Maximizing Revenue for Airbnb Properties in Chicago

TEAM 6

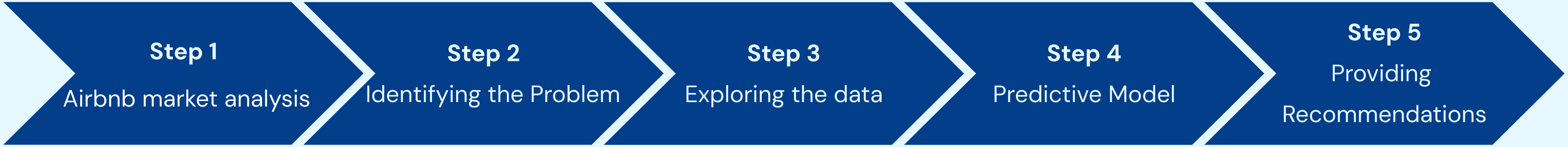
AMRUTHA GABBITA

MITHILA REDDY CHITUKULA

SRINIJA SRIMAMILLA

THANMAYEE ANSETTY

Project Flow



Step 1

Airbnb market analysis

Step 2

Identifying the Problem

Step 3

Exploring the data

Step 4

Predictive Model

Step 5

Providing Recommendations

Business Problem

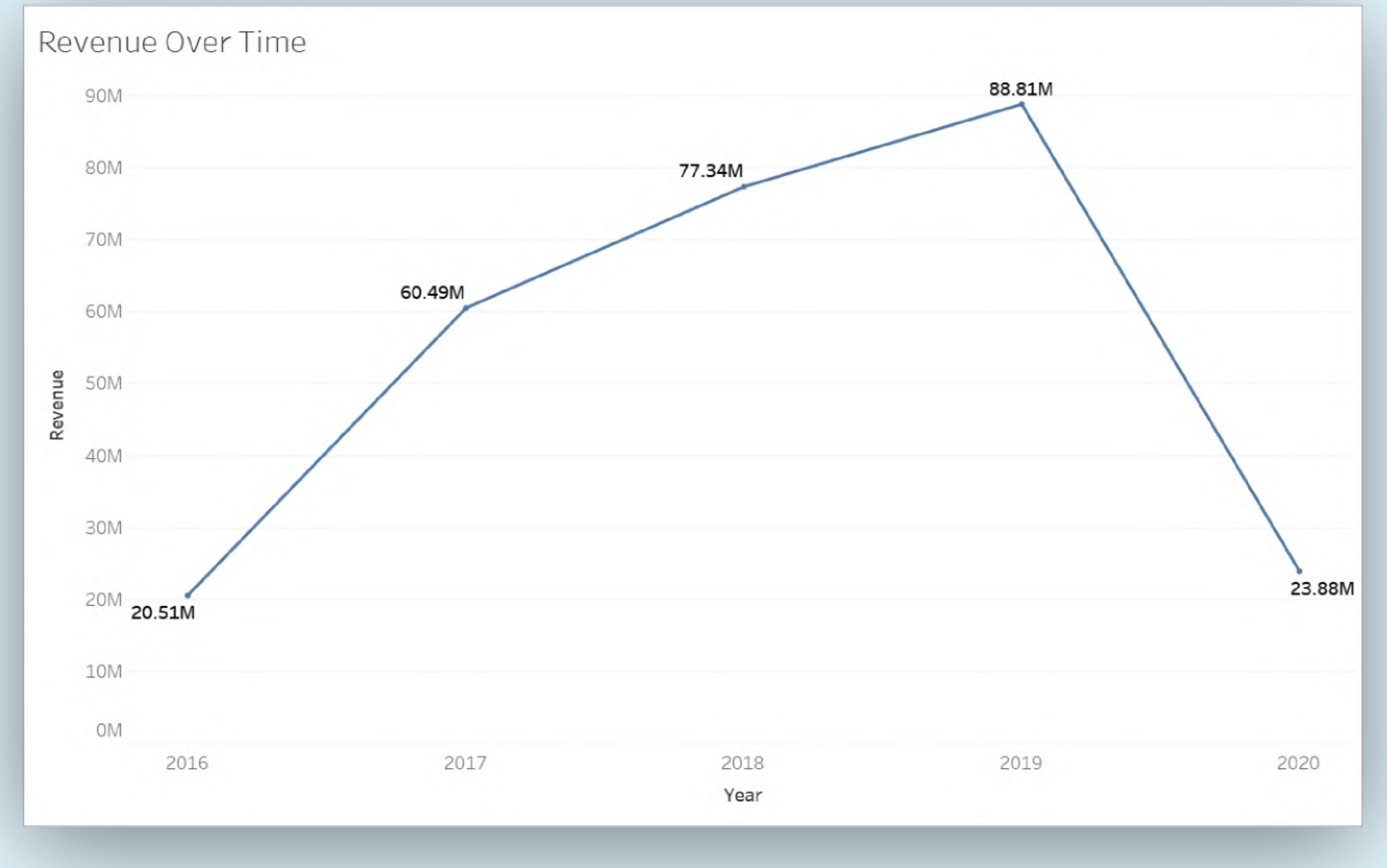
Assessing the Interplay between Occupancy Rates and Geographic Location in Maximizing Revenue for Airbnb Properties in Chicago

Dataset description

The dataset provides a detailed analysis of Airbnb listings in Chicago, tracking Superhost status, occupancy, revenue, and property details over eight evaluation periods, integrating host performance, demographic data, and geographic factors to offer a comprehensive view of the local Airbnb market dynamics.

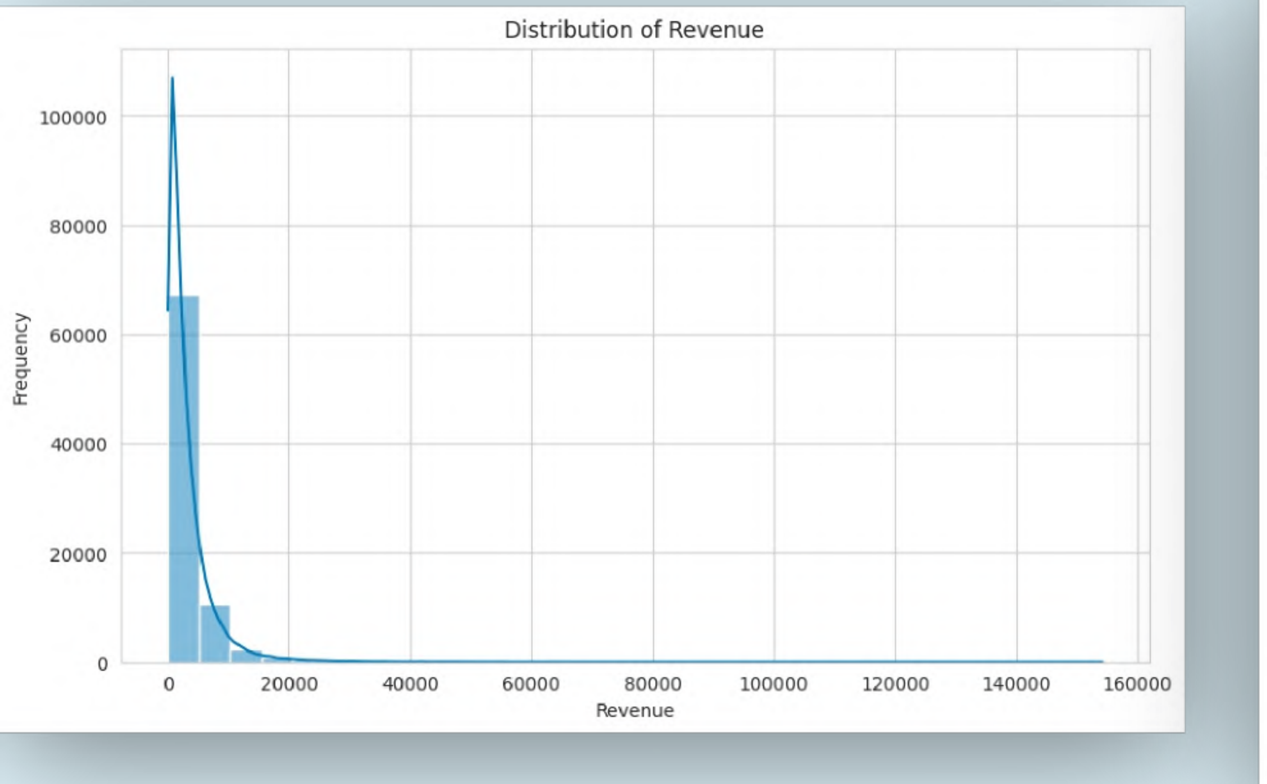
How it all started...

1. We were presented with a very complex and a real-time dataset of Airbnb for different cities across the US.
2. We chose Chicago as it is one of the major cities
3. Nicholas Gerli - Reventure Consulting CEO | Data-Driven Real Estate said "***The supply of airbnb's has surged out of control, way more than demand. Certain markets, causing revenue to go down substantially, particularly for mom and pop airbnb owners without access to fancy pricing tools the big guys do. I'm talking with owners around America, their listing is doing 40% less revenue this year compared to last year.***" - Source: CNBC Television, August 17th 2023
4. There began our ideology to work on revenue as the problem statement.
5. The source from CNBC was just our starting point, but we wanted to validate our data...



