PROJECT (3)

August 1, 2024

1 Project Requirements

- Final dataset should include 3 changes (deleting, inserting, adding, etc.)
- Pandas, matplot, and scikit packages should be part of the solution.
- Able to display results based on scenarios (For example, if I have different output, how the result will change. It could be a graph or display of results in row format)
- Minimum of 5 charts for insights. Use Python libraries to draw graphs.
- Total ten insights from the dataset

1.0.1 The Final Report should include the following.

- Description of project
- Overview of the dataset
- Exploratory analysis
- Details of insights will be identified from the dataset.
- Python programs for all 10 insights
- Minimum of 5 charts

1.1 1. Abstract

Airbnb is a prominent rental property service catering to vacationers, winter avoiders, getaway seekers, and travelers worldwide. This project delves into the insights behind Airbnb listings in the United States. We will analyze the top cities with the most rental properties, evaluate the pricing trends across different states, and develop a predictive model for estimating rental prices.

[1]:
```python
#Import the required packages:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#import data using pandas

df = pd.read_csv("airbnb_listings_usa.csv", low_memory=False)
```

1.2 Overview of Data and Exploratory Data Analysis (EDA)

To begin we need to get a better understanding of the data. We will look at things such as number of rows, columns, descriptive varaiables to help describe the data, and even visualizations to get a better understanding of how the data is formatted in the file. Below is the column names and descriptions from the dataset. To view the dataset plesae see (https://www.kaggle.com/datasets/tamle507/airbnb-listings-usa)

Column Name	Description
`id`	Unique identifier for the listing
`name`	Name of the listing
`host_id`	Unique identifier for the host
`host_name`	Name of the host
`neighbourhood_group`	Broad geographical area of the listing (e.g., borough or city district)
`neighbourhood`	Specific neighbourhood where the listing is located
`latitude`	Latitude coordinate of the listing
`longitude`	Longitude coordinate of the listing
`room_type`	Type of room offered (e.g., entire home/apt, private room, shared room)
`price`	Price per night for the listing (in local currency)
`minimum_nights`	Minimum number of nights required to book the listing
`number_of_reviews`	Total number of reviews the listing has received
`last_review`	Date of the last review received
`reviews_per_month`	Average number of reviews per month
`calculated_host_listings_count`	Total number of listings the host has
`availability_365`	Number of days the listing is available for booking in a year
`number_of_reviews_ltm`	Number of reviews in the last twelve months
`license`	License information for the listing (if applicable)
`state`	State where the listing is located
`city`	City where the listing is located

[2]:
```python
#head, this will show us the top 5 rows in the dataset. Giving us a good idea
↪of the layout of columns and what they may include

df.head()
```

[2]:
```
   Unnamed: 0      id                                            name  \
0           0  183319                  Panoramic Ocean View Venice Beach
1           1     109  Amazing bright elegant condo park front *UPGRA…
2           2   51307  Spanish Bungalow Guest House LA CA. 30 plus ni…
3           3  184314                    Boho Chic Flat..Steps to Beach!
4           4   51498  Guest House With Its Own Entrance/Exit and Hot…

   host_id  host_name  neighbourhood_group    neighbourhood  latitude  \
0   867995  Barbara X  City of Los Angeles           Venice  33.99211
1      521      Paolo         Other Cities      Culver City  33.98301
```

```
2   235568      David  City of Los Angeles  Atwater Village  34.12206
3   884031     Ashley  City of Los Angeles           Venice  33.97487
4   236758        Bay  City of Los Angeles        Mar Vista  34.00389

   longitude        room_type  …  minimum_nights  number_of_reviews  \
0 -118.47600  Entire home/apt  …              30                  3
1 -118.38607  Entire home/apt  …              30                  2
2 -118.26783  Entire home/apt  …              30                138
3 -118.46312  Entire home/apt  …              30                 30
4 -118.44126  Entire home/apt  …               3                378

   last_review reviews_per_month  calculated_host_listings_count  \
0   2019-02-25              0.02                               2
1   2016-05-15              0.01                               1
2   2020-12-13              0.98                               2
3   2017-12-24              0.22                               1
4   2022-08-21              2.60                               1

   availability_365  number_of_reviews_ltm       license state         city
0                 0                      0           NaN    CA  Los Angeles
1               139                      0           NaN    CA  Los Angeles
2               224                      0           NaN    CA  Los Angeles
3                 0                      0           NaN    CA  Los Angeles
4               348                     41  HSR19-001336    CA  Los Angeles

[5 rows x 21 columns]
```

