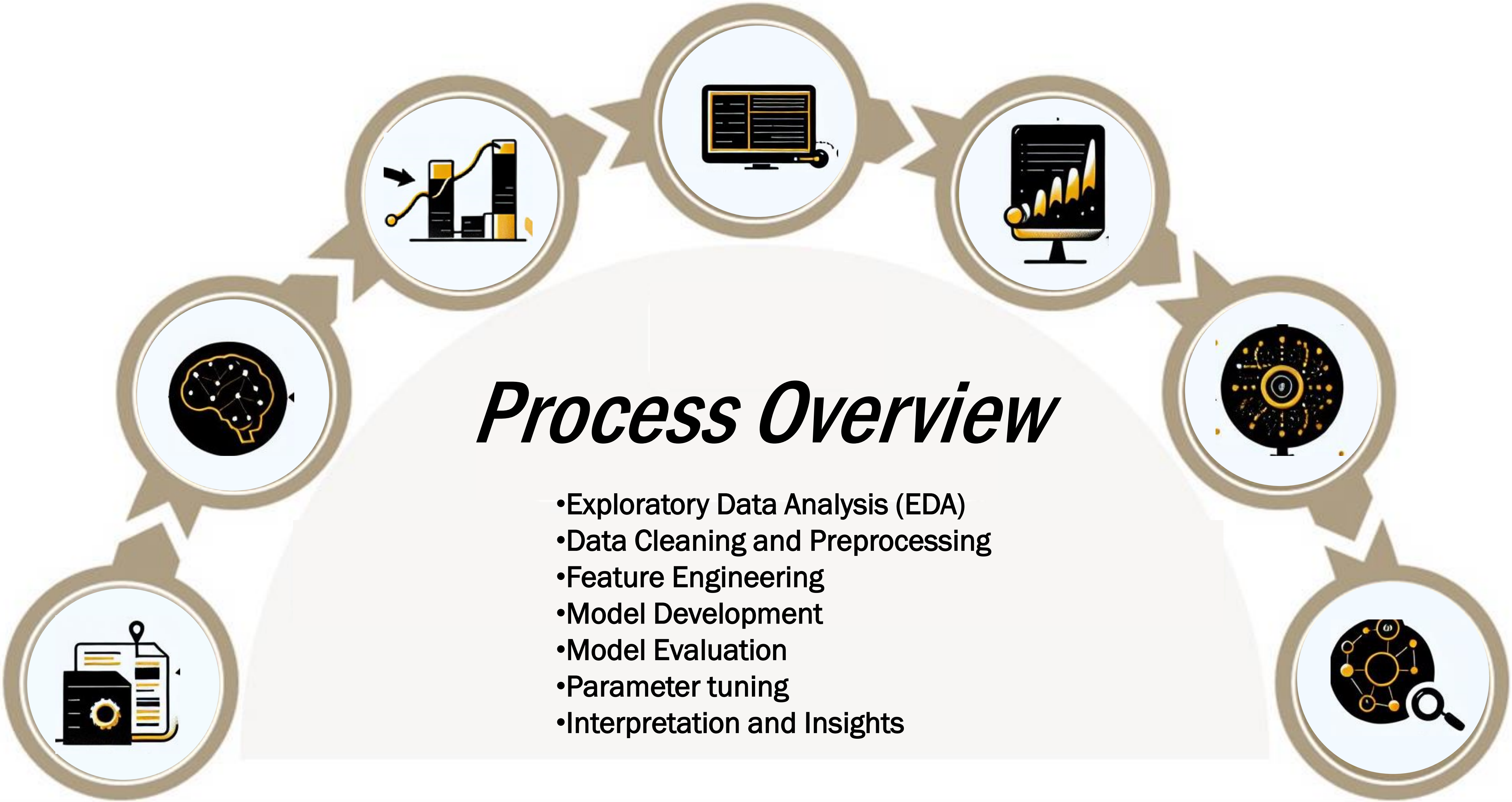
Adaptability to Market Dynamics

Utilizing Technology for Informed Decisions

Dataset Description

- The Airbnb dataset for Boston contains a total of 47,606 records and 111 columns.
- Among these columns are 8 categorical variables and 103 numerical variables, that including various types of data such as IDs, ratings, number of reviews, financial figures, and geographical data
- Detailed information on Airbnb listings and host performance.
  - IDs for hosts and properties.
  - Location details and geographical data.
  - Property characteristics and booking specifics.
  - Host performance metrics and financial data.
  - Indicators: pet allowance, instant booking availability.

PURDUE UNIVERSITY



Process Overview

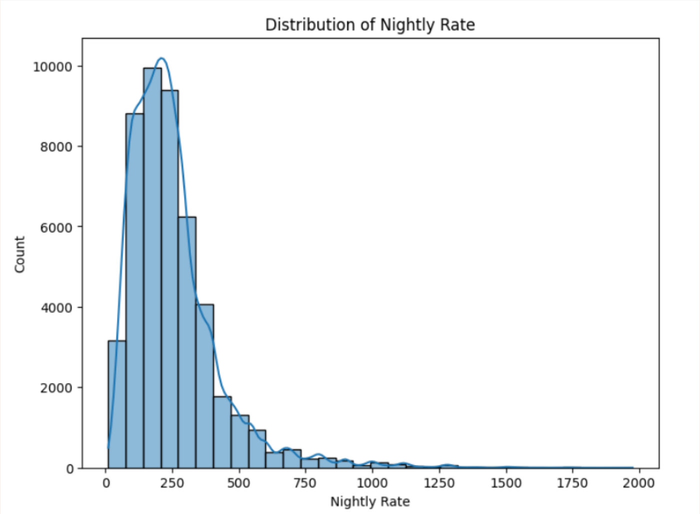
- Exploratory Data Analysis (EDA)
- Data Cleaning and Preprocessing
- Feature Engineering
- Model Development
- Model Evaluation
- Parameter tuning
- Interpretation and Insights

Exploratory Data Analysis (EDA)

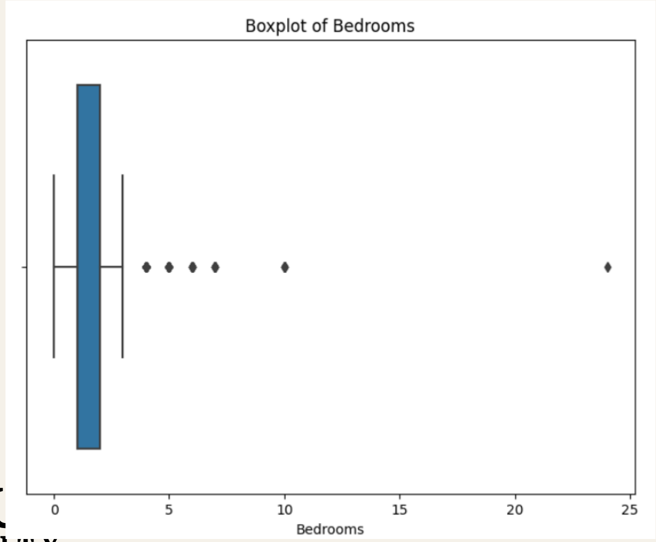
Summary Statistics of the target variable (Nightly Rate)