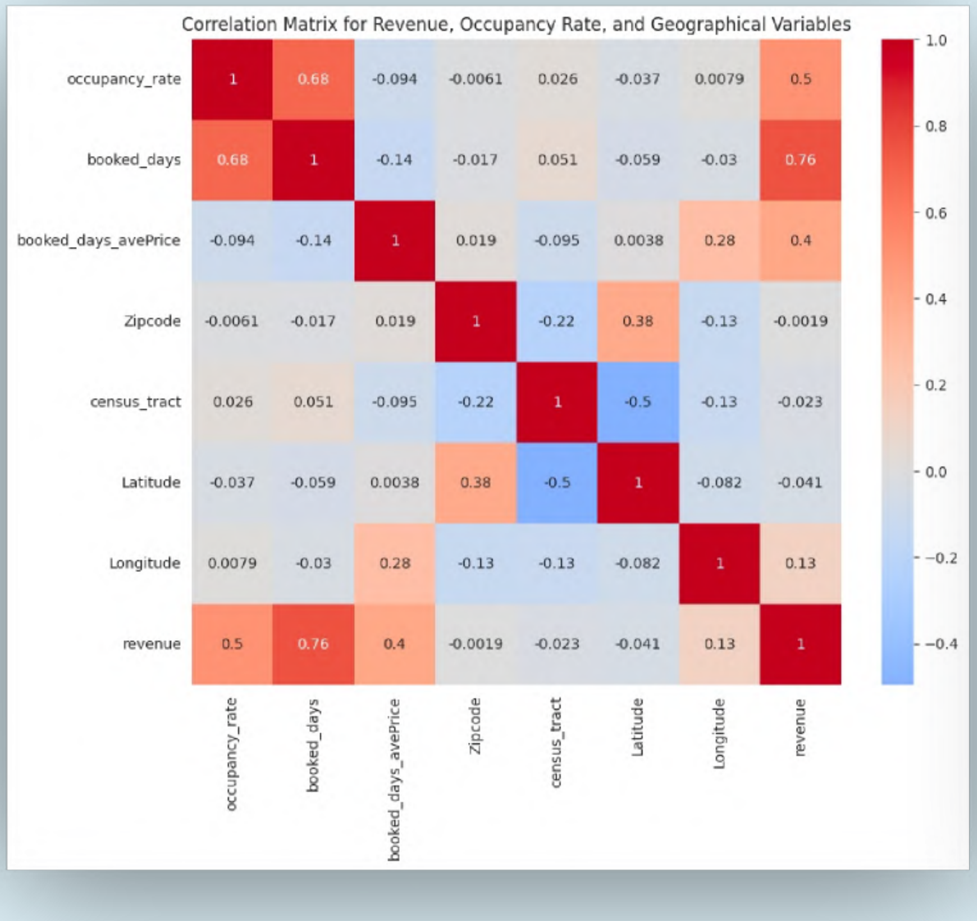
We validated with our data by checking the trend of revenue over time

Exploratory Data Analysis



DISTRIBUTION OF REVENUE WAS RIGHT SKEWED

Most listings have **lower** revenue, while a smaller number of listings have very **high** revenue.

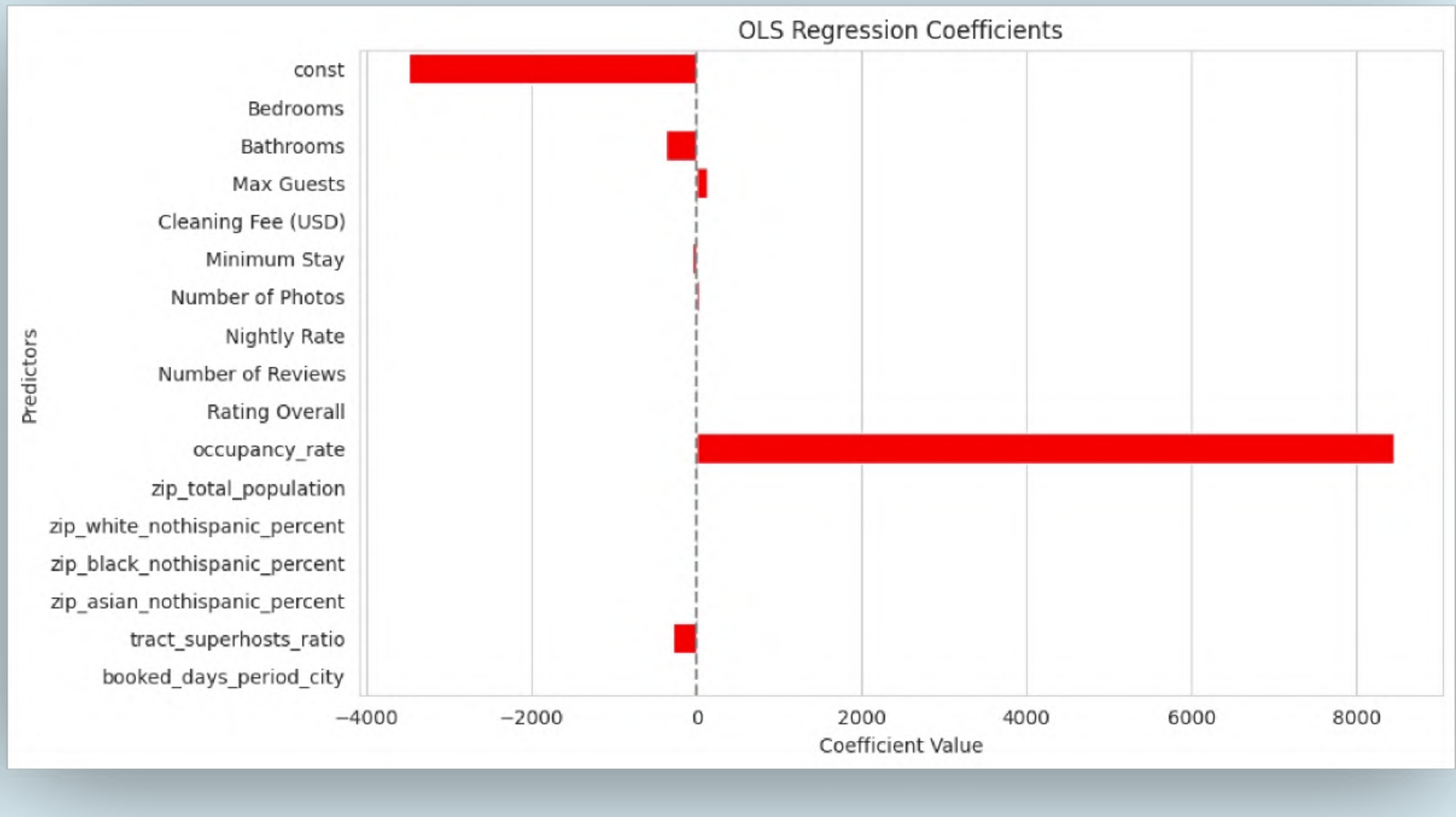


Correlation Matrix for Revenue, Occupancy Rate, and Geographical Variables

	occupancy_rate	booked_days	booked_days_avePrice	Zipcode	census_tract	Latitude	Longitude	revenue
occupancy_rate	1	0.68	-0.094	-0.0061	0.026	-0.037	0.0079	0.5
booked_days	0.68	1	-0.14	-0.017	0.051	-0.059	-0.03	0.76
booked_days_avePrice	-0.094	-0.14	1	0.019	-0.095	0.0038	0.28	0.4
Zipcode	-0.0061	-0.017	0.019	1	-0.22	0.38	-0.13	-0.0019
census_tract	0.026	0.051	-0.095	-0.22	1	-0.5	-0.13	-0.023
Latitude	-0.037	-0.059	0.0038	0.38	-0.5	1	-0.082	-0.041
Longitude	0.0079	-0.03	0.28	-0.13	-0.13	-0.082	1	0.13
revenue	0.5	0.76	0.4	-0.0019	-0.023	-0.041	0.13	1

CORRELATION BETWEEN REVENUE AND RELEVENT SUBSET OF VARIABLES

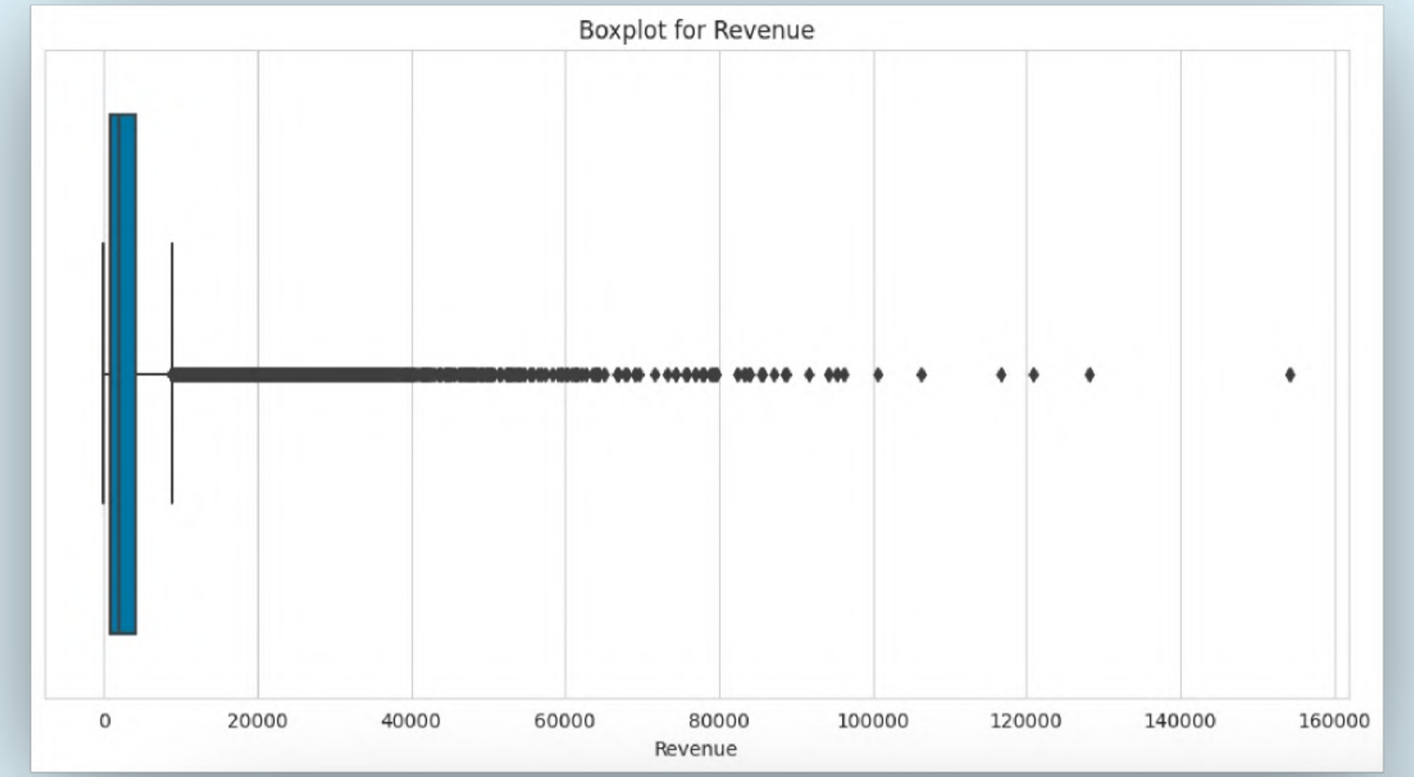
occupancy_rate has a **strong positive correlation with revenue** (0.5), suggesting that listings with **higher occupancy rates tend to earn more revenue.**



