MGMT 683 Final Project:
Navigating Business Solutions with AI Models

Group 4

Team Members:

Liana Simopoulos

Bhavan Sekar

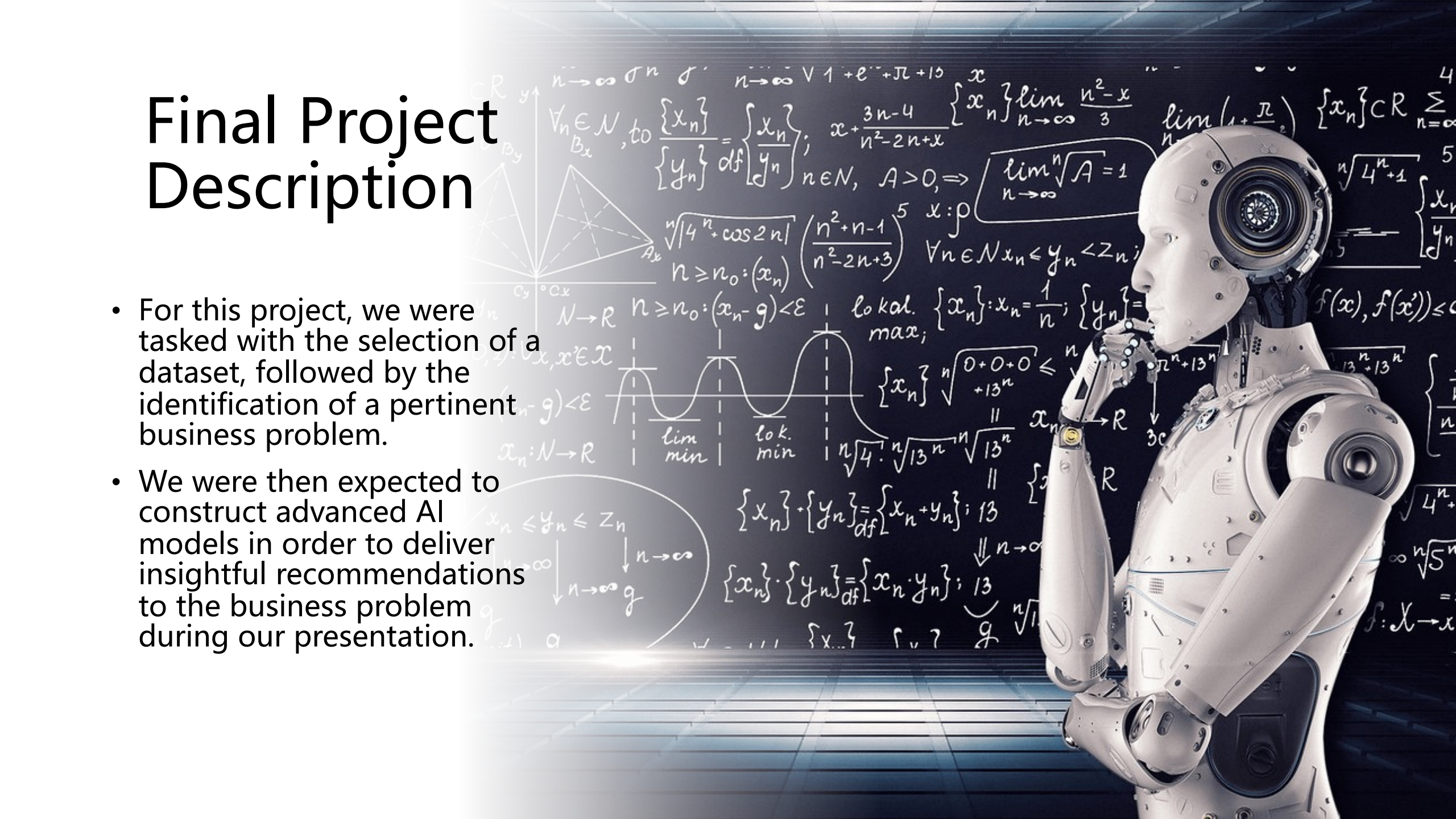
Haden Foster

Saurav S Borah

Anurati Kulkarni

Final Project Description

- For this project, we were tasked with the selection of a dataset, followed by the identification of a pertinent business problem.
- We were then expected to construct advanced AI models in order to deliver insightful recommendations to the business problem during our presentation.



I. Business Problem Identification:

What is a Superhost?

Superhosts on Airbnb are distinguished hosts recognized for consistently providing exceptional hospitality, earning them a special status that denotes a high level of reliability and guest satisfaction

Criteria for being a Superhost:

1. **Response Rate:** Maintain a response rate of 90% or higher, reflecting the host's responsiveness to guest inquiries and booking requests.
2. **Acceptance Rate:** Maintain an acceptance rate of at least 88%, indicating a high level of accommodation acceptance by the host.
3. **Cancellation Rate:** Ensure a cancellation rate of 1% or lower for confirmed reservations, showcasing commitment to hosting without cancellations.
4. **Hosting Frequency:** Host a minimum of 10 stays within the past year or accumulate reservations equivalent to at least 100 nights.
5. **Overall Rating:** Maintain an overall rating of 4.8 or higher, indicating consistently positive feedback from guests.

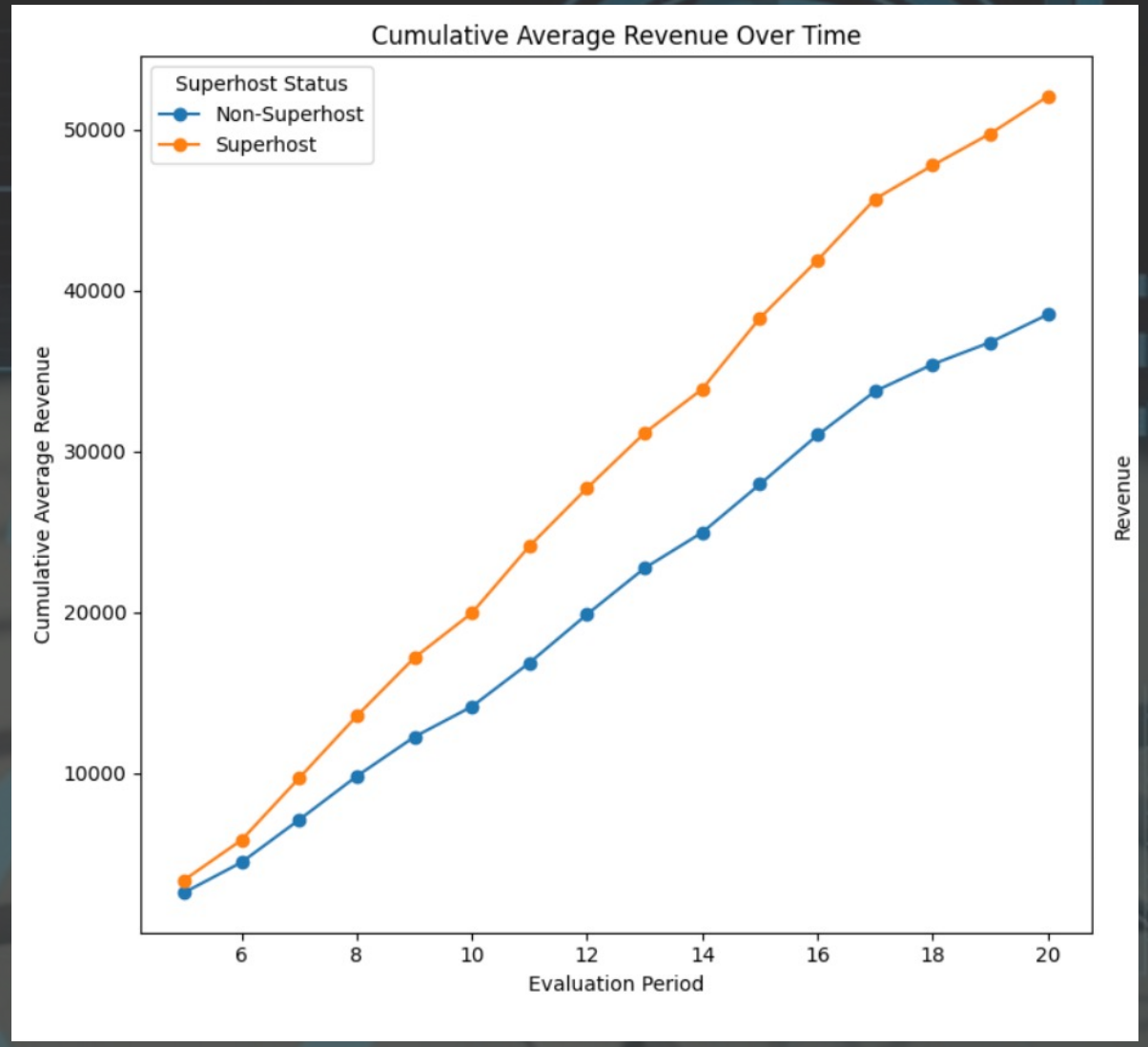
Unlocking the Impact of Superhost Status on Airbnb Revenue