	Nightly Rate
count	47606.000000
mean	253.504951
std	171.638728
min	10.000000
25%	140.000000
50%	220.000000
75%	309.000000
max	1976.000000

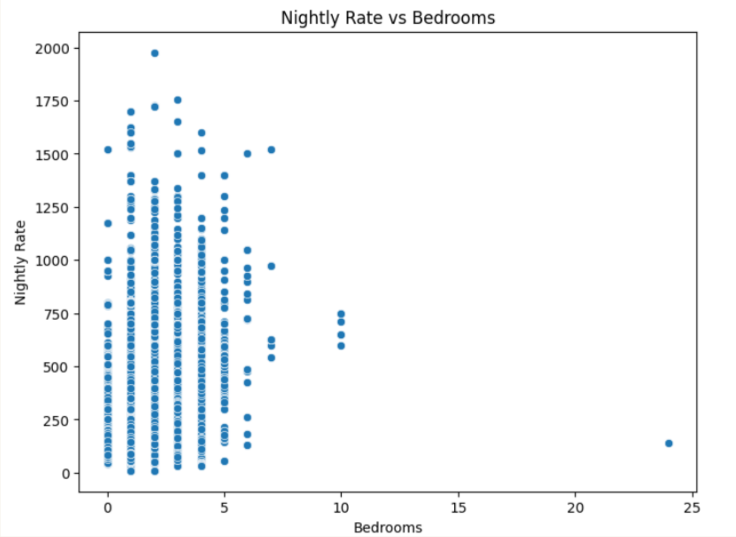
Distribution of Nightly Rate



Boxplot of Bedrooms

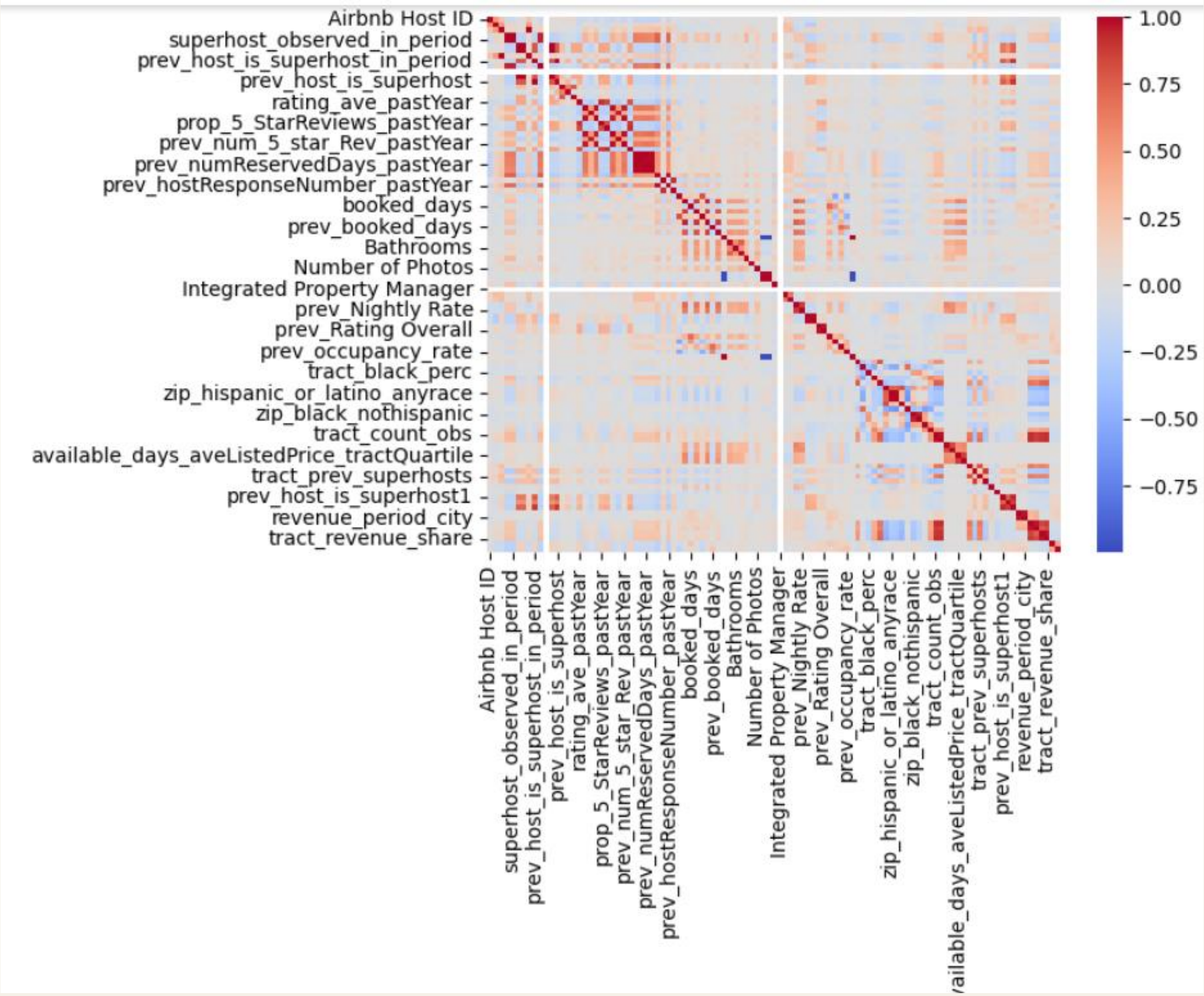


Nightly Rate vs Bedrooms

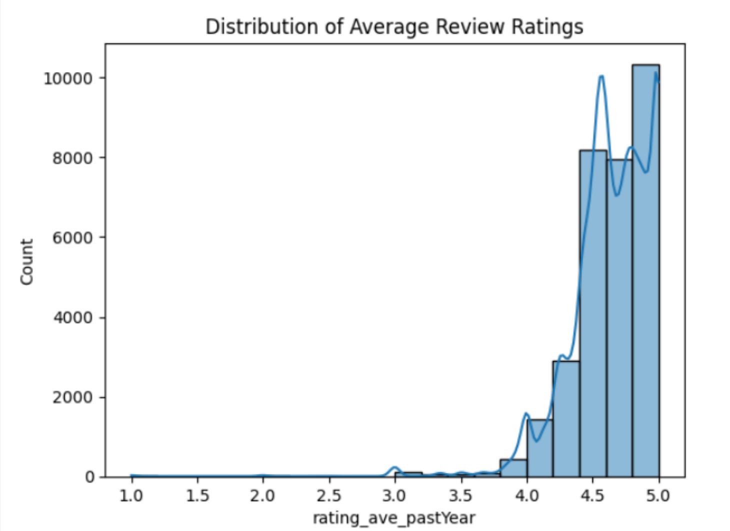


Exploratory Data Analysis (EDA)