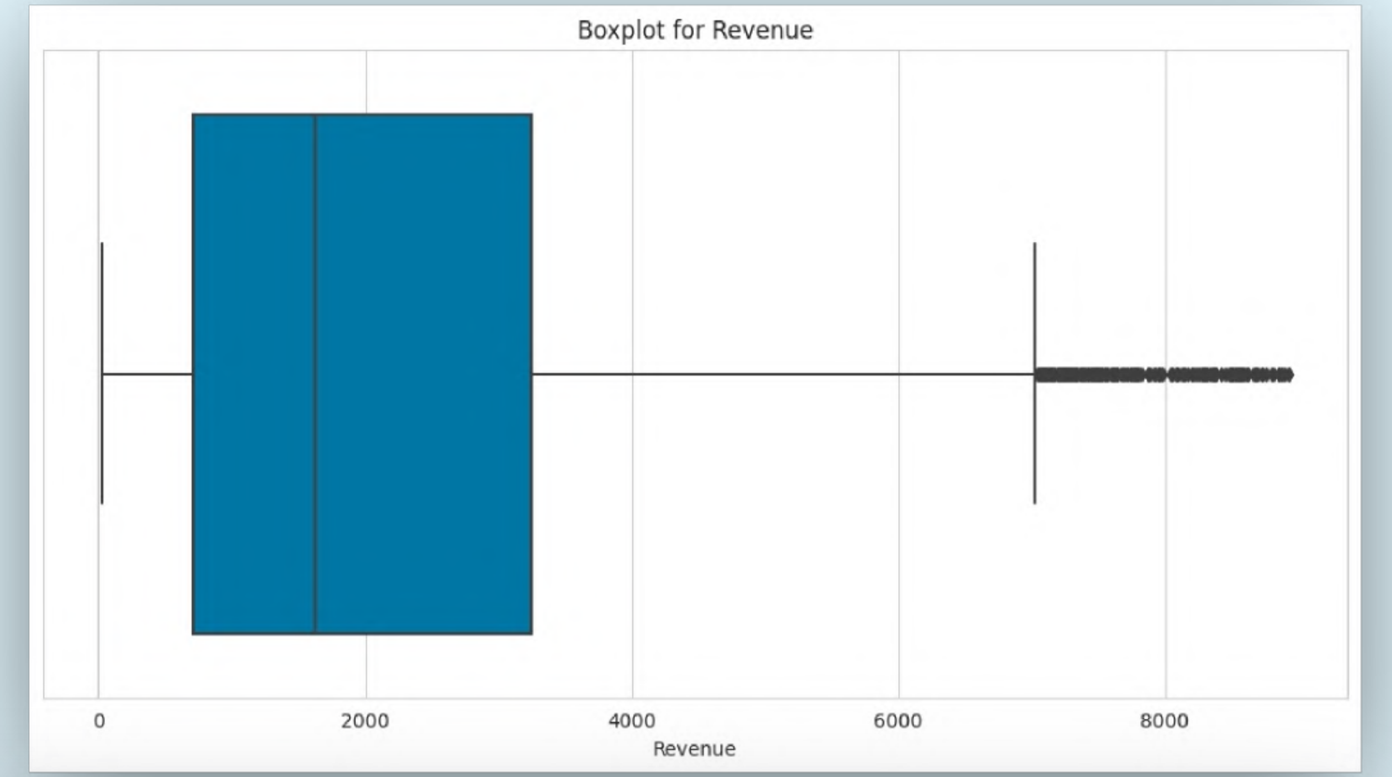
DETERMINING OCCUPANY RATE TO BE THE MOST STRONG PREDICTOR OF REVENUE

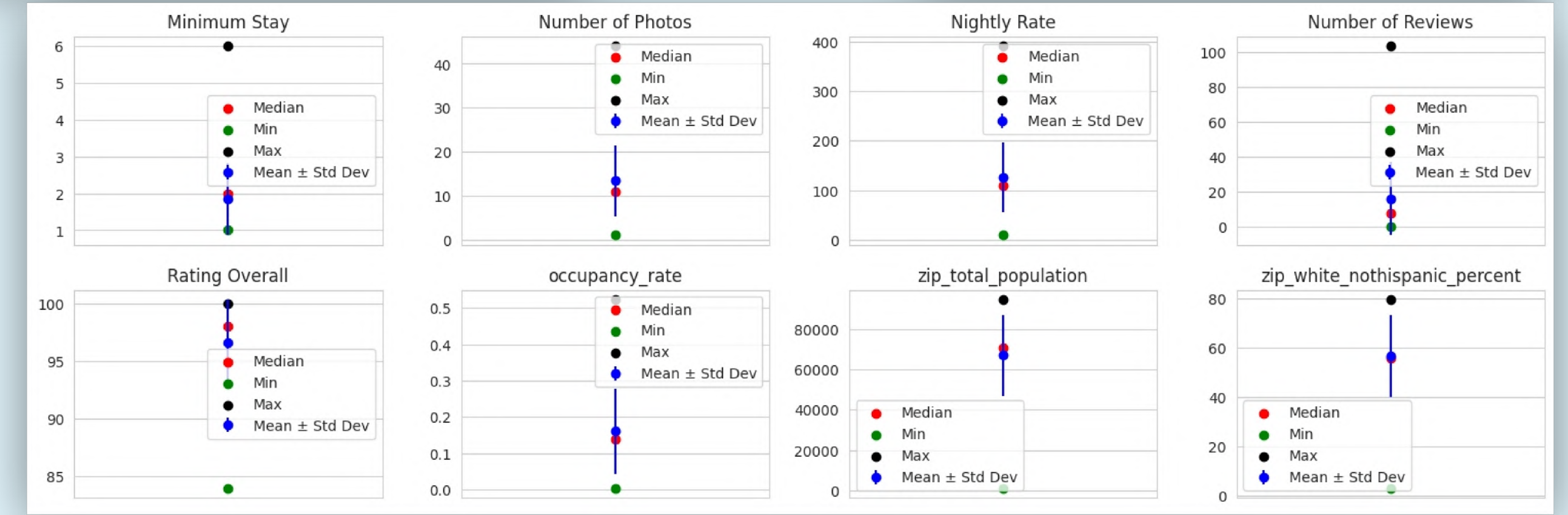
'occupancy_rate' has long red bars, suggesting it has a **large and statistically significan**t impact on 'revenue'.