- This study initially attempts to explore if Superhost status significantly influences host revenue.
- If the EDA proves that superhosts do in fact, make more revenue compared to non-superhosts, we then build a model predicting superhost status for the next period based on the variables in the dataset.
- We employ marketing strategies to target predicted superhosts to nudge them to achieve superhost status, thus benefiting all parties involved: the host, the customer and AirBnB.

II. EDA

- We have selected the Oakland dataset for out study to reduce compute time.
- EDA has been conducted with Python.
- Average Metrics for Superhosts vs Non-Superhosts are given below.

	Superhosts	Non-Superhosts
rating_ave_pastYear	4.894380	4.667348
numReviews_pastYear	54.185153	58.048639
numCancel_pastYear	0.009442	0.466140
occupancy_rate	0.244680	0.225986
revenue	3274.255250	2494.281171



II. EDA – T-test for Revenue, Superhosts vs Non-Superhosts

```python
import pandas as pd
from scipy import stats

# Drop NaN values in the revenue column
oakland_data = oakland_data.dropna(subset=['revenue'])

# Splitting the data into two groups
superhosts_revenue = oakland_data[oakland_data['Superhost'] == 1]['revenue']
non_superhosts_revenue = oakland_data[oakland_data['Superhost'] == 0]['revenue']

# Check variance
if superhosts_revenue.var() == 0 or non_superhosts_revenue.var() == 0:
    raise ValueError("One of the groups has no variance.")

# Since the dataset is large, we might skip the normality test and proceed directly to Mann-Whitney U test
test_stat, p_value = stats.mannwhitneyu(superhosts_revenue, non_superhosts_revenue, alternative='two-sided')

# Print results
print(f"Test Statistic: {test_stat}, P-value: {p_value}")
```

```
Test Statistic: 59457767.0, P-value: 3.807817905690481e-124
```

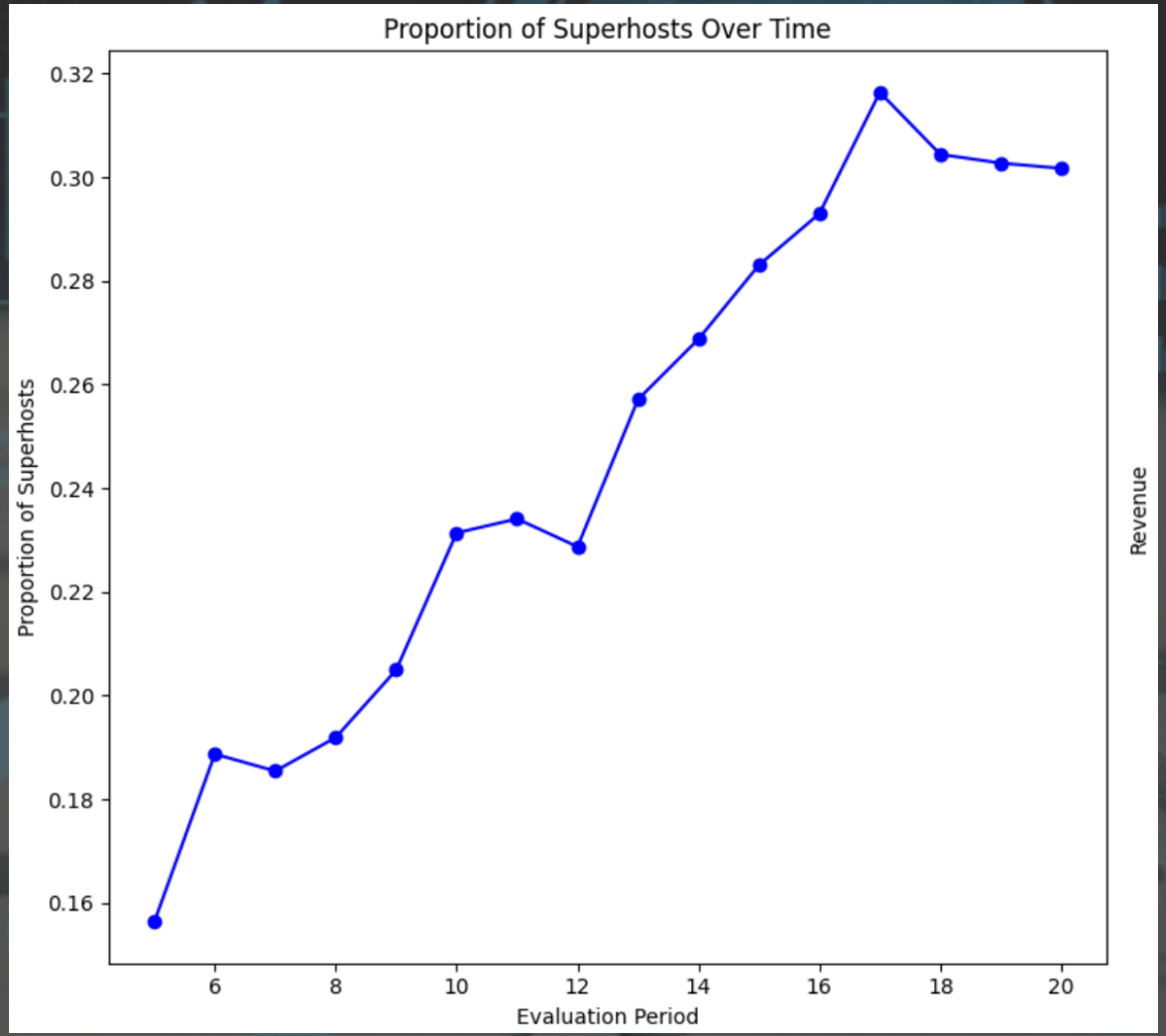
The results are highly significant with a very low p-value.

We reject the null hypothesis that there is no difference between the revenues of the two groups.