[3]:
```
#this command allows us to quickly see all column names in the dataset
df.columns
```

[3]:
```
Index(['Unnamed: 0', 'id', 'name', 'host_id', 'host_name',
       'neighbourhood_group', 'neighbourhood', 'latitude', 'longitude',
       'room_type', 'price', 'minimum_nights', 'number_of_reviews',
       'last_review', 'reviews_per_month', 'calculated_host_listings_count',
       'availability_365', 'number_of_reviews_ltm', 'license', 'state',
       'city'],
      dtype='object')
```

[4]:
```
#This will show the shape in (rows, column) order
df.shape
```

[4]:
```
(325858, 21)
```

[5]:
```
#use of dtypes shows the type of data in each column, either int64 or integer/
 ↪numeric, object/string, or float64 which is decimal value
df.dtypes
```

[5]:
```
Unnamed: 0                          int64
id                                  int64
name                               object
host_id                             int64
host_name                          object
neighbourhood_group                object
neighbourhood                      object
latitude                          float64
longitude                         float64
room_type                          object
price                               int64
minimum_nights                      int64
number_of_reviews                   int64
last_review                        object
reviews_per_month                 float64
calculated_host_listings_count      int64
availability_365                    int64
number_of_reviews_ltm               int64
license                            object
state                              object
city                               object
dtype: object
```

[6]:
```
#Describe the numeric columns in the dataframe
df.describe()
```

[6]:
```
          Unnamed: 0            id       host_id       latitude  \
count  325858.000000  3.258580e+05  3.258580e+05  325858.000000
mean   162928.500000  1.541106e+17  1.446528e+08      34.676058
std     94067.246346  2.736013e+17  1.449951e+08       6.213029
min         0.000000  1.090000e+02  2.300000e+01      18.920250
25%     81464.250000  2.394619e+07  2.311836e+07      32.775202
50%    162928.500000  4.511097e+07  9.320864e+07      34.102360
75%    244392.750000  5.420558e+07  2.388338e+08      39.948015
max    325857.000000  7.251653e+17  4.810023e+08      47.748000

           longitude          price  minimum_nights  number_of_reviews  \
count  325858.000000  325858.000000   325858.000000      325858.000000
mean     -106.354815     284.915304       13.430175          39.457850
std        24.176674     835.569711       28.783033          75.724832
min      -159.714620       0.000000        1.000000           0.000000
25%      -118.410259      97.000000        2.000000           1.000000
50%      -117.131590     159.000000        3.000000           9.000000
75%       -82.549423     275.000000       30.000000          42.000000
max       -70.913250  100000.000000     1250.000000        2600.000000

       reviews_per_month  calculated_host_listings_count  availability_365  \
```

```
count       263166.00000                    325858.000000      325858.000000
mean             1.69220                        27.108900          182.361099
std              2.01294                        79.983052          134.095115
min              0.01000                         1.000000            0.000000
25%              0.33000                         1.000000           55.000000
50%              1.01000                         2.000000          174.000000
75%              2.49000                        12.000000          322.000000
max            190.48000                       660.000000          365.000000

       number_of_reviews_ltm
count          325858.000000
mean               11.889817
std                20.672830
min                 0.000000
25%                 0.000000
50%                 3.000000
75%                16.000000
max              1284.000000
```