Summary Statistics



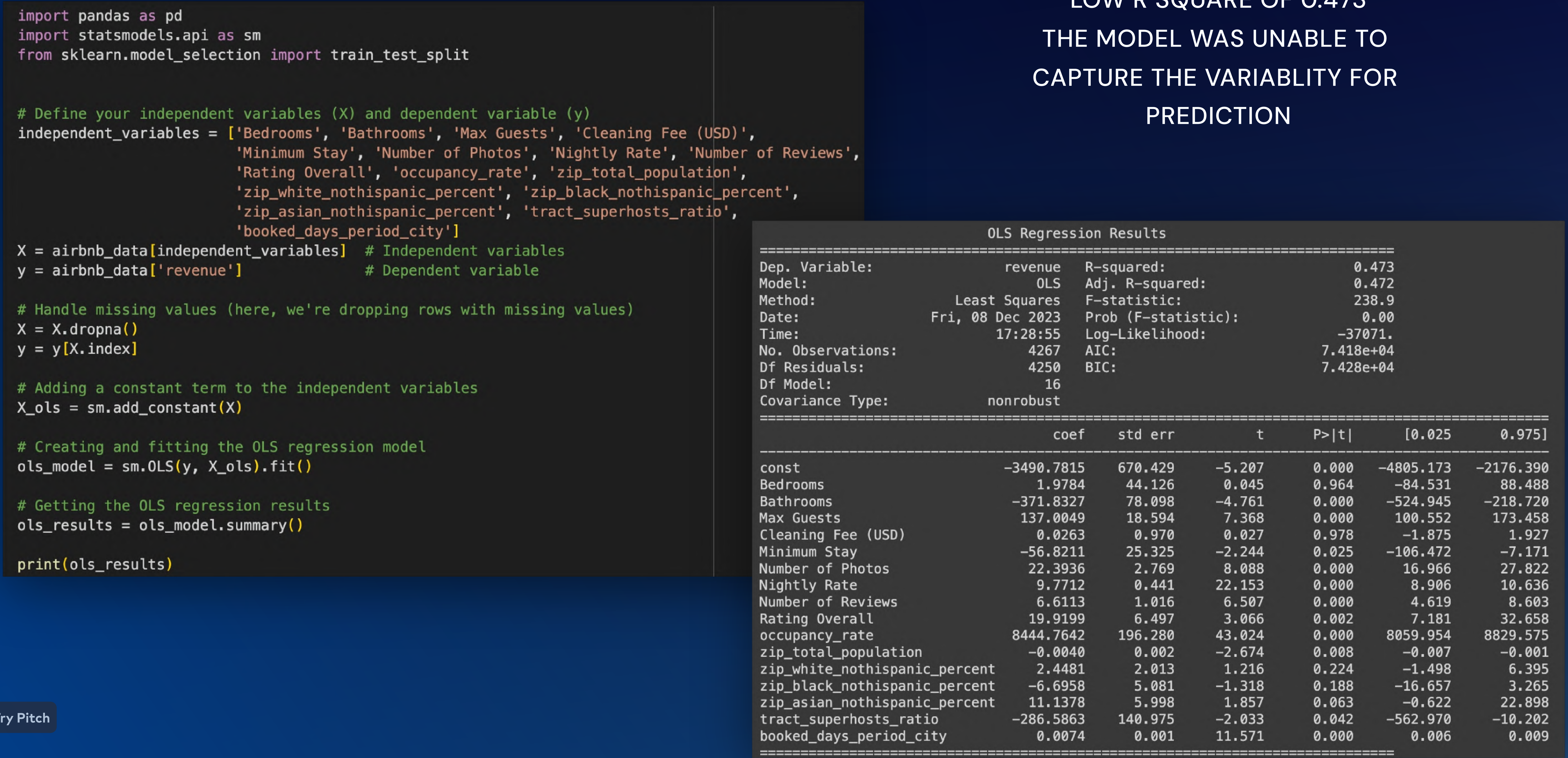
Outliers handling





Linear Regression Model

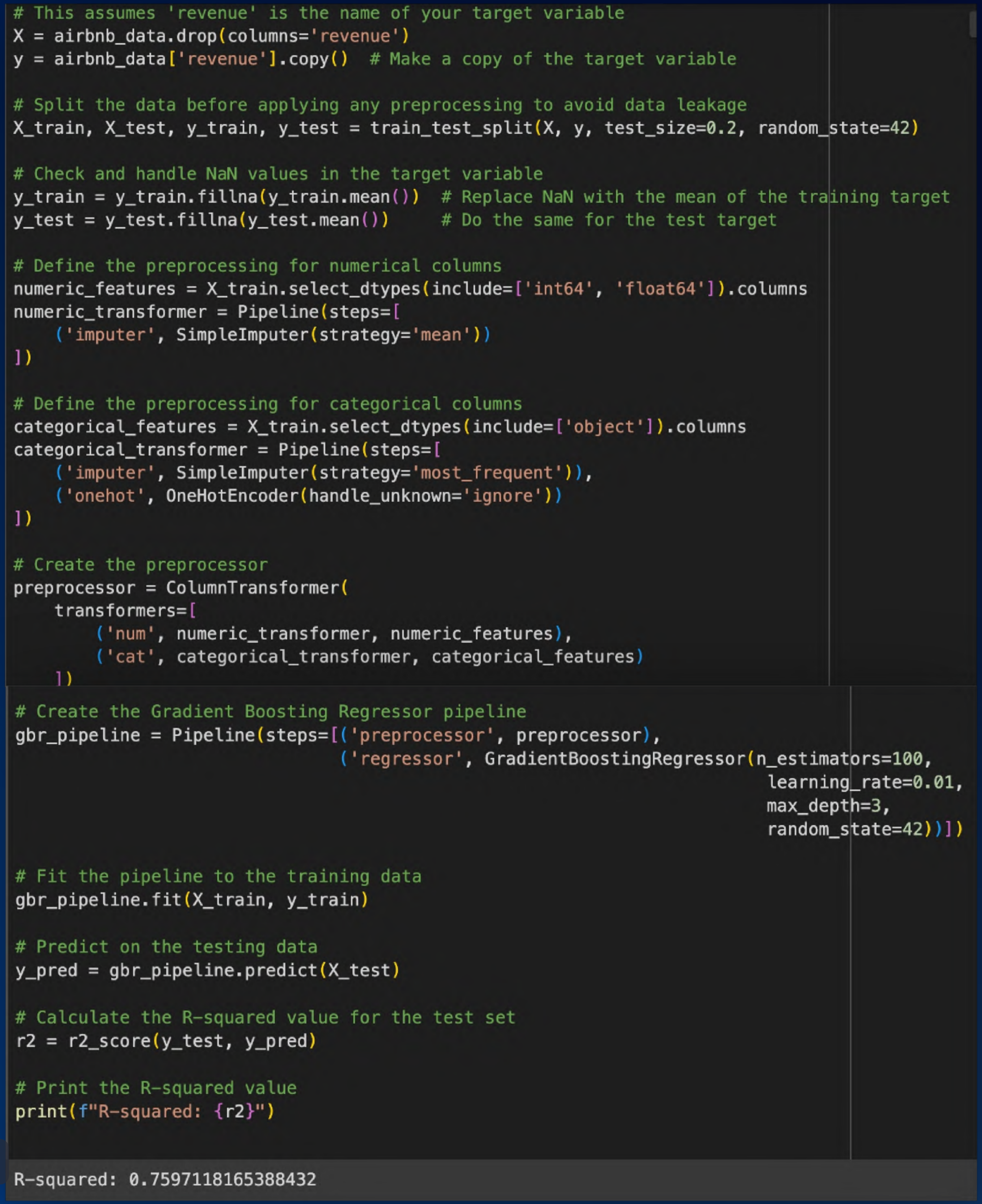
LOW R SQUARE OF 0.473
THE MODEL WAS UNABLE TO
CAPTURE THE VARIABLITY FOR
PREDICTION



```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split


# Define your independent variables (X) and dependent variable (y)
independent_variables = ['Bedrooms', 'Bathrooms', 'Max Guests', 'Cleaning Fee (USD)',
                         'Minimum Stay', 'Number of Photos', 'Nightly Rate', 'Number of Reviews',
                         'Rating Overall', 'occupancy_rate', 'zip_total_population',
                         'zip_white_nothispanic_percent', 'zip_black_nothispanic_percent',
                         'zip_asian_nothispanic_percent', 'tract_superhosts_ratio',
                         'booked_days_period_city']
X = airbnb_data[independent_variables]  # Independent variables
y = airbnb_data['revenue']              # Dependent variable

# Handle missing values (here, we're dropping rows with missing values)
X = X.dropna()
y = y[X.index]

# Adding a constant term to the independent variables
X_ols = sm.add_constant(X)

# Creating and fitting the OLS regression model
ols_model = sm.OLS(y, X_ols).fit()

# Getting the OLS regression results
ols_results = ols_model.summary()

print(ols_results)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.473
Model:                            OLS   Adj. R-squared:                  0.472
Method:                 Least Squares   F-statistic:                     238.9
Date:                Fri, 08 Dec 2023   Prob (F-statistic):               0.00
Time:                        17:28:55   Log-Likelihood:                -37071.
No. Observations:                4267   AIC:                         7.418e+04
Df Residuals:                    4250   BIC:                         7.428e+04
Df Model:                          16
Covariance Type:            nonrobust
=================================================================================================
                                    coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
const                         -3490.7815    670.429     -5.207      0.000   -4805.173   -2176.390
Bedrooms                          1.9784     44.126      0.045      0.964     -84.531      88.488
Bathrooms                      -371.8327     78.098     -4.761      0.000    -524.945    -218.720
Max Guests                      137.0049     18.594      7.368      0.000     100.552     173.458
Cleaning Fee (USD)                0.0263      0.970      0.027      0.978      -1.875       1.927
Minimum Stay                    -56.8211     25.325     -2.244      0.025    -106.472      -7.171
Number of Photos                 22.3936      2.769      8.088      0.000      16.966      27.822
Nightly Rate                      9.7712      0.441     22.153      0.000       8.906      10.636
Number of Reviews                 6.6113      1.016      6.507      0.000       4.619       8.603
Rating Overall                   19.9199      6.497      3.066      0.002       7.181      32.658
occupancy_rate                 8444.7642    196.280     43.024      0.000    8059.954    8829.575
zip_total_population             -0.0040      0.002     -2.674      0.008      -0.007      -0.001
zip_white_nothispanic_percent     2.4481      2.013      1.216      0.224      -1.498       6.395
zip_black_nothispanic_percent    -6.6958      5.081     -1.318      0.188     -16.657       3.265
zip_asian_nothispanic_percent    11.1378      5.998      1.857      0.063      -0.622      22.898
tract_superhosts_ratio         -286.5863    140.975     -2.033      0.042    -562.970     -10.202
booked_days_period_city           0.0074      0.001     11.571      0.000       0.006       0.009
==============================================================================
```

Gradient Boosting Model



```
# This assumes 'revenue' is the name of your target variable
X = airbnb_data.drop(columns='revenue')
y = airbnb_data['revenue'].copy()  # Make a copy of the target variable

# Split the data before applying any preprocessing to avoid data leakage
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Check and handle NaN values in the target variable
y_train = y_train.fillna(y_train.mean())  # Replace NaN with the mean of the training target
y_test = y_test.fillna(y_test.mean())     # Do the same for the test target

# Define the preprocessing for numerical columns
numeric_features = X_train.select_dtypes(include=['int64', 'float64']).columns
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean'))
])

# Define the preprocessing for categorical columns
categorical_features = X_train.select_dtypes(include=['object']).columns
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Create the preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

```
# Create the Gradient Boosting Regressor pipeline
gbr_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('regressor', GradientBoostingRegressor(n_estimators=100,
                                                                        learning_rate=0.01,
                                                                        max_depth=3,
                                                                        random_state=42))])