This is important for both hosts and Airbnb

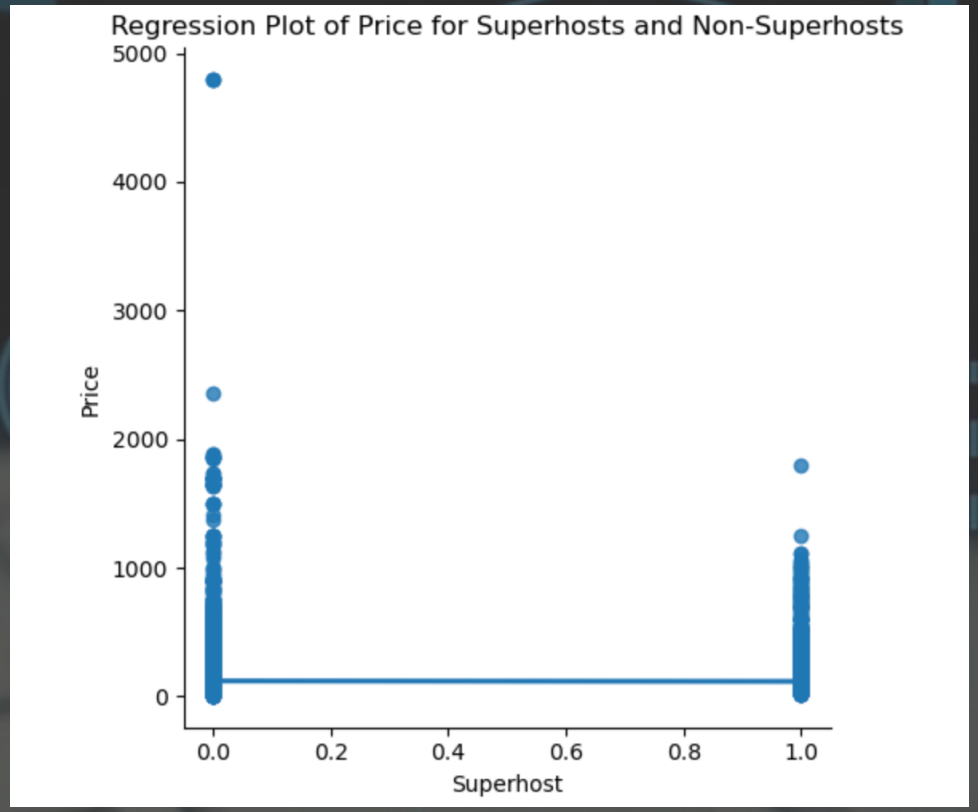
II. EDA

- The number of Superhosts over time generally increases
- Increase from ~18% to just under 32% Superhost proportion
- Occasional decreases in the proportion indicating those individuals weren't retained as superhosts
- Possible improvement in host service quality over time
- Potential changes in Airbnb's Superhost criteria or incentives
- Indication of enhanced guest experiences and host reliability



II. EDA

- From the regression analysis for Average Listed Price~ Superhost status, we observed that the price has a negative dependency on the Superhost status which is statistically significant.
- The graph on the right shows the Regression plot for the same.
- This analysis shows how the Superhost status benefits customers as well in addition to the host and Airbnb.



OLS Regression Results

Dep. Variable:	available_days_aveListedPrice	R-squared:	0.000
Model:	OLS	Adj. R-squared:	0.000
Method:	Least Squares	F-statistic:	4.747
Date:	Mon, 04 Dec 2023	Prob (F-statistic):	0.0294
Time:	21:01:11	Log-Likelihood:	-1.5988e+05
No. Observations:	25430	AIC:	3.198e+05
Df Residuals:	25428	BIC:	3.198e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	120.5935	0.990	121.815	0.000	118.653	122.534
Superhost	-3.8074	1.747	-2.179	0.029	-7.232	-0.382

Omnibus:	41079.449	Durbin-Watson:	0.360
Prob(Omnibus):	0.000	Jarque-Bera (JB):	62415842.129
Skew:	10.320	Prob(JB):	0.00
Kurtosis:	244.826	Cond. No.	2.42

II. EDA – T-test for Price, Superhosts vs Non-Superhosts

```python
import pandas as pd
from scipy import stats

# Drop NaN values in the price column
oakland_data = oakland_data.dropna(subset=['available_days_aveListedPrice'])

# Splitting the data into two groups
superhosts_price = oakland_data[oakland_data['Superhost'] == 1]['available_days_aveListedPrice']
non_superhosts_price = oakland_data[oakland_data['Superhost'] == 0]['available_days_aveListedPrice']

# Check variance
if superhosts_price.var() == 0 or non_superhosts_price.var() == 0:
    raise ValueError("One of the groups has no variance.")

# Since the dataset is large, we might skip the normality test and proceed directly to Mann-Whitney U test
test_stat, p_value = stats.mannwhitneyu(superhosts_price, non_superhosts_price, alternative='two-sided')

# Print results
print(f"Test Statistic: {test_stat}, P-value: {p_value}")
```

```
Test Statistic: 55963836.5, P-value: 2.4237385557939675e-52
```

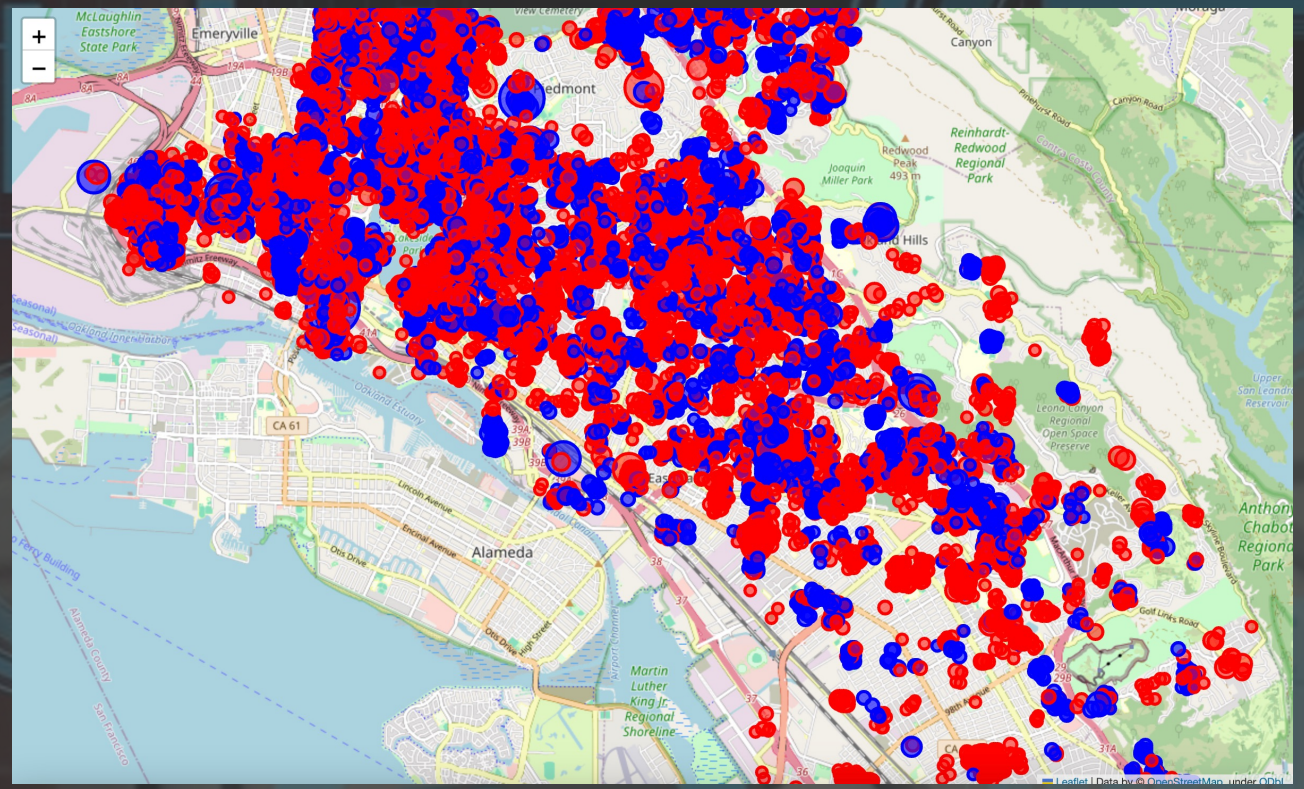
The results are highly significant with a very low p-value.

We reject the null hypothesis that there is no difference between the prices of the two groups.

This is important from the perspective of customers.

II. EDA

- To take this EDA further we created Maps to visualize the host distribution.
- We created maps showing all hosts, the top 1% of earners, and the bottom 1% of earners.
- The three different maps show that super hosts are beating most normal hosts in terms of revenue.
- The bottom 1% map illustrates that there are significantly more normal hosts underperforming.