[7]:
```
#describe the categorical columns in the dataframe
df.describe(include = 'O')
```

[7]:
```
                   name   host_name  neighbourhood_group  \
count            325839      324714               155047
unique           264019       33120                   34
top     Boutique Hostel  Blueground  City of Los Angeles
freq                152        4876                39002

               neighbourhood        room_type last_review license   state  \
count                 325146           325858      263166   86635  325858
unique                  1575                4        2965   48064      19
top     Unincorporated Areas  Entire home/apt  2022-09-05  Exempt      CA
freq                   13400           243098       13190    6603  127329

               city
count        325858
unique           31
top     Los Angeles
freq          91630
```

2 Cleaning

Now that we have a general understanding of the layout of the dataset, we can dive deeper into the data. This will consist of handling missing values, duplicates, drop columns, add columns, visualizations.

[8]:
```
#delete unneeded columns and missing values
df = df.drop(['Unnamed: 0','id','host_id','license', 'neighbourhood_group'],␣
↪axis = 1)
df.isna().sum()
```

[8]:
```
name                                  19
host_name                           1144
neighbourhood                        712
latitude                               0
longitude                              0
room_type                              0
price                                  0
minimum_nights                         0
number_of_reviews                      0
last_review                        62692
reviews_per_month                  62692
calculated_host_listings_count         0
availability_365                       0
number_of_reviews_ltm                  0
state                                  0
city                                   0
dtype: int64
```

[9]:
```
#drop rows where missing values in name, host_name, last review,␣
↪reviews_last_month, neighbourhood as these are probalby no longer active␣
↪airbnb

df.dropna(subset=['name','host_name','last_review','reviews_per_month',␣
↪'neighbourhood'], inplace = True)

df.isna().sum()
```

[9]:
```
name                              0
host_name                         0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
last_review                       0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
number_of_reviews_ltm             0
state                             0
```

```
city                                      0
dtype: int64
```