# Fit the pipeline to the training data
gbr_pipeline.fit(X_train, y_train)

# Predict on the testing data
y_pred = gbr_pipeline.predict(X_test)

# Calculate the R-squared value for the test set
r2 = r2_score(y_test, y_pred)

# Print the R-squared value
print(f"R-squared: {r2}")
```

```
R-squared: 0.7597118165388432
```

- **IMPROVED R SQUARE** TO 0.759, PROVING GRADIENT BOOSTING IS CAPTURING BETTER VARIABILITY

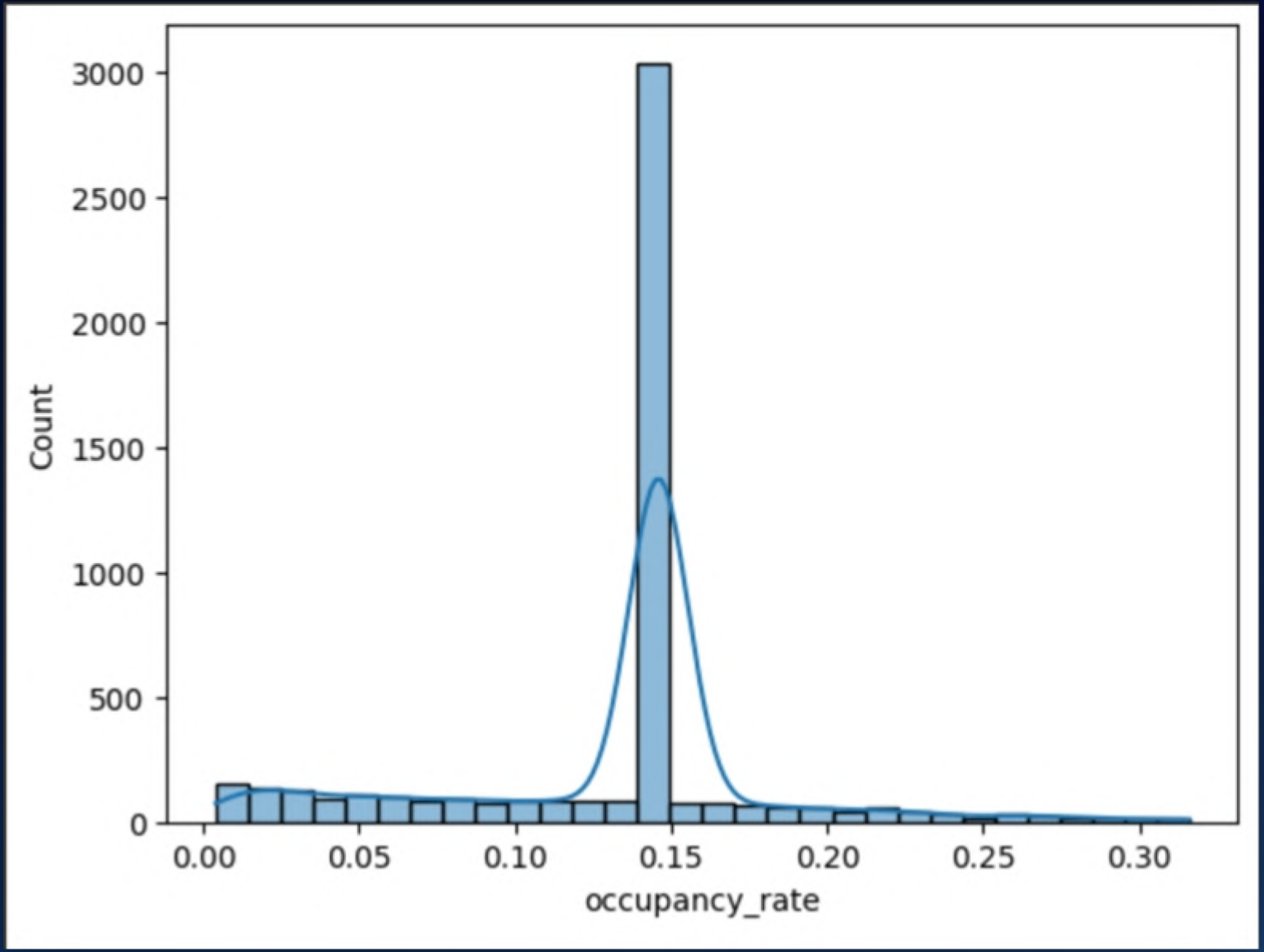
- **GOOD MODEL FIT** HAS A FAIRLY STRONG PREDICTIVE POWER AND IS CAPTURING A SIGNIFICANT PORTION OF THE VARIANCE IN THE DATA.

- **BALANCE BETWEEN BIAS AND VARIANCE:** EFFECTIVE LEARNING RATE TO AVOID OVERFITTING

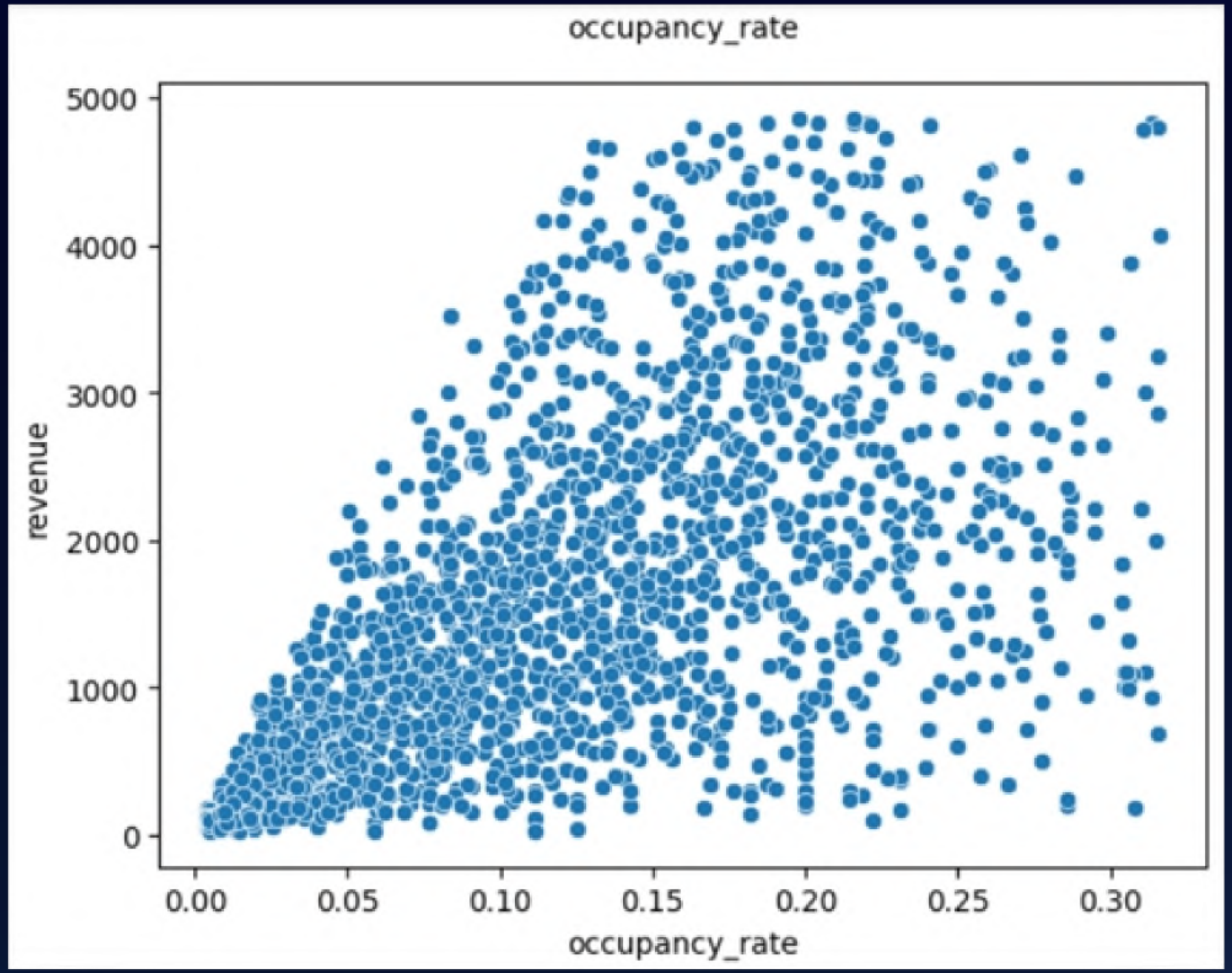