III. Model Building

- We build a Random Forest Model to predict superhost status for the next period.
- We used the previous year features to train the model with current superhost status are the target variable. We used the present year attributes as the test features to predict the next period superhost status.

```
Best Parameters:  {'max_depth': 20, 'n_estimators': 100}
Best Accuracy on Entire Dataset:  0.9343706474013949
Feature Importance:
                                                Feature  Importance
1                    prev_host_is_superhost_in_period    0.270409
3                            prev_rating_ave_pastYear    0.110298
7                   prev_prop_5_StarReviews_pastYear    0.079841
6                       prev_num_5_star_Rev_pastYear    0.056889
5                             prev_numCancel_pastYear    0.051849
18                             prev_Number of Reviews    0.046365
19                                prev_Rating Overall    0.043691
4                           prev_numReviews_pastYear    0.040961
11                 prev_hostResponseAverage_pastYear    0.031907
9                             prev_numReserv_pastYear    0.029584
8                    prev_numReservedDays_pastYear    0.027991
10                 prev_hostResponseNumber_pastYear    0.022096
20                                      prev_revenue    0.021354
13                 prev_available_days_aveListedPrice    0.019778
12                            prev_available_days    0.018916
17                              prev_Nightly Rate    0.017179
14                               prev_booked_days    0.017062
2                           prev_scrapes_in_period    0.016797
15                       prev_booked_days_avePrice    0.016131
0                        prev_superhost_period_all    0.015662
24                          prev_time_to_date_mean    0.015226
21                             prev_occupancy_rate    0.015079
22                  prev_Nightly Rate_tractQuartile    0.005967
23  prev_available_days_aveListedPrice_tractQuartile    0.005941
16                          prev_Instantbook Enabled    0.003027
```

IV. Predictions

Classification Report:

	precision	recall	f1-score	support
0.0	0.95	0.97	0.96	5335
1.0	0.89	0.85	0.87	1766
accuracy			0.94	7101
macro avg	0.92	0.91	0.91	7101
weighted avg	0.94	0.94	0.94	7101

Accuracy: 0.9367694690888607

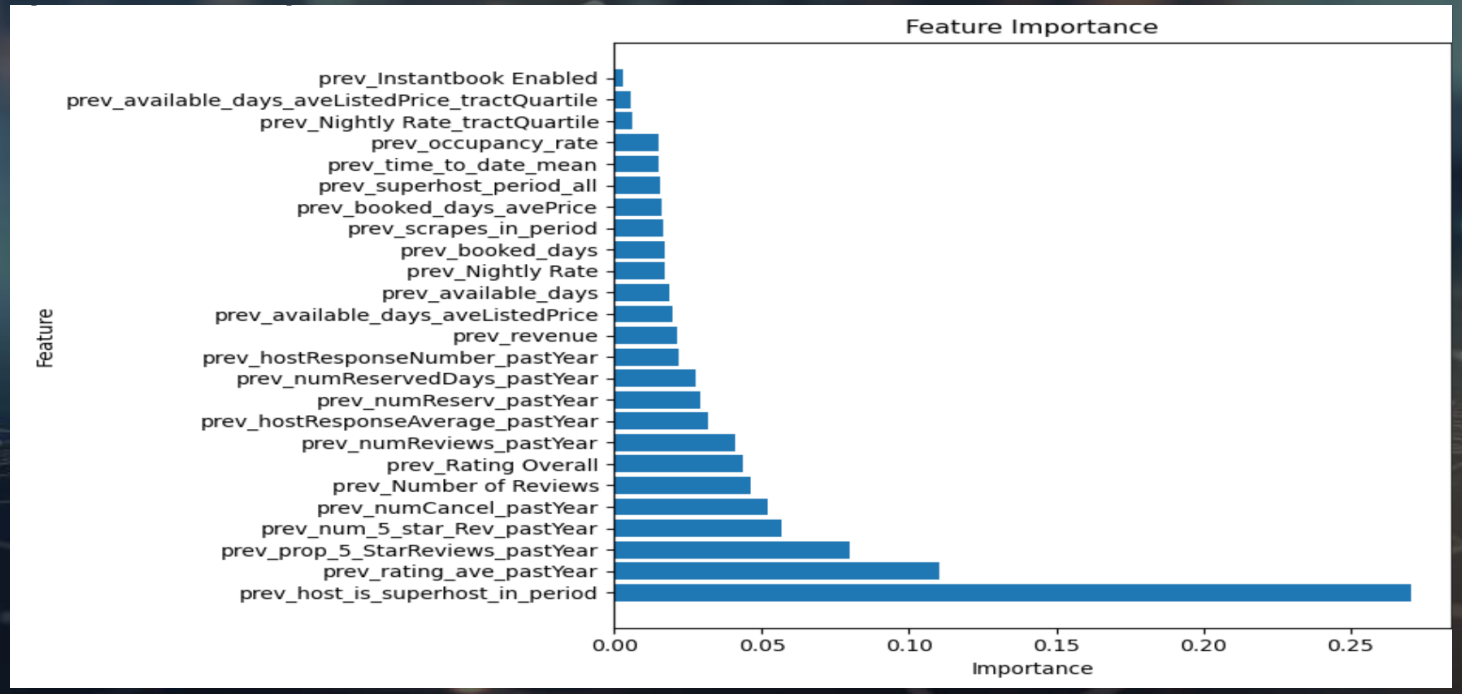
Predicted Superhost Status for Next Year:
[0. 0. 0. ... 0. 0. 0.]

The predicted Superhost status was added to the original Oakland dataset as one more column.

The model performs reasonably well on the sparse target outcome also as the precision and recall Superhost==1 is also high.

V. Inferences and Recommendations

- We see that a lot of indirect indicators of performance such as number of reviews, ratings, revenue and price are very important features in predicting "Superhosts" as some of the criteria for assigning "Superhosts" itself, such as host response rate and proportion of five-star reviews.
- Our primary recommendation is that we track the important attributes of predicted "Superhosts' and design a notification system that encourages hosts to improve on relevant features to perform better.
- Additional Recommendations:
    1. For superhosts to maintain their status, we can highlight the criterion with the highest feature importance out of the 5 superhost criteria (can vary with region/locality). Since in this case, it is the reviews/ratings criteria, the superhosts can be advised by Airbnb to follow certain guidelines/ways to improve the same.
    2. For hosts who want to become superhosts, we can suggest they try improve the indirect indicators with a relatively higher feature importance, if they are facing difficulty reaching the cutoff levels of the 5 direct criteria.



Marketing Strategy: Notification System

Notification for Superhosts at Risk:

Subject: Urgent: Protect Your Superhost Status!

Dear [Host's Name],

We trust this message finds you well. Our data analytics reveal a concerning trend that could impact your Superhost status. We understand the importance of your hosting efforts and are here to assist you in safeguarding your esteemed Superhost standing.

Consider the potential drawbacks of losing your Superhost status:

Revenue Impact: Superhosts in Oakland consistently outperform non-superhosts in revenue. A lapse in Superhost status may lead to a decline in earnings.

Visibility Concerns: Superhosts enjoy enhanced visibility on Airbnb, attracting more guests. Losing your Superhost status might result in reduced exposure and fewer booking opportunities.

Profit Retention Changes: Superhosts retain a higher percentage of profits. Falling out of Superhost status may alter your profit retention, affecting your overall financial gains.

We urge you to take immediate action to address any potential issues and maintain the exceptional hosting standards that earned you the Superhost distinction. Your commitment to excellence is vital to ensuring continued success on Airbnb.

If you have any questions or require assistance, please don't hesitate to reach out to our support team.

Best regards,
[Your Name]
Airbnb Host Success Team

Notification for Prospective Superhosts:

Subject: Elevate Your Hosting Status to Superhost Excellence!

Dear [Host's Name],

Congratulations! 🌟 Our data projections indicate that you are on the cusp of achieving Superhost status on Airbnb! Your exceptional hosting practices have positioned you as a frontrunner in providing outstanding experiences for guests.

Here are some perks awaiting you as a Superhost:

Increased Revenue: Superhosts in the Oakland area experience higher revenue compared to their non-superhost counterparts. Elevate your earnings by joining the ranks of esteemed Superhosts.

Enhanced Visibility: Superhosts enjoy heightened visibility on the Airbnb platform, attracting more guests to your listing. This increased exposure can lead to more bookings and opportunities to showcase your exceptional hospitality.

Higher Profit Retention: As a Superhost, you'll benefit from a higher percentage of retained profits, maximizing the financial rewards of your hosting efforts.