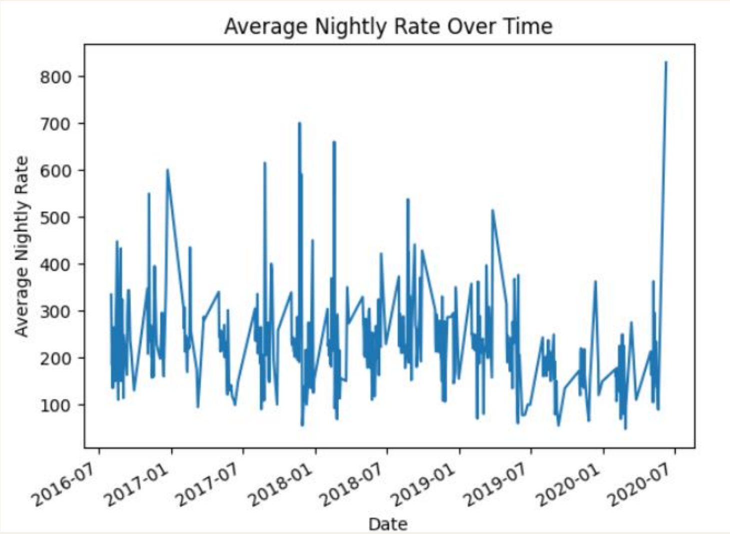
Correlation Matrix



Distribution of Average Review Ratings



Average Nightly Rate



PURDUE UNIVERSITY

Data Cleaning and Preprocessing

Outlier Treatment

Detected data points that significantly skewed the dataset, leading to impractical scenarios.

For instance, a data point featuring 25 bedrooms with a nightly rate lower than that of a one-bedroom apartment.

Handling Missing Values

Dropping Rows : Eliminated rows with missing bedroom data to ensure data integrity.

Imputing Missing Values:

a) Categorical Data: Employed the mode for imputing missing values in categorical data.

b) Geospatial Imputation: Additionally, imputed neighborhood values based on the corresponding zip code.

Preprocessing Relevant Data

Recognized columns highly correlated with the target variable.

Excluded unnecessary columns from the analysis; for instance, prev_nightly rate was highly correlated with nightly rate but omitting it prevented biased model results.

One-hot encoded categorical variables (Property Type, Listing Type, Neighborhood)

PURDUE UNIVERSITY

Feature Engineering

SuperhostRatio_Zipcode: Reflects the concentration of quality hosts in an area, influencing perceived value and price.

Superhost_Status_Frequency: Indicates host quality and reputation, affecting listing desirability and pricing power.

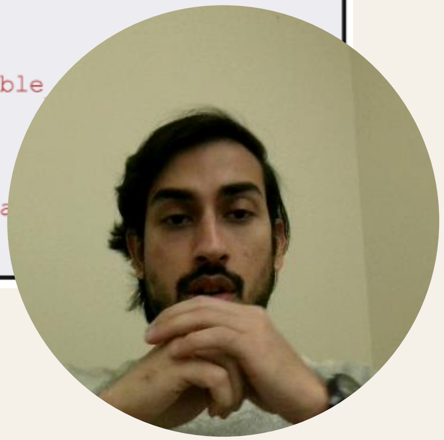
ListingsCount_Zipcode: Measures market saturation, which can impact pricing through supply and demand dynamics.

OccupancyRate_Zipcode: Signals area demand, informing potential rate setting based on local popularity.

Bathroom_Bedroom_Ratio: Suggests property comfort level, potentially justifying higher rates for greater convenience.

Ratio_5_Star_Superhost: Combines guest satisfaction with host status, which can correlate with higher achievable prices.

```
superhost_ratio_zipcode = subset_data.groupby('Zipcode')['host_is_superhost_in_period'].mean().rename('SuperhostRatio_Zipcode')
subset_data['Superhost_Status_Frequency'] = subset_data.groupby('Airbnb Host ID')['host_is_superhost_in_period'].transform('sum')
listings_count_zipcode = subset_data['Zipcode'].value_counts().rename('ListingsCount_Zipcode')
subset_data['OccupancyRate_Zipcode'] = subset_data.groupby('Zipcode')['numReserv_pastYear'].transform('mean') / subset_data['available
subset_data['Bathroom_Bedroom_Ratio'] = subset_data['Bathrooms'] / subset_data['Bedrooms']
subset_data['Ratio_5_Star_Superhost'] = subset_data.apply(
    lambda x: x['num_5_star_Rev_pastYear'] / x['numReviews_pastYear'] if (x['host_is_superhost_in_period'] == 1 and x['numReviews_pa
    axis=1
```



PURDUE UNIVERSITY

Models Chosen and Why?

Our approach optimizes model accuracy for robustness and interpretability for analyzing business impact

Gradient Boosting:

- For selecting significant variables based on feature importance
- For predicting nightly rates based on the selected predictors, that can be used by the hosts for competitive edge
- Higher model accuracy that withstands inconsistencies in real world data

Train-test : Prediction – 85 : 15

Train : Test – 80 : 20

R-squared – 0.76 MAPE ~ 10%

Prediction R-squared: 0.72

Correlation of prediction vs actual night rate 0.9

Lasso Regression

- For better model interpretability that helped us analyze business impact
- Selecting features from gradient boosting, including interaction terms and refining using lasso shrinkage further increased accuracy
- Coefficients gave us the monetary impact of the most significant features filtered by the model