- THE MODEL IS COMPLEX ENOUGH TO CAPTURE THE ESSENTIAL PATTERNS IN THE DATA, BUT NOT SO COMPLEX THAT IT'S HEAVILY INFLUENCED BY THE NOISE OR SPECIFIC PECULIARITIES OF THE TRAINING DATASET.

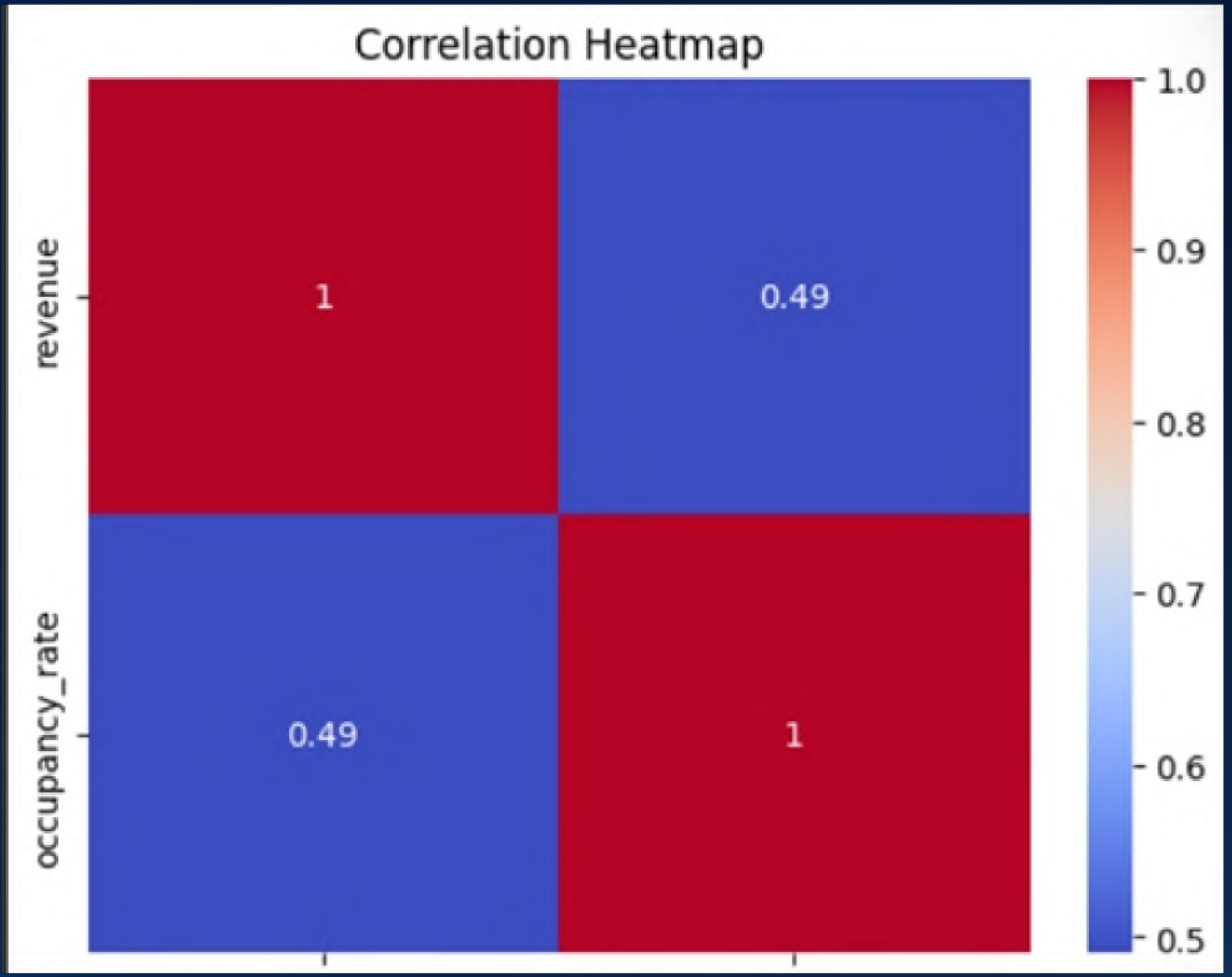
Most Listings are Rarely Booked

- The vast majority of properties have very low occupancy rates, clustering close to 0.15
- There is a lack of properties with moderate or high occupancy rates



Revenue and Occupancy_rate are positively correlated

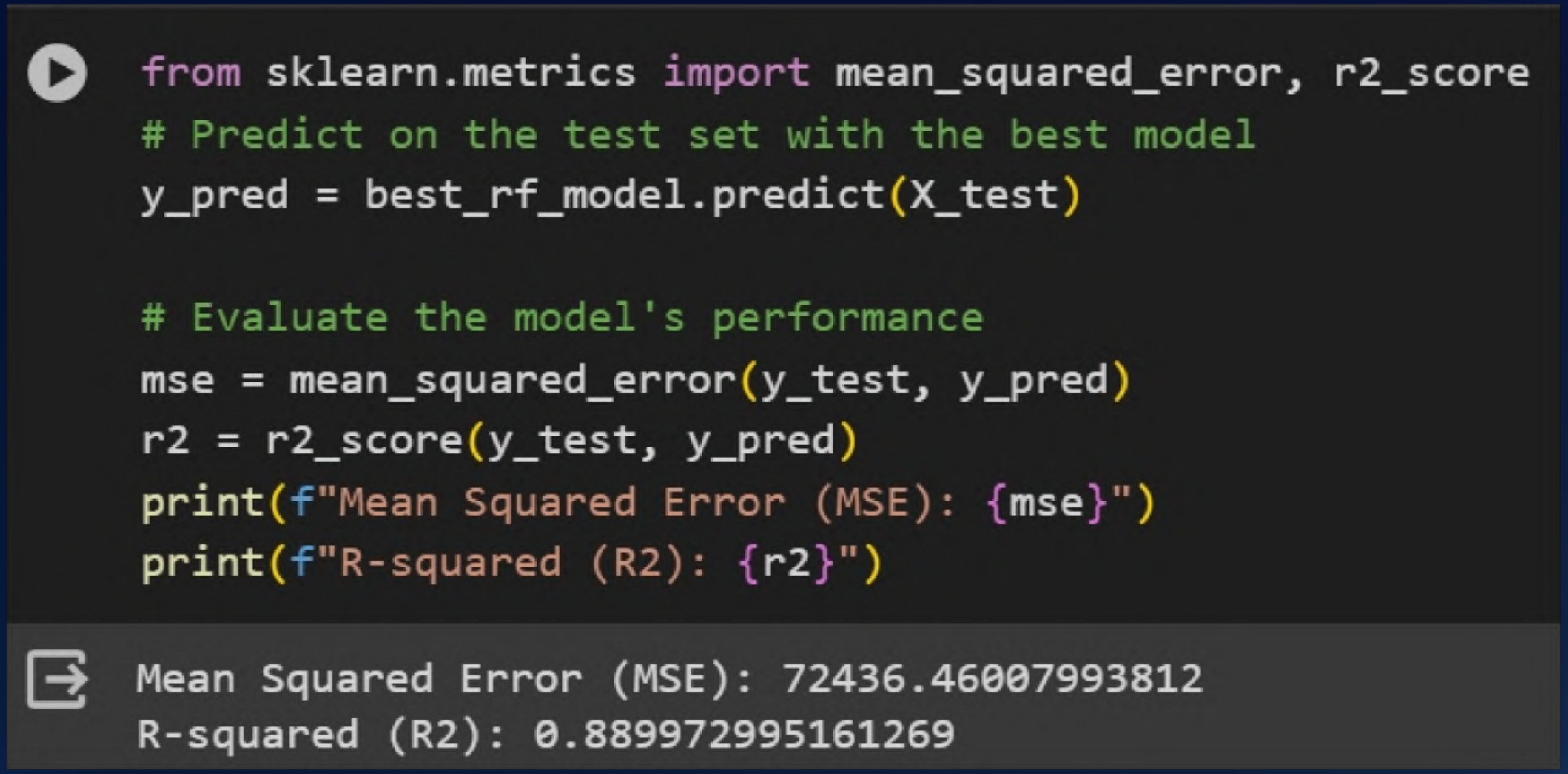




Model Building: A Random Forest Model with R-square Value

88.9%

is built which signifies that the Model is Robust

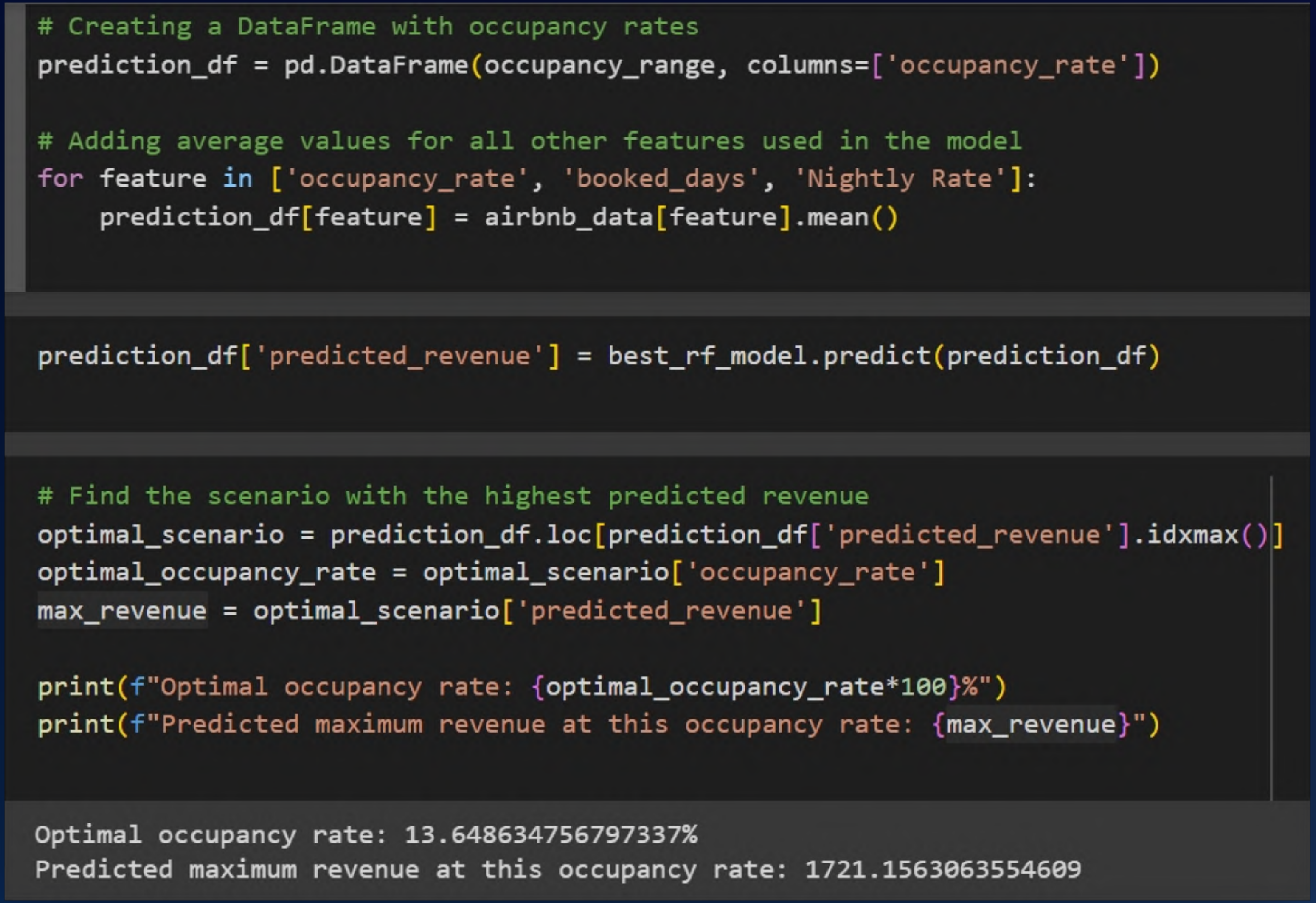