Keep up the fantastic work! We believe that achieving Superhost status will not only boost your hosting profile but also enhance your overall Airbnb experience.

Best regards,
[Your Name]
Airbnb Host Success Team

Conclusion

- Key Finding: Superhost Status Matters to AirBnB, Hosts and Customers.
  Metrics for superhosts consistently better than non-superhosts.

- Strategic Solution: Notification System
  - Proposing an intelligent notification system to empower hosts:
    - Alert hosts on the verge of gaining Superhost status.
    - Notify hosts at risk of losing Superhost status.

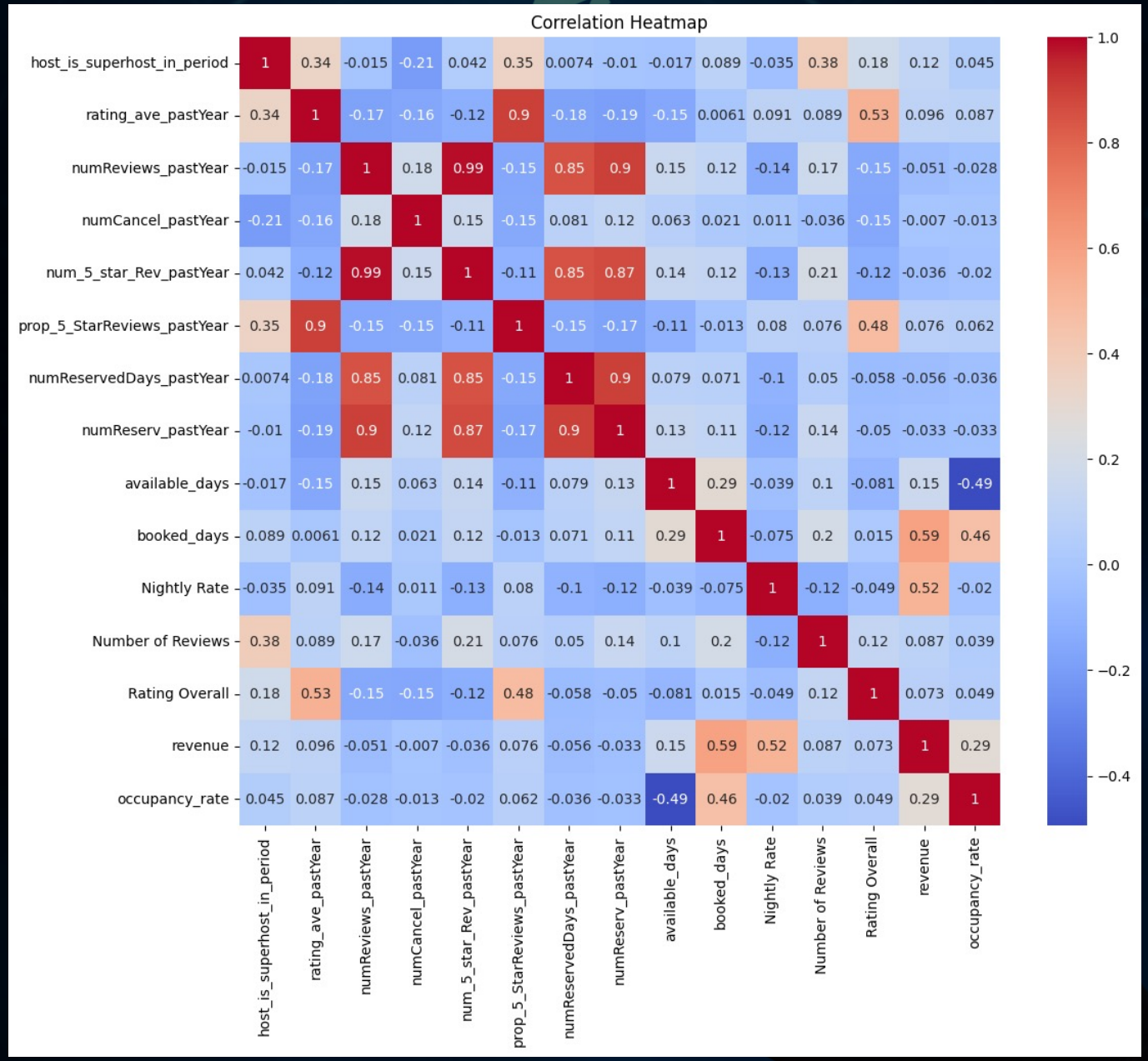
- Project Impact
  - This project worked off a data-driven insight about superhost data and presented a potential recommendation based on it to benefit AirBnB.

Appendix A: Code for EDA

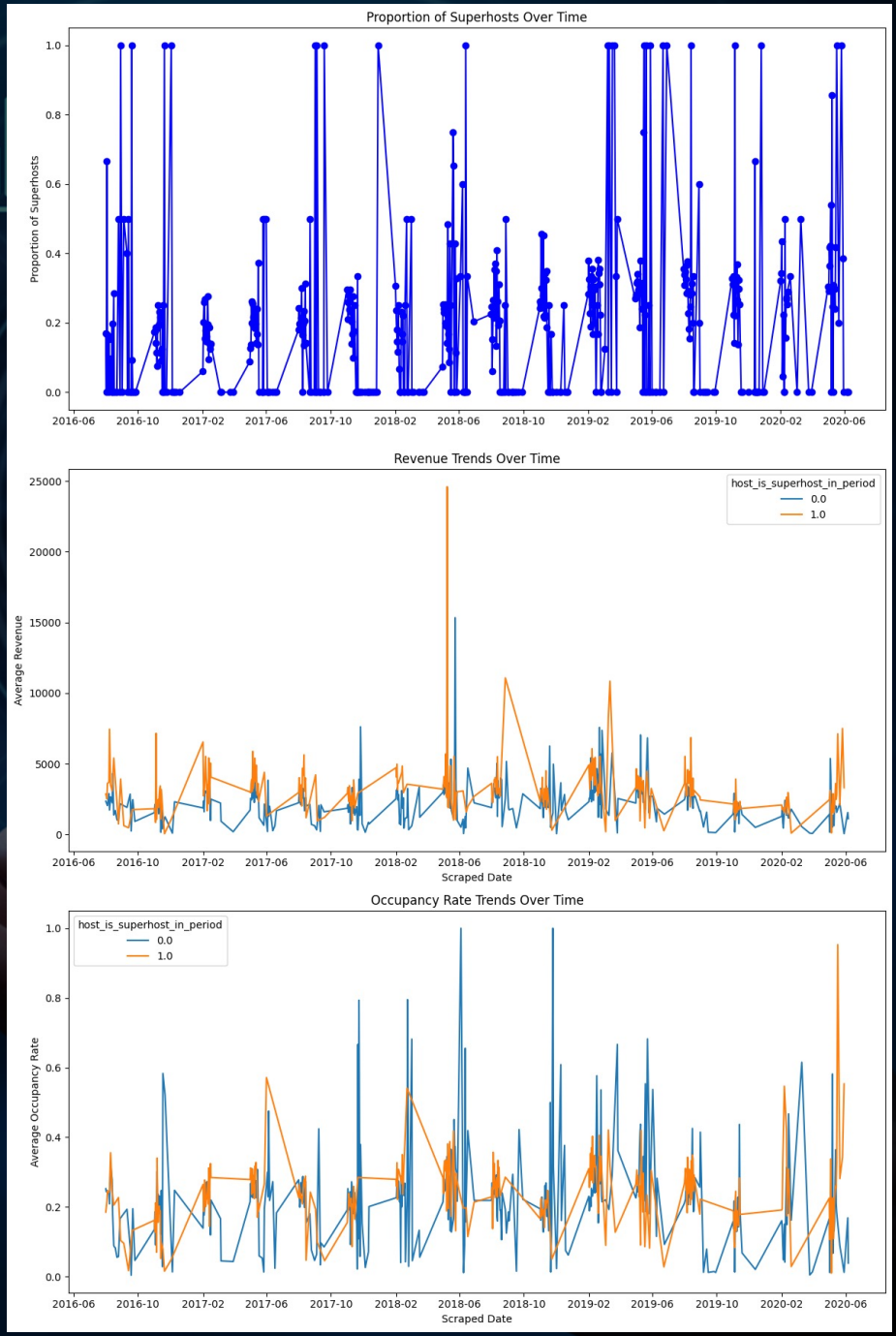
```
# Creating subsets for Superhosts and non-Superhosts
superhosts = oakland_data[oakland_data['host_is_superhost_in_period'] == 1]
non_superhosts = oakland_data[oakland_data['host_is_superhost_in_period'] == 0]

# Key metrics for comparison
metrics = ['rating_ave_pastYear', 'numReviews_pastYear', 'numCancel_pastYear',
           'occupancy_rate', 'revenue']

# Calculating average values for each metric for both groups
avg_metrics_superhosts = superhosts[metrics].mean()
avg_metrics_non_superhosts = non_superhosts[metrics].mean()

# Creating a DataFrame for easier comparison
comparison_df = pd.DataFrame({'Superhosts': avg_metrics_superhosts, 'Non-Superhosts': avg_metrics_non_superhosts})

comparison_df
```