Validation R-squared: 0.68

MAPE ~ 16.83%

PURDUE UNIVERSITY

Gradient Boosting

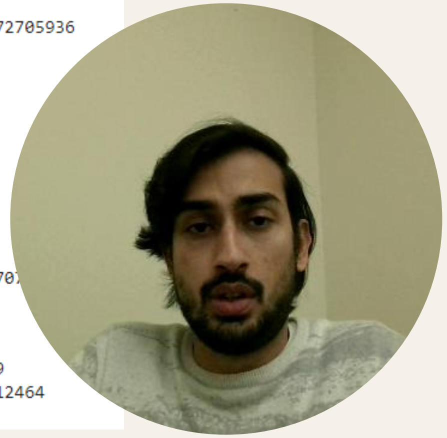
Selecting the most important features

- Excluding variables correlated to pricing or “Nightly Rate” steers the model to assess actionable host-controlled factors, guiding strategy improvements rather than mirroring past prices.
- Omitting direct pricing variables avoids circular logic, ensuring the model's insights reflect genuine market influences on pricing decisions.

```
Feature Importance in the Selected Model:
prev_Nightly Rate: 0.547615110874176
prev_available_days_aveListedPrice: 0.08846108615398407
Quarter_2019Q2: 0.08601415157318115
prev_numReserv_pastYear: 0.06635438650846481
available_days_aveListedPrice: 0.037523627281188965
Cleaning Fee (USD): 0.03171306103467941
Neighborhood_West End: 0.029507147148251534
Scraped Month: 0.027274372056126595
rating_ave_pastYear: 0.02705472521483898
Bathrooms: 0.021449275314807892
Instantbook Enabled: 0.02056662179529667
ListingsCount_Zipcode: 0.014577098190784454
Zipcode_2110.0: 0.0018894386012107134
```

- Feature importance analysis identified key determinants of nightly rates for Airbnb listings in Boston. The top features influencing pricing decisions include:

```
Feature Importance in the Selected Model:
ListingType_Entire home/apt: 0.233995258808136
Bedrooms: 0.13582372665405273
Bathrooms: 0.11631103605031967
Cleaning Fee (USD): 0.07857580482959747
SuperhostRatio_Zipcode: 0.03350621461868286
numReviews_pastYear: 0.025189891457557678
prev_numReserv_pastYear: 0.023794282227754593
Max Guests: 0.0188003983348608
Neighborhood_South End: 0.018769090995192528
Superhost_Status_Frequency: 0.01838596537709236
Bathroom_Bedroom_Ratio: 0.017800912261009216
PropertyType_House: 0.017240483313798904
Neighborhood_Fenway/Kenmore: 0.016919365152716637
Number of Photos: 0.016075685620307922
num_5_star_Rev_pastYear: 0.015307492576539516
rating_ave_pastYear: 0.015287368558347225
ListingsCount_Zipcode: 0.014797091484069824
Pets Allowed: 0.013757587410509586
available_days: 0.013361964374780655
PropertyType_Serviced apartment: 0.013296765275299549
PropertyType_Entire apartment: 0.011352023109793663
Neighborhood_Downtown: 0.010532181710004807
PropertyType_Apartment: 0.010487598367035389
Neighborhood_Beacon Hill: 0.010253398679196835
PropertyType_Room in boutique hotel: 0.010149852372705936
PropertyType_Townhouse: 0.009668545797467232
PropertyType_Condominium: 0.009410306811332703
Instantbook Enabled: 0.009309399873018265
Neighborhood_West End: 0.008953629061579704
OccupancyRate_Zipcode: 0.007971134036779404
Scraped Month: 0.007817287929356098
Neighborhood_North End: 0.007343571167439222
Ratio_5_Star_Superhost: 0.006208427716046572
PropertyType_Loft: 0.0059889075346291065
PropertyType_Private room: 0.004956474993377924
prev_host_is_superhost_in_period: 0.00396058615297
PropertyType_Villa: 0.003432236844673753
PropertyType_Place: 0.002503449795767665
PropertyType_Resort: 0.0018218582263216376
PropertyType_Boutique hotel: 0.000560111424420289
PropertyType_Bed and breakfast: 0.0003226001281291246
PropertyType_Private room in house: 0.0
```



PURDUE UNIVERSITY

Lasso regression

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LassoCV
from sklearn.pipeline import Pipeline

# Assuming 'X_top_25' is your DataFrame with the original predictors, and 'y' is your target variable

# Generate interaction terms
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
X_interactions = poly.fit_transform(X_top_25)

# Create a DataFrame with interaction terms
column_names = poly.get_feature_names_out(input_features=X_top_25.columns)
X_interactions_df = pd.DataFrame(X_interactions, columns=column_names)

# List of interaction terms to remove
terms_to_remove = [
    'ListingType_Entire home/apt Neighborhood_South End',
    'ListingType_Entire home/apt Superhost_Status_Frequency',
    'Bedrooms Neighborhood_South End',
    'Bedrooms PropertyType_Serviced apartment',
    'Bathrooms available_days',
    'Cleaning Fee (USD) PropertyType_Serviced apartment',
    'SuperhostRatio_Zipcode ListingsCount_Zipcode',
    'Neighborhood_South End PropertyType_Serviced apartment',
    'Number of Photos PropertyType_Serviced apartment'
    # Add other terms as needed
]

# Remove these terms from your DataFrame
X_reduced = X_interactions_df.drop(columns=terms_to_remove, errors='ignore')

# Split the data into training and test sets for validation purposes
X_train, X_test, y_train, y_test = train_test_split(X_reduced, y, test_size=0.2, random_state=42)

# Create a pipeline that standardizes the data, then applies Lasso
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('lasso', LassoCV(cv=5, random_state=0, max_iter=10000))
])

# Fit the model on the training data
pipeline.fit(X_train, y_train)

# The best alpha value found by cross-validation
best_alpha = pipeline.named_steps['lasso'].alpha_
print(f"Best alpha found by cross-validation: {best_alpha}")

# Calculate R-squared on the test set
r_squared_test = pipeline.score(X_test, y_test)
```


```python
    'Bedrooms PropertyType_Serviced apartment',
    'Bathrooms available_days',
    'Cleaning Fee (USD) PropertyType_Serviced apartment',
    'SuperhostRatio_Zipcode ListingsCount_Zipcode',
    'Neighborhood_South End PropertyType_Serviced apartment',
    'Number of Photos PropertyType_Serviced apartment'
    # Add other terms as needed
]

# Remove these terms from your DataFrame
X_reduced = X_interactions_df.drop(columns=terms_to_remove, errors='ignore')

# Split the data into training and test sets for validation purposes
X_train, X_test, y_train, y_test = train_test_split(X_reduced, y, test_size=0.2, random_state=42)

# Create a pipeline that standardizes the data, then applies Lasso
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('lasso', LassoCV(cv=5, random_state=0, max_iter=10000))
])

# Fit the model on the training data
pipeline.fit(X_train, y_train)

# The best alpha value found by cross-validation
best_alpha = pipeline.named_steps['lasso'].alpha_
print(f"Best alpha found by cross-validation: {best_alpha}")

# Calculate R-squared on the test set
r_squared_test = pipeline.score(X_test, y_test)
```


```python
print(f"R-squared on the test set: {r_squared_test}")

mape = (mean_absolute_error(y_test, y_pred) / y_test).mean() * 100
print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")