```python
from sklearn.metrics import mean_squared_error, r2_score
# Predict on the test set with the best model
y_pred = best_rf_model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")
```

```
Mean Squared Error (MSE): 72436.46007993812
R-squared (R2): 0.889972995161269
```

Finding the Optimal Occupancy Rate

The Random Forest Model has been used to predict an optimal occupancy rate of 13.65% approximately with a maximum revenue of 1721.16 approximately

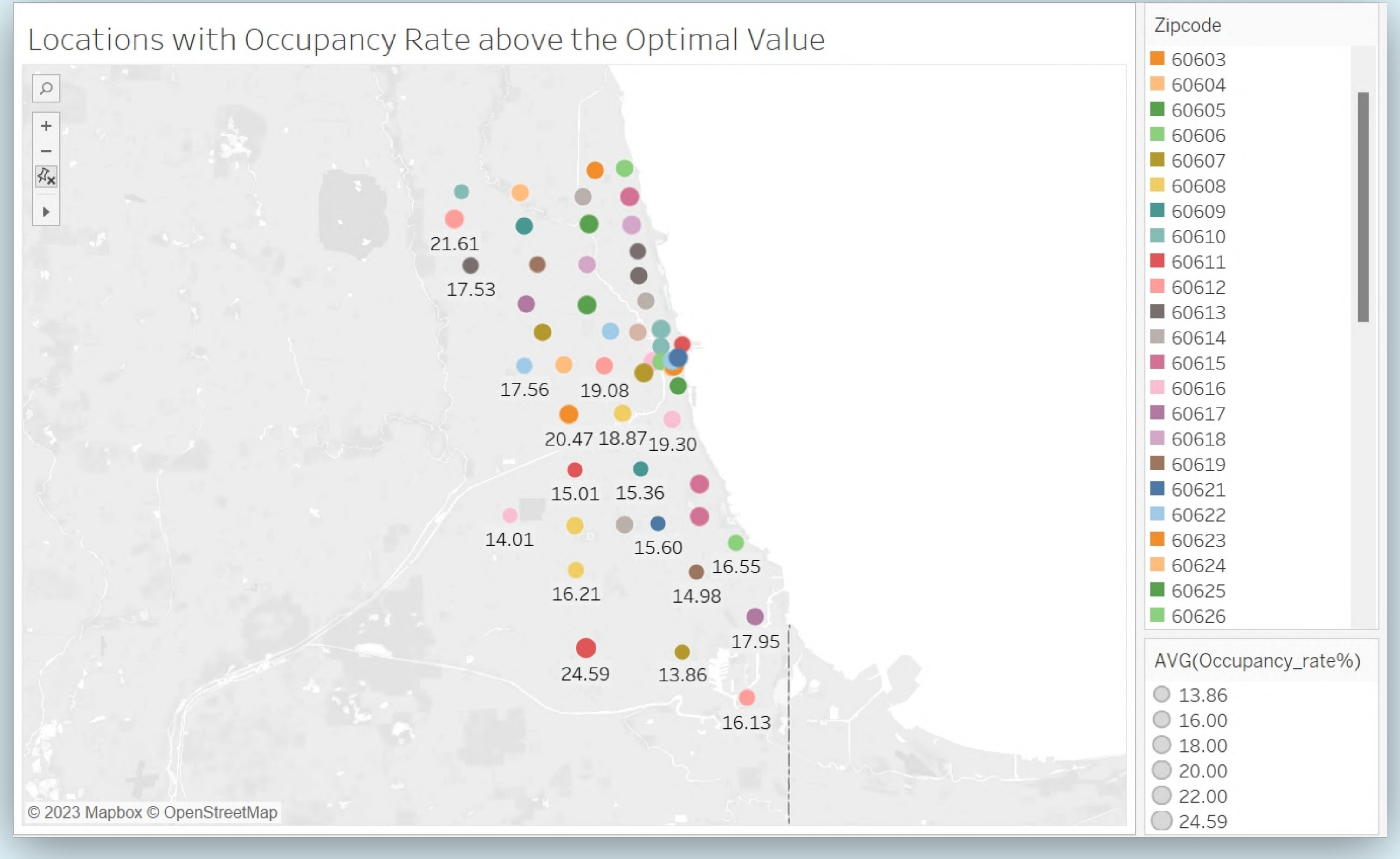


```
# Creating a DataFrame with occupancy rates
prediction_df = pd.DataFrame(occupancy_range, columns=['occupancy_rate'])

# Adding average values for all other features used in the model
for feature in ['occupancy_rate', 'booked_days', 'Nightly Rate']:
    prediction_df[feature] = airbnb_data[feature].mean()
```

```
prediction_df['predicted_revenue'] = best_rf_model.predict(prediction_df)
```

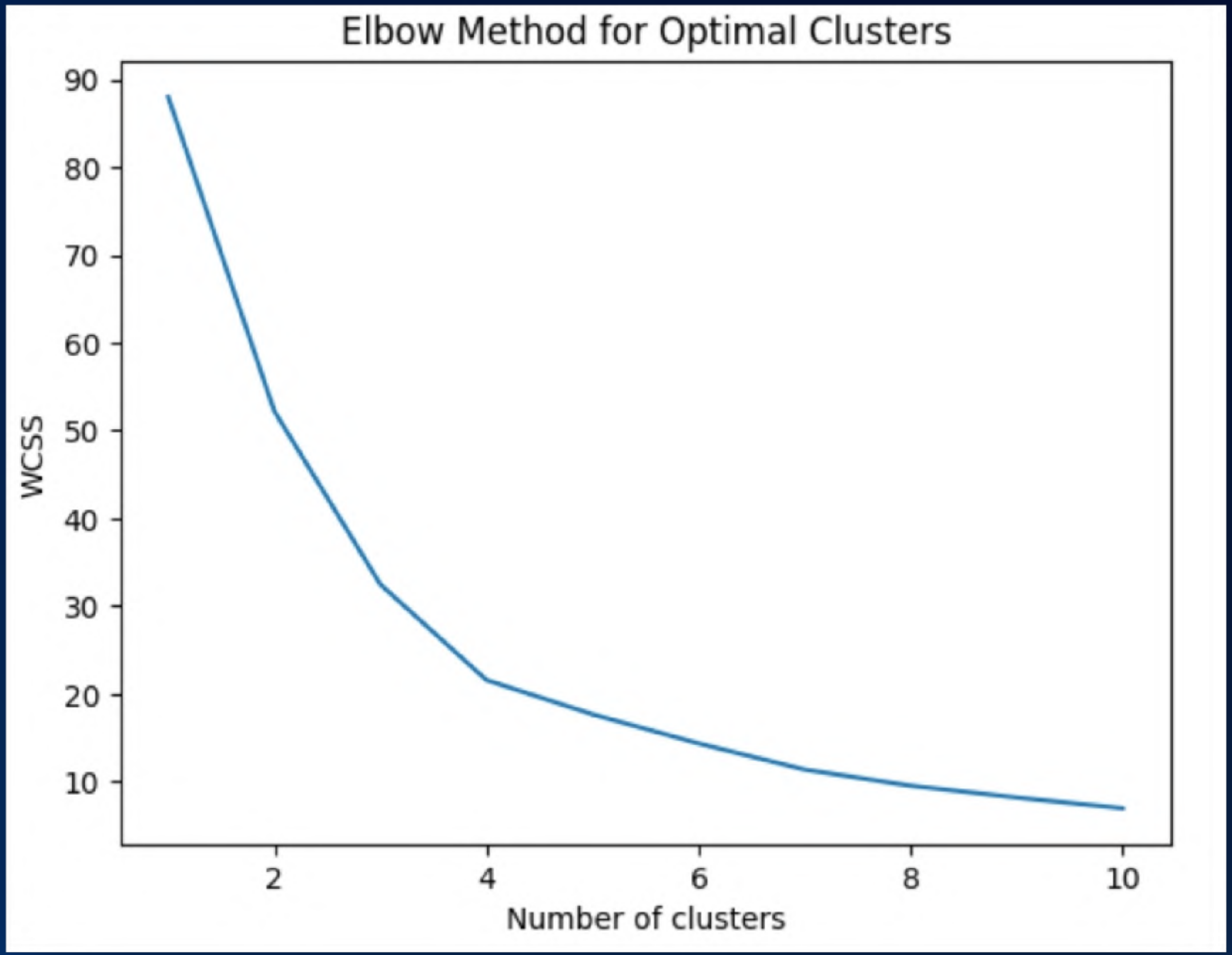
```
# Find the scenario with the highest predicted revenue
optimal_scenario = prediction_df.loc[prediction_df['predicted_revenue'].idxmax()]
optimal_occupancy_rate = optimal_scenario['occupancy_rate']
max_revenue = optimal_scenario['predicted_revenue']

print(f"Optimal occupancy rate: {optimal_occupancy_rate*100}%")
print(f"Predicted maximum revenue at this occupancy rate: {max_revenue}")
```