	Superhosts	Non-Superhosts
rating_ave_pastYear	4.894380	4.667348
numReviews_pastYear	54.185153	58.048639
numCancel_pastYear	0.009442	0.466140
occupancy_rate	0.244680	0.225986
revenue	3274.255250	2494.281171

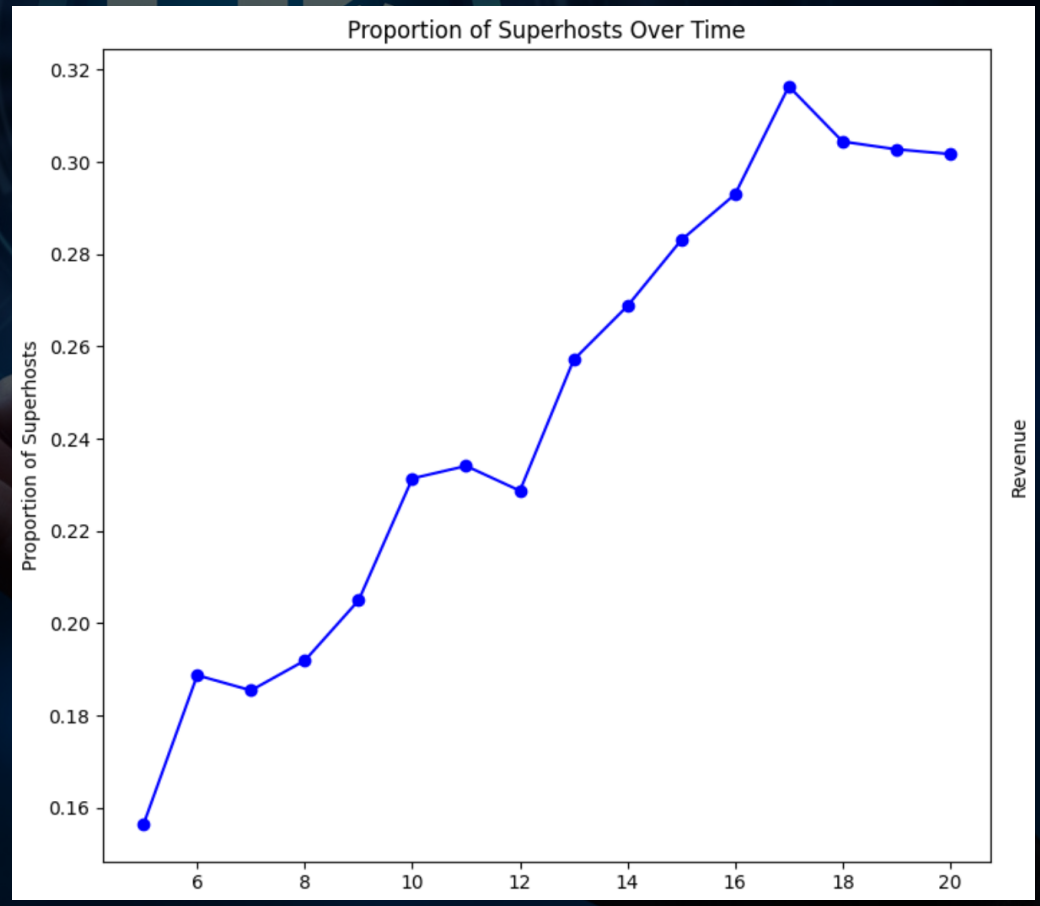
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Selecting relevant columns for correlation analysis
relevant_columns = [
    'host_is_superhost_in_period',
    'rating_ave_pastYear',
    'numReviews_pastYear',
    'numCancel_pastYear',
    'num_5_star_Rev_pastYear',
    'prop_5_StarReviews_pastYear',
    'numReservedDays_pastYear',
    'numReserv_pastYear',
    'available_days',
    'booked_days',
    'Nightly Rate',
    'Number of Reviews',
    'Rating Overall',
    'revenue',
    'occupancy_rate'
]

# Creating a subset of the data with the relevant columns
subset_data = oakland_data[relevant_columns]

# Calculating the correlation matrix
correlation_matrix = subset_data.corr()

# Plotting the heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



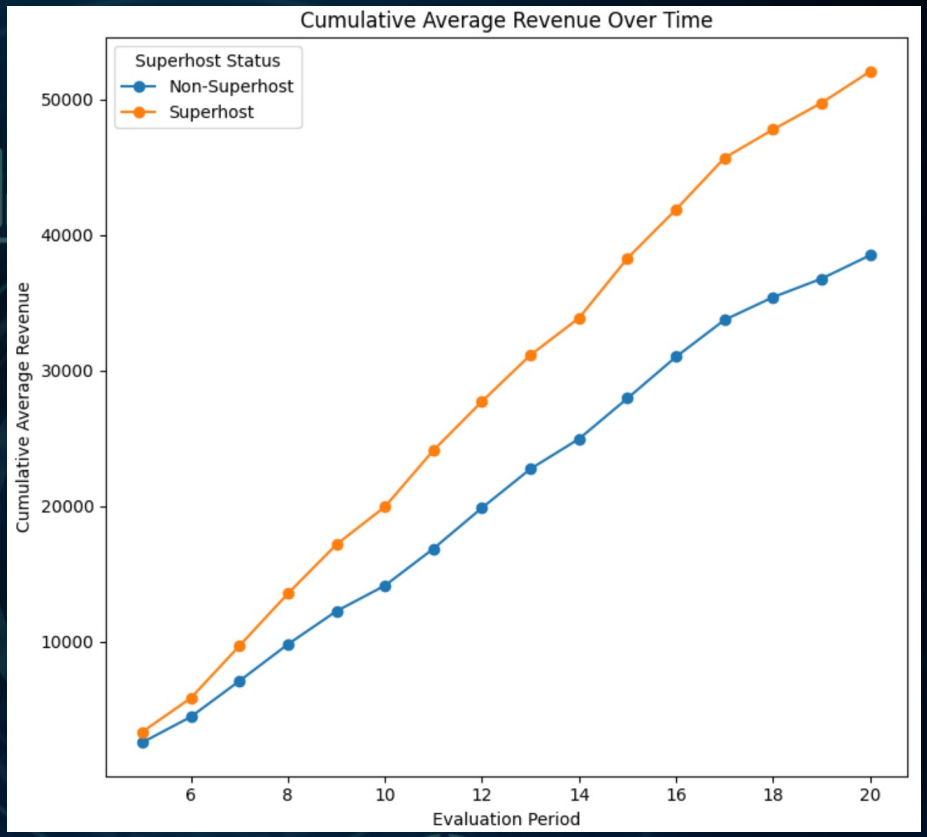
```python
# Converting 'Scraped Date' to datetime for trend analysis
oakland_data['Scraped Date'] = pd.to_datetime(oakland_data['Scraped Date'])

# Grouping data by date and calculating the proportion of Superhosts
date_grouped = oakland_data.groupby('Scraped Date').agg({'host_is_superhost_in_period': 'mean'})

# Revenue and occupancy rate trends
revenue_trends = oakland_data.groupby(['Scraped Date', 'host_is_superhost_in_period']).agg({'revenue': 'mean'})
occupancy_trends = oakland_data.groupby(['Scraped Date', 'host_is_superhost_in_period']).agg({'occupancy_rate': 'mean'})

# Resetting index for plotting
revenue_trends = revenue_trends.reset_index()
occupancy_trends = occupancy_trends.reset_index()

# Adjusting the x-axis to make it more legible in the time series plots

# Plotting
fig, axes = plt.subplots(3, 1, figsize=(12, 18))

# Proportion of Superhosts over time
axes[0].plot(date_grouped.index, date_grouped['host_is_superhost_in_period'], marker='o', color='blue')
axes[0].set_title('Proportion of Superhosts Over Time')
axes[0].set_ylabel('Proportion of Superhosts')
axes[0].xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
axes[0].xaxis.set_major_locator(mdates.MonthLocator(interval=4))  # Adjusting interval for better readability

# Revenue trends
sns.lineplot(data=revenue_trends, x='Scraped Date', y='revenue', hue='host_is_superhost_in_period', ax=axes[1])
axes[1].set_title('Revenue Trends Over Time')
axes[1].set_ylabel('Average Revenue')
axes[1].xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
axes[1].xaxis.set_major_locator(mdates.MonthLocator(interval=4))  # Adjusting interval for better readability

# Occupancy rate trends
sns.lineplot(data=occupancy_trends, x='Scraped Date', y='occupancy_rate', hue='host_is_superhost_in_period', ax=axes[2])
axes[2].set_title('Occupancy Rate Trends Over Time')
axes[2].set_ylabel('Average Occupancy Rate')
axes[2].xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
axes[2].xaxis.set_major_locator(mdates.MonthLocator(interval=4))  # Adjusting interval for better readability

plt.tight_layout()
plt.show()
```