# Extract and display the coefficients
lasso_coefs = pipeline.named_steps['lasso'].coef_
feature_names = X_reduced.columns
lasso_coefficients = pd.Series(lasso_coefs, index=feature_names)
print("Lasso coefficients:")
print(lasso_coefficients)
lasso_coefficients.to_csv('lasso_coefficients.csv',header=True)

non_zero_coefs = lasso_coefficients[lasso_coefficients != 0]
print("Non-zero Lasso coefficients:")
print(non_zero_coefs)
```

```
Best alpha found by cross-validation: 0.09675834192763925
R-squared on the test set: 0.6802052922027048
Mean Absolute Percentage Error (MAPE): 16.83%
```

PURDUE UNIVERSITY

Pricing Analysis

Nightly Rates:

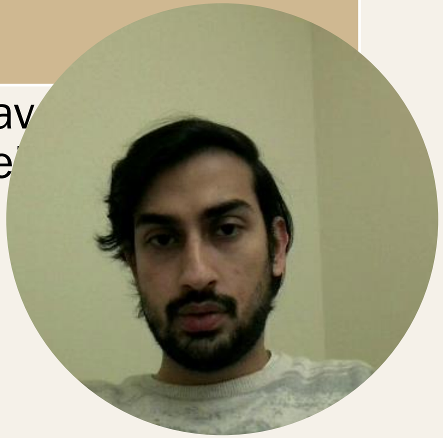
- The difference between predicated nightly rates and actual nightly rates was calculated for each neighborhood, revealing variations across regions.

Overpriced Neighborhoods:

- The analysis identified several neighborhoods where the average nightly rates exceed the overall average. Notable overpriced neighborhoods include **Back Bay**.

Underpriced Neighborhoods:

- Conversely, some neighborhoods display lower av rates, indicating potential opportunities for trave **falling** into this category.



Recommendations:

Pricing Adjustments:

- Property owners in overpriced neighborhoods might consider adjusting their rates during high-demand seasons to remain competitive.

Marketing Strategies:

- Owners in underpriced neighborhoods could leverage their pricing advantage in marketing campaigns to attract cost-conscious travelers.

Limitations:

1. **Data Scope :** The analysis is based on available data and assumes that nightly rates accurately reflect property value.
2. **External Factors :** External factors such as local events, festivals, or economic conditions were not considered in this analysis.

Key Take-aways from Lasso Model

Entire homes/apt attract higher rates (+$19.22).

House listings tend to be priced lower (-$24.04).

Consider bathroom-bedroom ratio for optimal pricing (-$30.46).

Quality photos positively influence rates (+$23.47).

Interaction Terms:

•More guests in entire home/apt listings reduce rates.(-$18.51)

•Cleaning fee impact amplified by the number of bedrooms.($40.29)

•Higher super host ratios positively affect rates with increased guests, basically , in the areas having more super hosts, having more guest increases the nightly rate. ($21.21)

Conclusion

Impact for Stakeholders

Hosts/Property Owners

- Hosts can set competitive prices that increase occupancy rates and revenue, leading to a higher return on investment.

Airbnb Platform

- Accurate pricing maximizes bookings, directly impacting Airbnb's commission-based revenue and enhances customer satisfaction

Customers

- Customers benefit from clear and reasonable pricing, enhancing trust and engagement with the Airbnb platform.

Business Insights

Location-Based Pricing

- Properties in sought-after areas such as Beacon Hill and Fenway/Kenmore command higher rates, reflecting their high demand.

Influence of Features on Pricing

- Factors like instant booking availability and the number of photos listed significantly affect nightly rates

Dynamic and Responsive

- The model is capable of adjusting to seasonal trends, major events, and changing travel patterns, facilitating flexible pricing strategies

Extra Slides

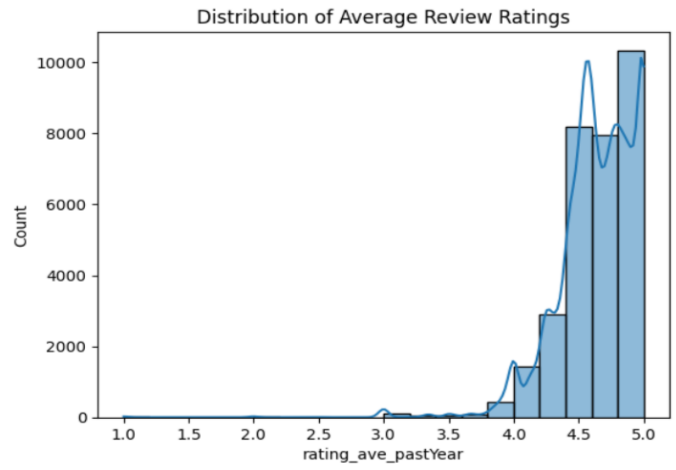
EDA (Extra)

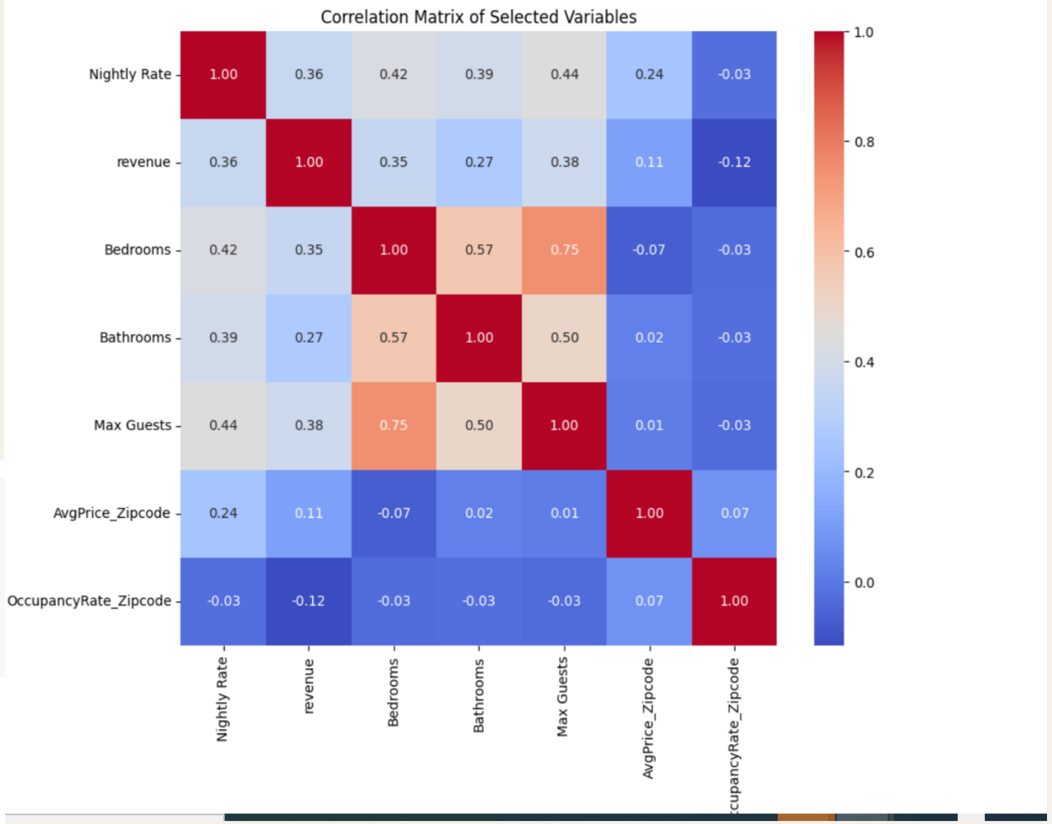
Categorical Variables Summary:

	Property Type	Listing Type	City_x
Apartment	36059.0	NaN	NaN
Bed & Breakfast	15.0	NaN	NaN
Bed & Breakfast	3.0	NaN	NaN
Bed and breakfast	485.0	NaN	NaN
Boat	57.0	NaN	NaN
Boston	NaN	NaN	47606.0
Boutique hotel	2.0	NaN	NaN
Bungalow	6.0	NaN	NaN
Bus	2.0	NaN	NaN
Cabin	4.0	NaN	NaN
Camper/RV	2.0	NaN	NaN
Castle	16.0	NaN	NaN
Chalet	16.0	NaN	NaN
Condominium	4092.0	NaN	NaN
Cottage	1.0	NaN	NaN

[122]
```
#sns.boxplot(data=subset_data, x='Property Type', y='Nightly Rate')
#plt.xticks(rotation=45)
#plt.title('Boxplot of Nightly Rates by Property Type')
#plt.show()