```
Optimal occupancy rate: 13.648634756797337%
Predicted maximum revenue at this occupancy rate: 1721.1563063554609
```

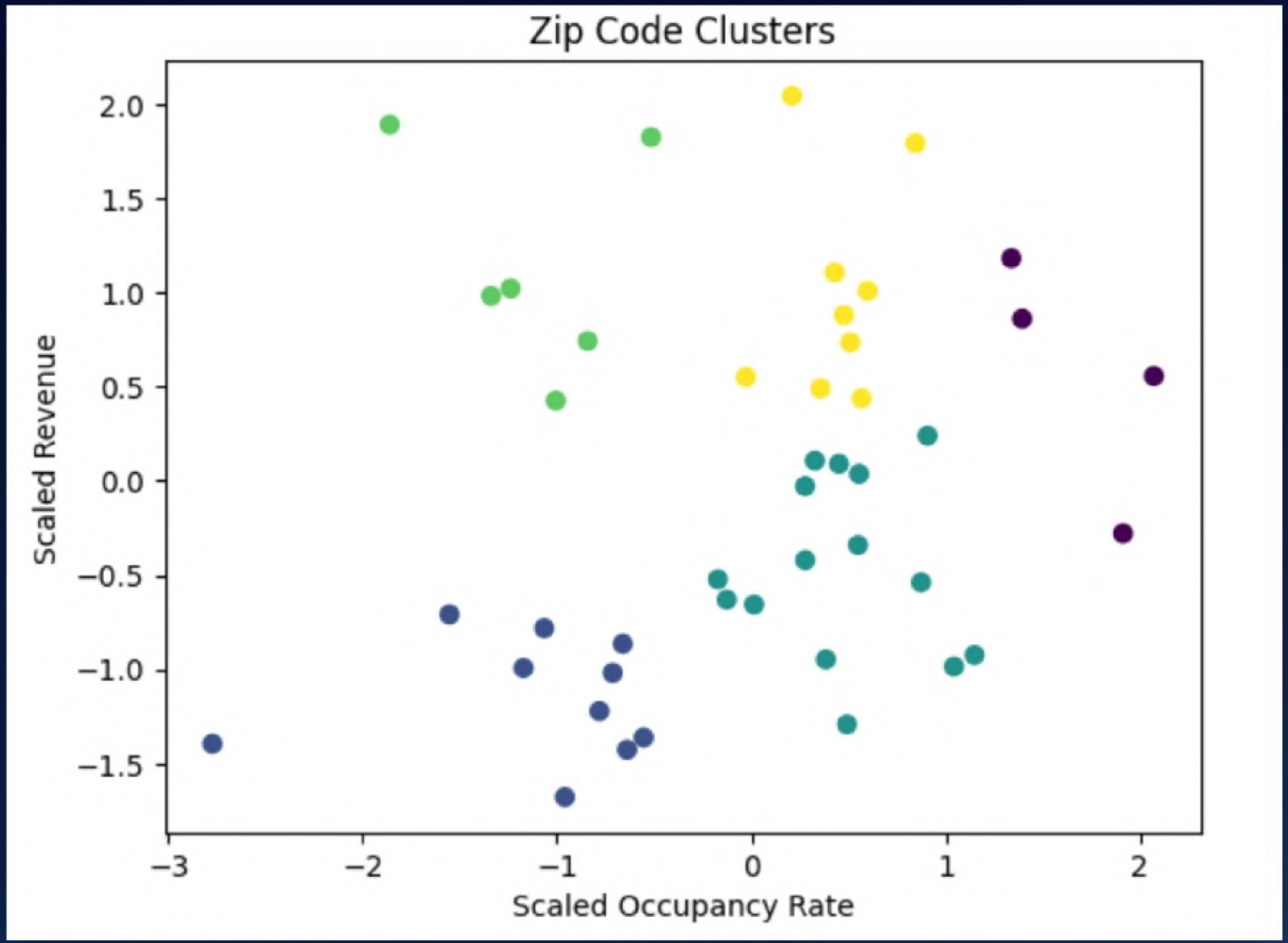


Locations with Occupancy Rate above the Optimal Value

K-Means Clustering Algorithm



Identifying the optimal number of clusters to segment zip codes based on average occupancy rate and revenue



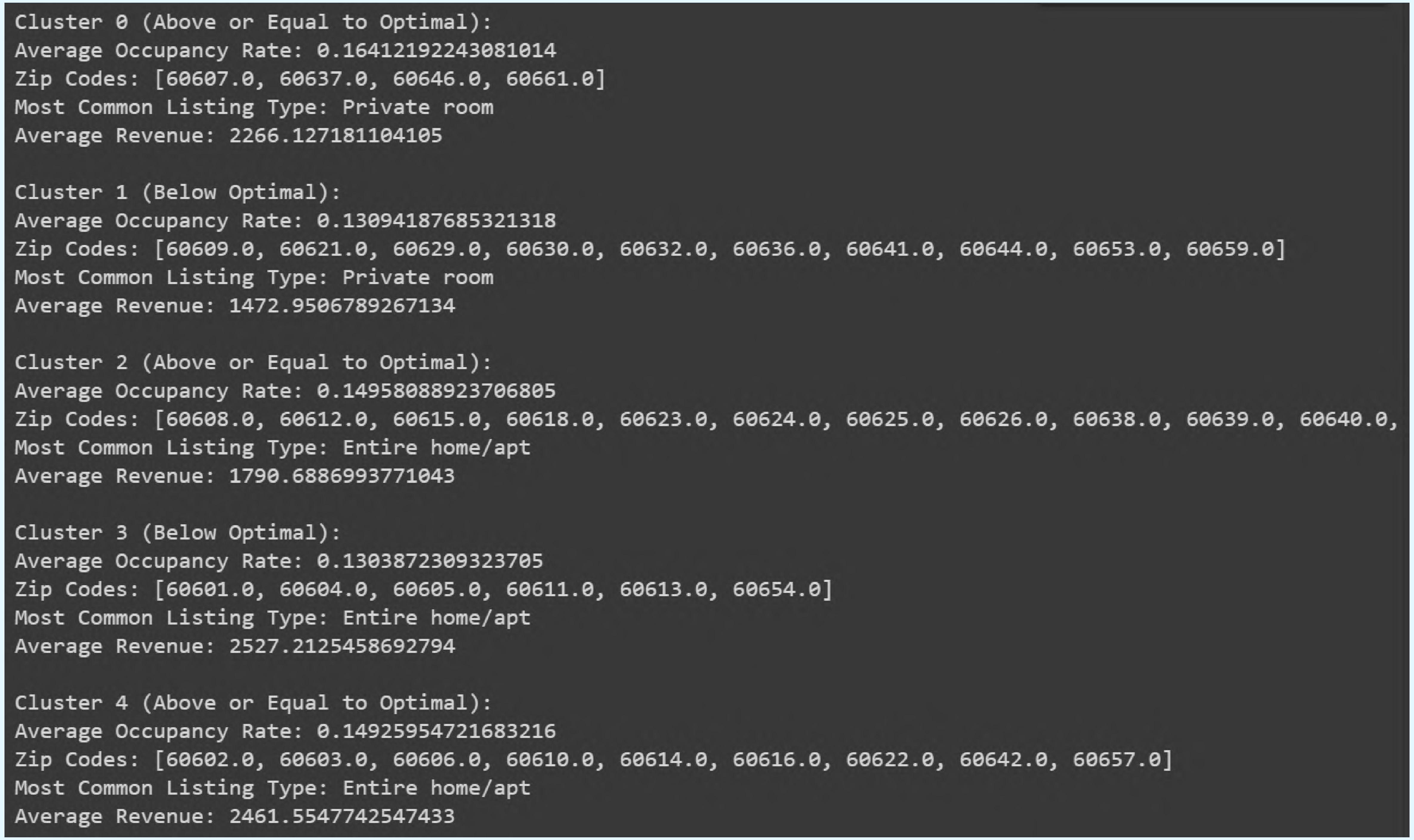
The zip codes are segmented into 5 clusters

Analysis of Occupancy Rate and Revenue by Geographic Locations (Zipcode)

Grouping by cluster and listing zip codes, along with average revenue, and most common Listing Type

Clusters 1 and 3 are below Optimal Occupancy Rate

Clusters 0, 2, and 4 are At or Above Optimal Occupancy Rate



```
Cluster 0 (Above or Equal to Optimal):
Average Occupancy Rate: 0.16412192243081014
Zip Codes: [60607.0, 60637.0, 60646.0, 60661.0]
Most Common Listing Type: Private room
Average Revenue: 2266.127181104105

Cluster 1 (Below Optimal):
Average Occupancy Rate: 0.13094187685321318
Zip Codes: [60609.0, 60621.0, 60629.0, 60630.0, 60632.0, 60636.0, 60641.0, 60644.0, 60653.0, 60659.0]
Most Common Listing Type: Private room
Average Revenue: 1472.9506789267134

Cluster 2 (Above or Equal to Optimal):
Average Occupancy Rate: 0.14958088923706805
Zip Codes: [60608.0, 60612.0, 60615.0, 60618.0, 60623.0, 60624.0, 60625.0, 60626.0, 60638.0, 60639.0, 60640.0,
Most Common Listing Type: Entire home/apt
Average Revenue: 1790.6886993771043

Cluster 3 (Below Optimal):
Average Occupancy Rate: 0.1303872309323705
Zip Codes: [60601.0, 60604.0, 60605.0, 60611.0, 60613.0, 60654.0]
Most Common Listing Type: Entire home/apt
Average Revenue: 2527.2125458692794

Cluster 4 (Above or Equal to Optimal):
Average Occupancy Rate: 0.14925954721683216
Zip Codes: [60602.0, 60603.0, 60606.0, 60610.0, 60614.0, 60616.0, 60622.0, 60642.0, 60657.0]
Most Common Listing Type: Entire home/apt
Average Revenue: 2461.5547742547433
```

Try Pitch

CONCLUSION

Summary and Key Takeaways

Business Insights

1. **Efficiency Over Occupancy**: A moderate occupancy rate might lead to higher revenues, possibly due to optimal pricing strategies that balance demand and profitability.
2. **Strategic Pricing**: Hosts may not need to aim for full occupancy to maximize revenue. Instead, they should price their listings to target the optimal occupancy rate, which could imply higher rates per night with less frequent bookings.
3. **Quality Over Quantity**: Focusing on attracting the right type of guest who is willing to pay more, possibly by offering premium amenities or experiences, can be more profitable than simply increasing the number of guests.
4. **Market Positioning**: Airbnb hosts in areas with higher-than-average occupancy rates might be underpricing their properties. Conversely, those with lower occupancy rates might be overpriced or could improve their listing's attractiveness.

Recommendations

01

Dynamic Pricing

Implement a dynamic pricing model that adjusts nightly rates to approach the optimal occupancy rate, considering demand surges during holidays or local events.

02

Property Upgrades

Invest in property enhancements that justify higher rates and attract guests who may book longer stays or pay premium rates, aligning with the optimal occupancy rate.

03

Marketing Focus

Develop marketing strategies for properties below the optimal occupancy rate, highlighting local attractions, improving the visibility of listings, or offering competitive pricing.

04

Performance Monitoring

Regularly review occupancy and revenue metrics to ensure they align with the optimal rate, and adjust strategies as needed.

Summary

1. Regression and correlation analysis identified occupancy rate as a key predictor of revenue for Airbnb hosts in Chicago
2. A random forest model, using occupancy-related variables, estimated the optimal occupancy rate for maximizing revenue
3. K-means clustering segmented zip codes into five groups by occupancy rate, revenue, and listing type, comparing each to the optimal occupancy rate

Appendix

References

- stackoverflow.com
- "Understanding Regression Analysis: An Introductory Guide" by Schroeder, Sjoquist, and Stephan.
- https://www.youtube.com/watch?v=4HjyezWs1Po
- https://www.youtube.com/watch?v=X1MRbEnEq2s
- https://youtu.be/Dirg-uDAfOM?si=3WEprwtWYgY9FCNp
- https://www.youtube.com/watch?v=bMccdk8EdGo
- Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial.
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine.
- Breiman, L. (2001). Random forests.