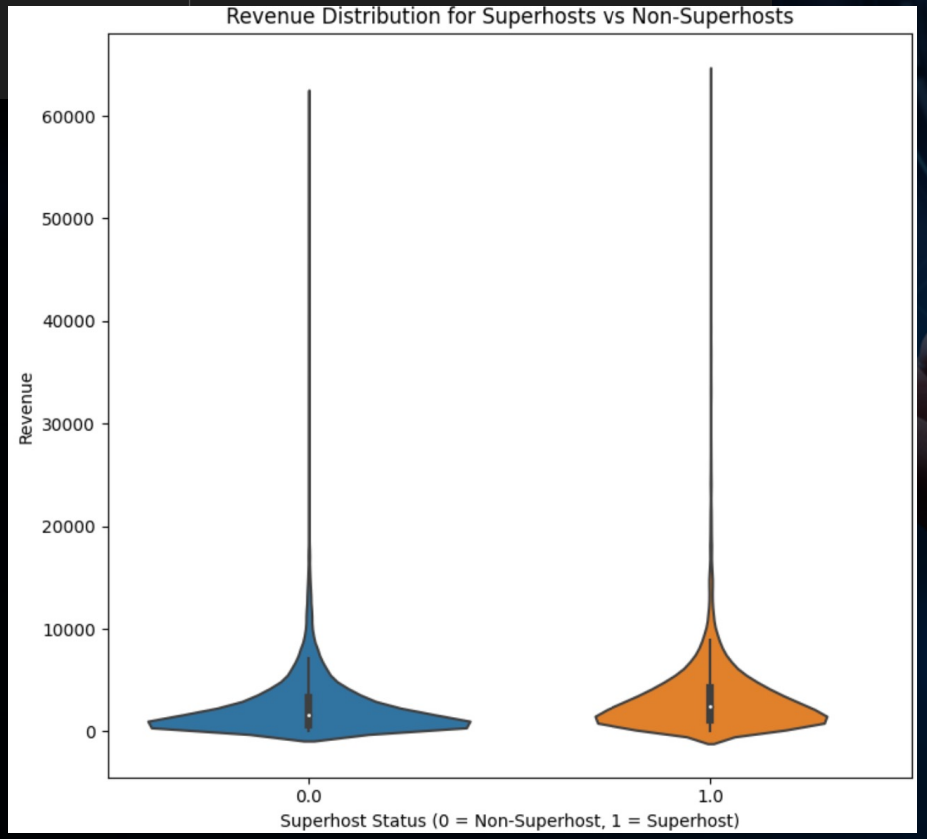


```python
# Converting 'Scraped Date' to datetime for trend analysis
oakland_data['Scraped Date'] = pd.to_datetime(oakland_data['Scraped Date'])

# Grouping data by date and calculating the proportion of Superhosts
date_grouped = oakland_data.groupby('Scraped Date').agg({'host_is_superhost_in_period': 'mean'})

# Revenue and occupancy rate trends
revenue_trends = oakland_data.groupby(['Scraped Date', 'host_is_superhost_in_period']).agg({'revenue': 'mean'})
occupancy_trends = oakland_data.groupby(['Scraped Date', 'host_is_superhost_in_period']).agg({'occupancy_rate': 'mean'})

# Resetting index for plotting
revenue_trends = revenue_trends.reset_index()
occupancy_trends = occupancy_trends.reset_index()

# Adjusting the x-axis to make it more legible in the time series plots

# Plotting
fig, axes = plt.subplots(3, 1, figsize=(12, 18))

# Proportion of Superhosts over time
axes[0].plot(date_grouped.index, date_grouped['host_is_superhost_in_period'], marker='o', color='blue')
axes[0].set_title('Proportion of Superhosts Over Time')
axes[0].set_ylabel('Proportion of Superhosts')
axes[0].xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
axes[0].xaxis.set_major_locator(mdates.MonthLocator(interval=4))  # Adjusting interval for better readability
```



```python
# Visualization 5: Cumulative Revenue Over Time for Superhosts vs Non-Superhosts
# Grouping by superhost status and evaluation period, then calculating cumulative average revenue
cumulative_revenue = oakland_data.groupby(['superhost_period_all', 'host_is_superhost_in_period'])['revenue'].mean().groupby(level=1).cumsum()

# Visualization 6: Revenue Distribution as Violin Plot
# Violin plot for revenue distribution for Superhosts and Non-Superhosts
violin_data = oakland_data[['revenue', 'host_is_superhost_in_period']].dropna()

# Creating the plots
plt.figure(figsize=(15, 7))

# Cumulative Revenue Over Time
plt.subplot(1, 2, 1)
cumulative_revenue.unstack().plot(kind='line', marker='o', ax=plt.gca())
plt.title('Cumulative Average Revenue Over Time')
plt.xlabel('Evaluation Period')
plt.ylabel('Cumulative Average Revenue')
plt.legend(title='Superhost Status', labels=['Non-Superhost', 'Superhost'])

# Revenue Distribution as Violin Plot
plt.subplot(1, 2, 2)
sns.violinplot(data=violin_data, x='host_is_superhost_in_period', y='revenue')
plt.title('Revenue Distribution for Superhosts vs Non-Superhosts')
plt.xlabel('Superhost Status (0 = Non-Superhost, 1 = Superhost)')
plt.ylabel('Revenue')

plt.tight_layout()
plt.show()
```