sns.histplot(subset_data['rating_ave_pastYear'], bins=20, kde=True)
plt.title('Distribution of Average Review Ratings')
plt.show()
```





R^2 and MAPE score for the gradient boosting model

```python
import xgboost as xgb
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from math import sqrt
import numpy as np

# Define the MAPE function
def mean_absolute_percentage_error(y_true, y_pred):
    """Calculate the mean absolute percentage error from y_true and y_pred"""
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    # Avoid division by zero
    y_true = np.where(y_true == 0, np.nan, y_true)
    mape = np.mean(np.abs((y_true - y_pred) / y_true)) * 100
    return np.nanmean(mape)

# Predicting on the test set
y_pred = model_selected.predict(X_test)

# Calculate R-squared
r2 = r2_score(y_test, y_pred)
print(f"R-squared: {r2}")


# Calculate MAPE
mape = mean_absolute_percentage_error(y_test, y_pred)
print(f"Mean Absolute Percentage Error (MAPE): {mape}%")


# Optional: Feature Importance
print("\nFeature Importance in the Selected Model:")
for idx in sorted_idx_selected:
    print(f"{X_train.columns[idx]}: {feature_importance_selected[idx]}")
```

```
R-squared: 0.7678296257408863
Mean Absolute Percentage Error (MAPE): 10.728018848757856%

Feature Importance in the Selected Model:
ListingType_Entire home/apt: 0.233995258808136
Bedrooms: 0.13582372665405273
Bathrooms: 0.11631103605031967
Cleaning Fee (USD): 0.07857580482959747
SuperhostRatio_Zipcode: 0.03350621461868286
numReviews_pastYear: 0.025189891457557678
prev_numReserv_pastYear: 0.023794282227754593
Max Guests: 0.0188003983348608
Neighborhood_South End: 0.018769090995192528
Superhost_Status_Frequency: 0.018385965377709236
Bathroom_Bedroom_Ratio: 0.017800912261009216
PropertyType_House: 0.017240483313798904
Neighborhood_Fenway/Kenmore: 0.016919365152716637
Number of Photos: 0.016075685620307922
num_5_star_Rev_pastYear: 0.015307492576539516
rating_ave_pastYear: 0.015287368558347225
ListingsCount_Zipcode: 0.014797091484069824
```

Data pre-processing: imputing 'Neighborhood'

Mapping Neighborhood and Zipcode

[ ]

```
import pandas as pd

# Assuming 'subset_data' is your DataFrame

# Step 1: Create a mapping from Zipcode to Neighborhood
zipcode_to_neighborhood = subset_data.dropna(subset=['Neighborhood', 'Zipcode'])
zipcode_to_neighborhood = zipcode_to_neighborhood.groupby('Zipcode')['Neighborhood'].agg(pd.Series.mode)

# Step 2: Fill missing Neighborhood values based on Zipcode
def fill_neighborhood(row):
    if pd.isna(row['Neighborhood']) and row['Zipcode'] in zipcode_to_neighborhood:
        return zipcode_to_neighborhood[row['Zipcode']]
    else:
        return row['Neighborhood']

subset_data['Neighborhood'] = subset_data.apply(fill_neighborhood, axis=1)

# Check results
print(subset_data[['Zipcode', 'Neighborhood']].head())
```

```
  Zipcode Neighborhood
0  2128.0  East Boston
1  2128.0  East Boston
2  2128.0  East Boston
3  2128.0  East Boston
4  2128.0  East Boston
```

PURDUE UNIVERSITY

Data Splitting for Model Training and Testing and Prediction

```
model_building_set, prediction_set = train_test_split(subset_data, test_size=0.15, random_state=42)
X = model_building_set[selected_features]
y = model_building_set['Nightly Rate']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # 0.2 here is 20% of 85%
# Train the XGBoost model
model_selected = xgb.XGBRegressor()
model_selected.fit(X_train, y_train)

# Evaluate the model using X_test and y_test, as before
# ...
```

Nightly rate vs Predicted Nightly Rate

```
[184] import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split

# Assuming subset_data is your main DataFrame and selected_features contains the feature names used in the model
selected_features = [

    'Scraped Month',
    'Instantbook Enabled', 'ListingsCount_Zipcode', 'Cleaning Fee (USD)',
    'rating_ave_pastYear',
    'Bathrooms', 'prev_numReserv_pastYear', 'Neighborhood_West End', 'PropertyType_Loft','num_5_star_Rev_pastYear','prev_host
    'Max Guests', 'Number of Photos', 'numReviews_pastYear', 'Bathroom_Bedroom_Ratio', 'Bedrooms', 'PropertyType_Serviced apa
    # Add other important features as needed
]
# Splitting the data into features (X) and target (y)
X = subset_data[selected_features]
y = subset_data['Nightly Rate']

# Training the model on the entire dataset
model = xgb.XGBRegressor()
model.fit(X, y)

# Generating predictions for the entire dataset
subset_data['Predicted Nightly Rate'] = model.predict(X)

# Now, subset_data includes a new column with predicted nightly rates
print(subset_data[['Nightly Rate', 'Predicted Nightly Rate']].head())
```

```
   Nightly Rate  Predicted Nightly Rate
0    159.000000              196.936356
1    169.666667              188.439224
2    193.750000              188.792007
3    200.000000              206.698532
4    200.000000              190.301590
```

```
False
```

```
# Calculating the Pearson correlation coefficient between actual and predicted nightly rates
correlation = subset_data['Nightly Rate'].corr(subset_data['Predicted Nightly Rate'])

print(f"Correlation between actual and predicted nightly rates: {correlation}")
```

```
Correlation between actual and predicted nightly rates: 0.9071222311071525
```

PURDUE UNIVERSITY

Over priced vs Under priced

```python
# Create a new column for price difference
subset_data['Price Difference'] = subset_data['Predicted Nightly Rate'] - subset_data['Nightly Rate']

# Categorize listings based on price difference
conditions = [
    (subset_data['Price Difference'] > 0),  # Charging less than predicted
    (subset_data['Price Difference'] < 0)   # Charging more than predicted
]
choices = ['Underpriced', 'Overpriced']
subset_data['Pricing Category'] = np.select(conditions, choices, default='Accurately Priced')

# View the first few rows
print(subset_data[['Nightly Rate', 'Predicted Nightly Rate', 'Price Difference', 'Pricing Category']].head())
```

```
   Nightly Rate  Predicted Nightly Rate  Price Difference Pricing Category
0    159.000000              196.936356         37.936356      Underpriced
1    169.666667              188.439224         18.772558      Underpriced
2    193.750000              188.792007         -4.957993       Overpriced
3    200.000000              206.698532          6.698532      Underpriced
4    200.000000              190.301590         -9.698410       Overpriced
```

PURDUE UNIVERSITY

Downtown vs Back Bay

- https://www.fodors.com/community/united-states/best-location-to-stay-when-visiting-boston-back-bay-or-downtown-927274/

```python
# Assuming 'nightly_rate' is the column name for the nightly rate in your DataFrame
# and 'Neighborhood' is the column for neighborhood names

# Group by 'Neighborhood' and calculate the mean of 'nightly_rate'
average_rates = subset_data.groupby('Neighborhood')['Nightly Rate'].mean()

# Display the average nightly rate for each neighborhood
print(average_rates)
```

```
Neighborhood
Back Bay          276.132634
Beacon Hill       239.306306
Brookline         249.811458
Downtown          263.250962
East Boston       188.727573
Fenway/Kenmore    268.410087
North End         241.834864
South Boston      257.977901
South End         241.387673
West End          302.902623
Name: Nightly Rate, dtype: float64
```

```python
[201]
# Group by 'Neighborhood' and calculate the mean of 'nightly_rate'
average_rates = subset_data.groupby('Neighborhood')['Nightly Rate'].mean()

# Display the average nightly rate for each neighborhood
print(average_rates)
```

```
Neighborhood
Back Bay          276.132634
Beacon Hill       239.306306
Brookline         249.811458
Downtown          263.250962
East Boston       188.727573
Fenway/Kenmore    268.410087
North End         241.834864
South Boston      257.977901
South End         241.387673
West End          302.902623
Name: Nightly Rate, dtype: float64
```

```python
[205] # Analyzing common features for 'Overpriced' listings
overpriced_listings = subset_data[subset_data['Pricing Category'] == 'Overpriced']

overpriced_analysis = {
    'Most Common Neighborhood': overpriced_listings['Neighborhood'].mode()[0],

}

# Output the analysis for overpriced listings
for feature, value in overpriced_analysis.items():
    print(f"{feature}: {value}")
```

```
Most Common Neighborhood: Back Bay
```

```python
data.reset_index(drop=True, inplace=True)
subset_data.reset_index(drop=True, inplace=True)

# Adding the original 'Property Type' column back to the subset_data
subset_data['Property Type'] = data['Property Type']

# Ensure indices are aligned between 'data' and 'subset_data'
data.reset_index(drop=True, inplace=True)
subset_data.reset_index(drop=True, inplace=True)

# Adding the original 'Neighborhood' column back to the subset_data
subset_data['Neighborhood'] = data['Neighborhood']

# Example: Analyzing common features for 'Underpriced' listings
underpriced_listings = subset_data[subset_data['Pricing Category'] == 'Underpriced']

# Calculate mean, median for numerical features and mode for categorical features
underpriced_analysis = {
    'Most Common Neighborhood': underpriced_listings['Neighborhood'].mode()[0],

}

# Output the analysis for underpriced listings
for feature, value in underpriced_analysis.items():
    print(f"{feature}: {value}")
```

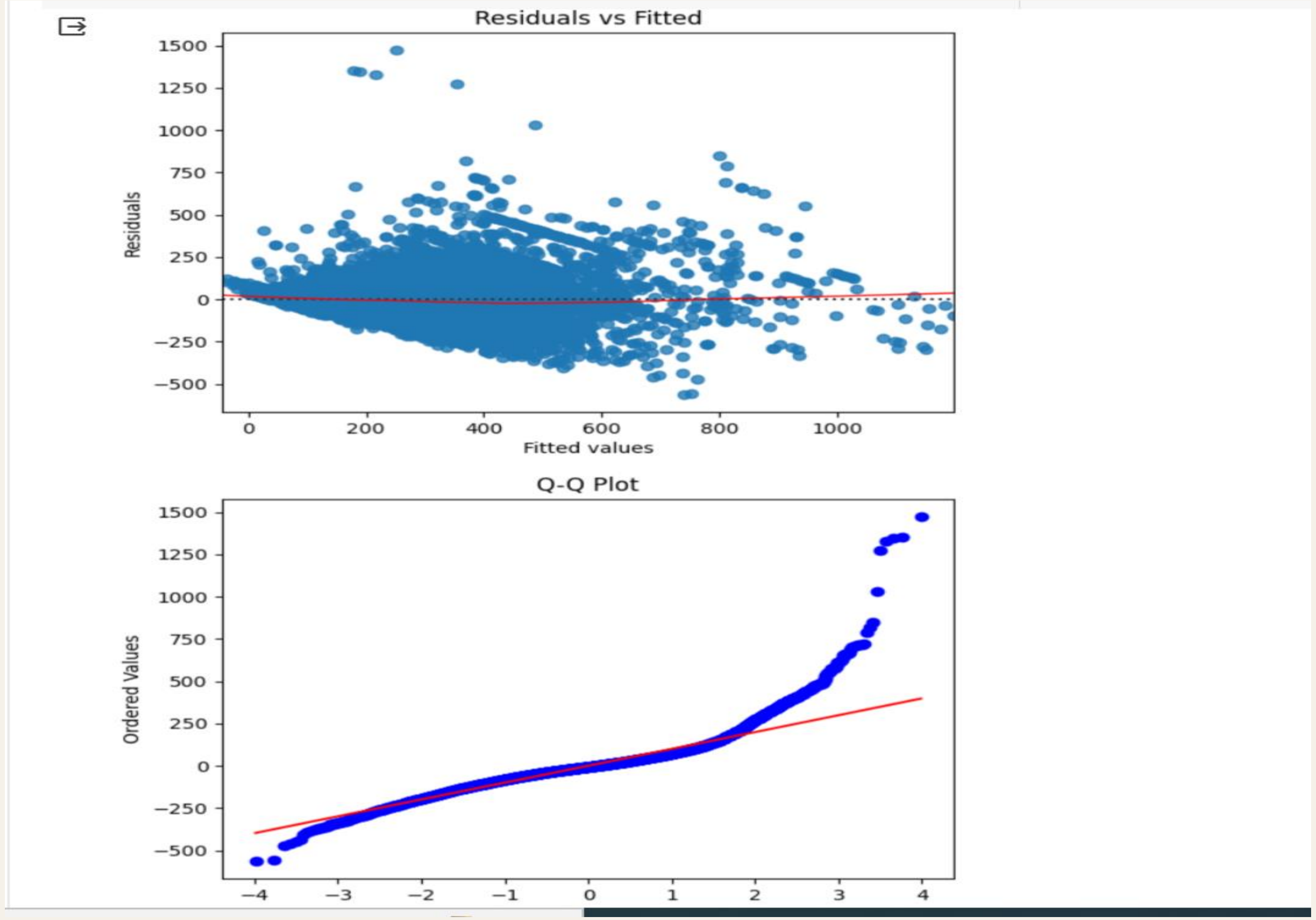
```
Most Common Neighborhood: Downtown
```

Backward Selection Model Tried