Appendix B: Code for T-tests

```
import pandas as pd
from scipy import stats

# Drop NaN values in the price column
oakland_data = oakland_data.dropna(subset=['available_days_aveListedPrice'])

# Splitting the data into two groups
superhosts_price = oakland_data[oakland_data['Superhost'] == 1]['available_days_aveListedPrice']
non_superhosts_price = oakland_data[oakland_data['Superhost'] == 0]['available_days_aveListedPrice']

# Check variance
if superhosts_price.var() == 0 or non_superhosts_price.var() == 0:
    raise ValueError("One of the groups has no variance.")

# Since the dataset is large, we might skip the normality test and proceed directly to Mann-Whitney U test
test_stat, p_value = stats.mannwhitneyu(superhosts_price, non_superhosts_price, alternative='two-sided')

# Print results
print(f"Test Statistic: {test_stat}, P-value: {p_value}")
```

```
Test Statistic: 55963836.5, P-value: 2.4237385557939675e-52
```

```python
import pandas as pd
from scipy import stats

# Drop NaN values in the revenue column
oakland_data = oakland_data.dropna(subset=['revenue'])

# Splitting the data into two groups
superhosts_revenue = oakland_data[oakland_data['Superhost'] == 1]['revenue']
non_superhosts_revenue = oakland_data[oakland_data['Superhost'] == 0]['revenue']

# Check variance
if superhosts_revenue.var() == 0 or non_superhosts_revenue.var() == 0:
    raise ValueError("One of the groups has no variance.")

# Since the dataset is large, we might skip the normality test and proceed directly to Mann-Whitney U test
test_stat, p_value = stats.mannwhitneyu(superhosts_revenue, non_superhosts_revenue, alternative='two-sided')

# Print results
print(f"Test Statistic: {test_stat}, P-value: {p_value}")
```

```
Test Statistic: 59457767.0, P-value: 3.807817905690481e-124
```

Appendix C: Code for Map

```
#100ofOaklandHosts

import pandas as pd
import folium
import ipywidgets as widgets
from IPython.display import display, clear_output
import random

# Load a subset of the dataset
df = pd.read_csv("airbnb_Oakland.csv")
df_subset = df  # Adjust the number based on your dataset size

latitude_column = 'Latitude'
longitude_column = 'Longitude'
revenue_column = 'revenue'  # Correct the revenue column name
property_type_column = 'Property Type'
review_score_column = 'rating_ave_pastYear'  # Correctly used as a string
superhost_column = 'host_is_superhost_in_period'  # Assuming this is the superhost column name

# Calculate the maximum revenue for scaling the circle size
max_revenue = df[revenue_column].max()

# Function to add a small random offset to latitude and longitude
def add_offset(lat, lon, offset=0.0008):
    lat_offset = lat + random.uniform(-offset, offset)
    lon_offset = lon + random.uniform(-offset, offset)
    return lat_offset, lon_offset
```

```python
def create_map(df):
    mean_latitude = df[latitude_column].mean()
    mean_longitude = df[longitude_column].mean()
    temp_map = folium.Map(location=[mean_latitude, mean_longitude], zoom_start=12)

    for index, row in df.iterrows():
        superhost_status = 'Yes' if row[superhost_column] else 'No'

        # Check if initial review column has a value, otherwise use rating_ave_pastYear
        if pd.notna(row['rating_ave_pastYear']):
            review_score = row['rating_ave_pastYear']
        else:
            review_score = row['prev_rating_ave_pastYear']

        # Format the review score
        formatted_score = "{:.2f}".format(review_score) if pd.notna(review_score) else "N/A"

        # Set marker color based on superhost status
        marker_color = 'blue' if row[superhost_column] else 'red'

        # Calculate the circle radius based on revenue (scaled relative to max revenue)
        circle_radius = 5 + 20 * (row[revenue_column] / max_revenue)

        # Add a small random offset to latitude and longitude for visualization
        lat_offset, lon_offset = add_offset(row[latitude_column], row[longitude_column])

        popup_text = f"Property Type: {row[property_type_column]}<br>Revenue: {row[revenue_colu

        popup = folium.Popup(popup_text, max_width=300)

        folium.CircleMarker(
            location=[lat_offset, lon_offset],  # Use offset location
            popup=popup,
            radius=circle_radius,
            color=marker_color,
            fill=True,
            fill_color=marker_color,
            fill_opacity=0.6,
        ).add_to(temp_map)
    return temp_map

# Save the Folium map as an HTML file
def save_map_as_html(map_object, file_path):
    map_object.save(file_path)

map_output = widgets.Output()
```

```python
with map_output:
    m = create_map(df)
    display(m)  # Display the map with different marker sizes based on revenue

widget_container = widgets.VBox([map_output])
display(widget_container)

# Specify the file path where you want to save the HTML file
html_file_path = "OaklandHosts100%.html"

# Save the Folium map as an HTML file
save_map_as_html(m, html_file_path)
```

Appendix D: Code for Model

```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.impute import SimpleImputer
import seaborn as sns

# Load your dataset
df = pd.read_csv("airbnb_Oakland.csv")

# Use regular expression to filter column names. Use only the previous year predictor that starts with prev_
pattern = re.compile(r'^prev_')
prev_year_features = [col for col in df.columns if pattern.match(col)]

# Remove the specified features from the list of previous year features
features_to_remove = ['prev_host_is_superhost', 'prev_host_is_superhost1', 'prev_host_is_superhost2', 'prev_year_superhos
prev_year_features = [col for col in prev_year_features if col not in features_to_remove]

# Extract features and target variable
X = df[prev_year_features]
y = df["host_is_superhost_in_period"]

# Create a SimpleImputer to handle missing values (replace NaN with the mean, for example)
imputer = SimpleImputer(strategy='mean')  # You can choose 'mean', 'median', 'most_frequent', or a constant value

# Fit the imputer on the entire dataset and transform
X_imputed = imputer.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_imputed, y, test_size=0.2, random_state=42)

# Create a Random Forest classifier
clf = RandomForestClassifier()
```

```
# Define the parameter grid for GridSearchCV
param_grid = {
    'n_estimators': [100, 150],
    'max_depth': [10, 20],
}

# Perform GridSearchCV
grid_search = GridSearchCV(estimator=clf, param_grid=param_grid, scoring='accuracy', cv=5)
grid_search.fit(X_train, y_train)

# Print the best parameters and the corresponding accuracy on the entire dataset
print("Best Parameters: ", grid_search.best_params_)
print("Best Accuracy on Entire Dataset: ", grid_search.best_score_)

# Make predictions on the imputed testing data
y_pred = grid_search.predict(X_test)

# Feature Importance
feature_importances = grid_search.best_estimator_.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': prev_year_features, 'Importance': feature_importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# Print the feature importances
print("Feature Importance:")
print(feature_importance_df)

# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Plot the feature importances and the confusion matrix side by side
plt.figure(figsize=(15, 6))

# Feature Importance Plot
plt.subplot(1, 2, 1)
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importance')
```

Appendix D: Code for Model

```
# Feature Importance Plot
plt.subplot(1, 2, 1)
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importance')

# Prediction in current year features
# Remove "prev_" from the column names to get the present year features
present_year_features = [col.replace("prev_", "") for col in prev_year_features]

# Extract features for the new dataset (next year's data) and handle feature name mismatches
X_new = df[present_year_features]
X_new = X_new.rename(columns=dict(zip(X_new.columns, prev_year_features)))

# Impute missing values using the same imputer that was fitted on the training data
X_new_imputed = imputer.transform(X_new)

# Make predictions on the new imputed data
y_new_pred = grid_search.predict(X_new_imputed)

# Print the predicted values
print("\nPredicted Superhost Status for Next Year:")
print(y_new_pred)

# Append y_new_pred to the original dataframe as Predicted_Future_Superhost_Status
df['Predicted_Future_Superhost_Status'] = y_new_pred

# Save the dataframe with predictions to a new CSV file
df.to_csv("airbnb_Oakland_Predicted_Superhost.csv", index=False)

plt.tight_layout()
plt.show()
```

Appendix E: Articles Used for Research

- https://www.igms.com/airbnb-superhost/#:~:text=Although%20Superhosts%20have%20a%20lower,daily%20revenue%20than%20regular%20hosts.
- https://www.nerdwallet.com/article/travel/what-does-superhost-mean-on-airbnb
- https://www.airbnb.com/help/article/828
- https://www.gobankingrates.com/money/making-money/airbnb-superhost/
- https://lovethemaldives.com/wiki/how-much-does-a-superhost-make-on-airbnb
- https://davidanthonyscott.medium.com/is-being-an-airbnb-superhost-worth-it-we-think-not-c0663bfc9e71
- https://airbtics.com/airbnb-superhost-criteria/
- https://www.mashvisor.com/blog/what-does-superhost-mean-on-airbnb/
- https://www.mashvisor.com/blog/how-start-airbnb-business/
- https://helplama.com/airbnb-statistics-revenue-and-usage/