```
Number of interaction features: 277
Number of names in interaction_feature_names: 277
All feature names are unique
Shape of final DataFrame after renaming: (23035, 277)
First few feature names: ['Intercept', 'ListingType_Entire home/apt', 'ListingType_Entire home/apt_x_Bedrooms', 'ListingType_Entire home/apt_x_Bathrooms', 'ListingType_Entire home/apt_x_Cleaning Fee (USD)']
First few columns of DataFrame: Index(['Intercept', 'ListingType_Entire home/apt',
       'ListingType_Entire home/apt_x_Bedrooms',
       'ListingType_Entire home/apt_x_Bathrooms',
       'ListingType_Entire home/apt_x_Cleaning Fee (USD)'],
      dtype='object')
                            OLS Regression Results
==============================================================================
Dep. Variable:           Nightly Rate   R-squared:                       0.588
Model:                            OLS   Adj. R-squared:                  0.586
Method:                 Least Squares   F-statistic:                     218.0
Date:                Sat, 09 Dec 2023   Prob (F-statistic):               0.00
Time:                        01:00:02   Log-Likelihood:            -1.3939e+05
No. Observations:               23035   AIC:                         2.791e+05
Df Residuals:                   22884   BIC:                         2.803e+05
Df Model:                         150
Covariance Type:            nonrobust
==========================================================================================================================
                                                              coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------------------
Intercept                                                 277.0514     29.843      9.284      0.000     218.557     335.546
ListingType_Entire home/apt_x_Cleaning Fee (USD)           -6.1529      0.564    -10.919      0.000      -7.257      -5.048
ListingType_Entire home/apt_x_prop_5_star_Rev_pastYear   -889.7278     76.581    -11.618      0.000   -1039.831    -739.624
ListingType_Entire home/apt_x_prev_numReserv_pastYear       0.7810      0.056     14.048      0.000       0.672       0.890
ListingType_Entire home/apt_x_Max Guests                  104.0374     10.273     10.127      0.000      83.902     124.173
ListingType_Entire home/apt_x_Neighborhood_South End     -369.9175     77.762     -4.757      0.000    -522.337    -217.498
ListingType_Entire home/apt_x_Superhost_Status_Frequency   -6.9896      1.229     -5.687      0.000      -9.398      -4.581
ListingType_Entire home/apt_x_PropertyType_House          359.5196    177.049      2.031      0.042      12.493     706.547
ListingType_Entire home/apt_x_Number of Photos              8.8933      1.415      6.285      0.000       6.120      11.667
ListingType_Entire home/apt_x_ListingsCount_Zipcode        -0.0672      0.011     -5.971      0.000      -0.089      -0.045
```

PURDUE UNIVERSITY

Backward Selection Model indicating high VIF




```python
# Recalculate VIF with the updated DataFrame
vif_data = pd.DataFrame()
vif_data["feature"] = X_numeric.columns
vif_data["VIF"] = [variance_inflation_factor(X_numeric.values, i) for i in range(X_numeric.shape[1])]
print(vif_data)
```

	feature	VIF
0	ListingType_Entire home/apt	9.945248
1	Bedrooms	53.316272
2	Bathrooms	50.484812
3	Cleaning Fee (USD)	5.159395
4	SuperhostRatio_Zipcode	10.526079
5	numReviews_pastYear	67.454735
6	prev_numReserv_pastYear	2.463324
7	Max Guests	18.222496
8	Neighborhood_South End	1.315921
9	Superhost_Status_Frequency	2.608229
10	Bathroom_Bedroom_Ratio	54.993945
11	PropertyType_House	1.255246
12	Neighborhood_Fenway/Kenmore	1.564440
13	Number of Photos	4.956380
14	num_5_star_Rev_pastYear	74.091901
15	rating_ave_pastYear	67.141202
16	ListingsCount_Zipcode	7.602532
17	available_days	6.330518
18	PropertyType_Serviced apartment	1.105181

Feature Engineering

```
[151] # 1. Superhost Status Frequency
      # Count the number of times a host has achieved Superhost status
      subset_data['Superhost_Status_Frequency'] = subset_data.groupby('Airbnb Host ID')['host_is_superhost_in_period'].transform('sum')
```

```
# 2. Ratio of 5-Star Reviews for Superhosts
# This assumes 'num_5_star_Rev_pastYear' represents the number of 5-star reviews in the past year
# Only calculating for Superhosts (host_is_superhost_in_period == 1)
subset_data['Ratio_5_Star_Superhost'] = subset_data.apply(
    lambda x: x['num_5_star_Rev_pastYear'] / x['numReviews_pastYear'] if (x['host_is_superhost_in_period'] == 1 and x['numReviews_pastYear'] > 0) else 0,
    axis=1
)
```

```
[ ] # # 3. Superhost Impact on Revenue
```

PURDUE
UNIVERSITY