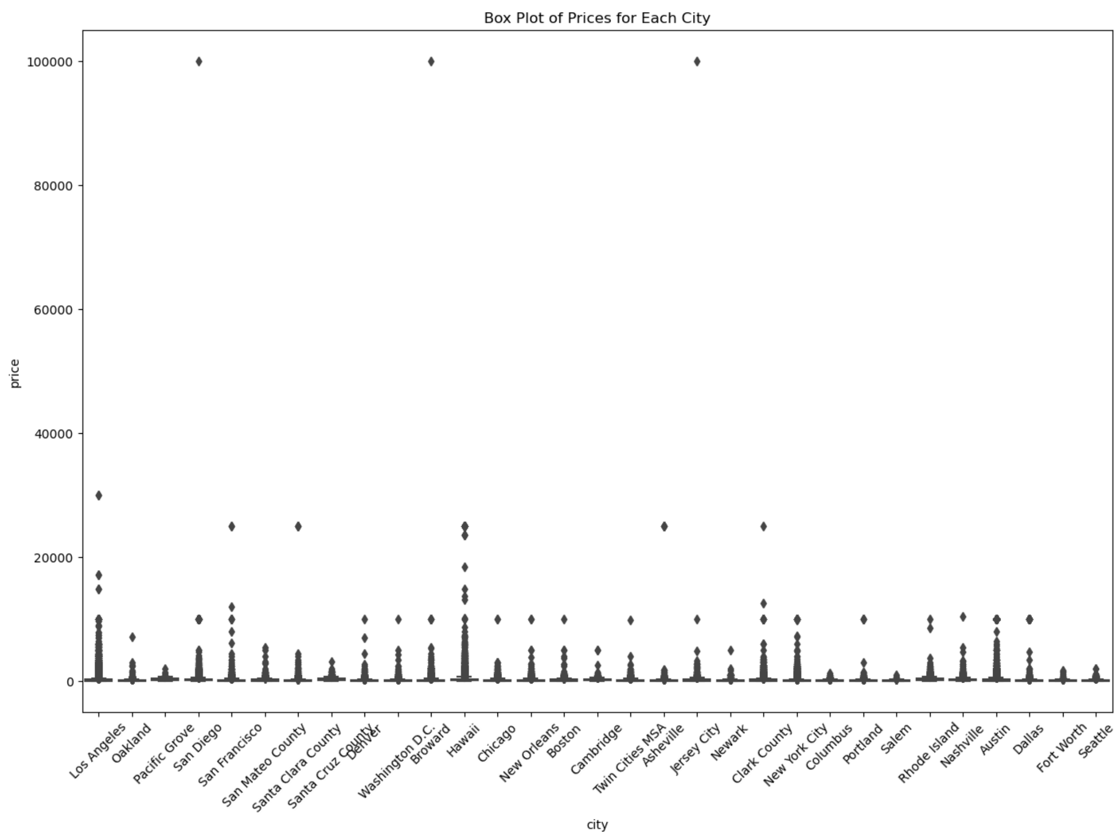
[10]:
```
df.shape
```

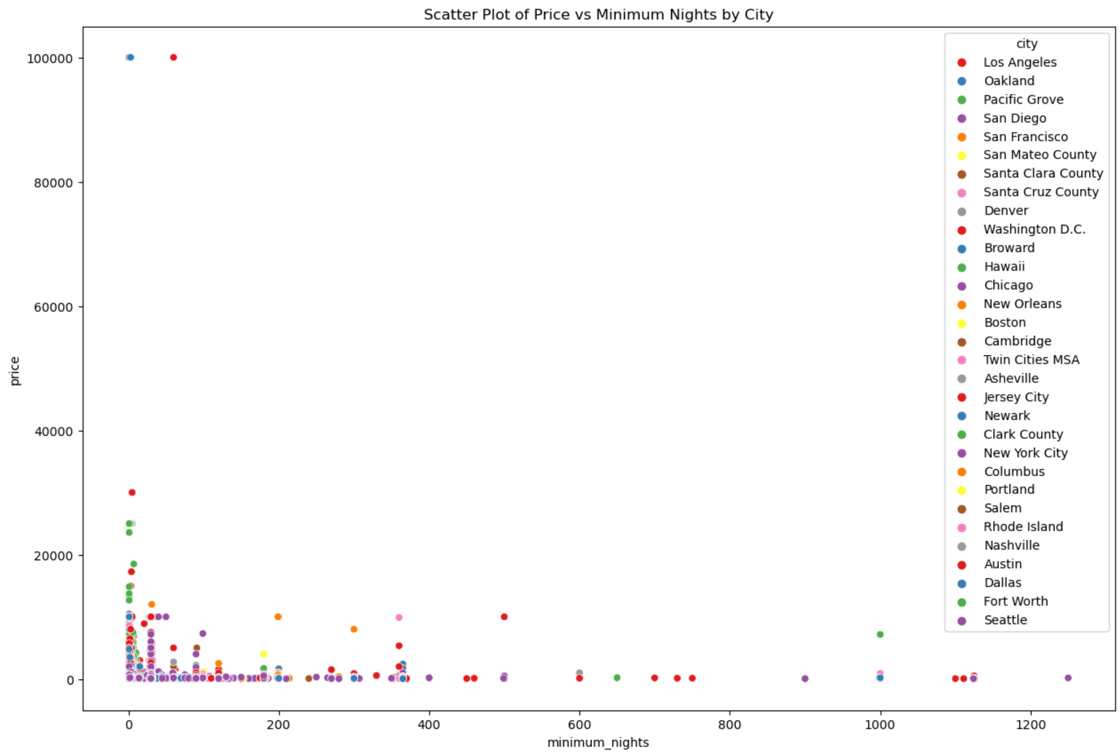
[10]:
```
(262040, 16)
```

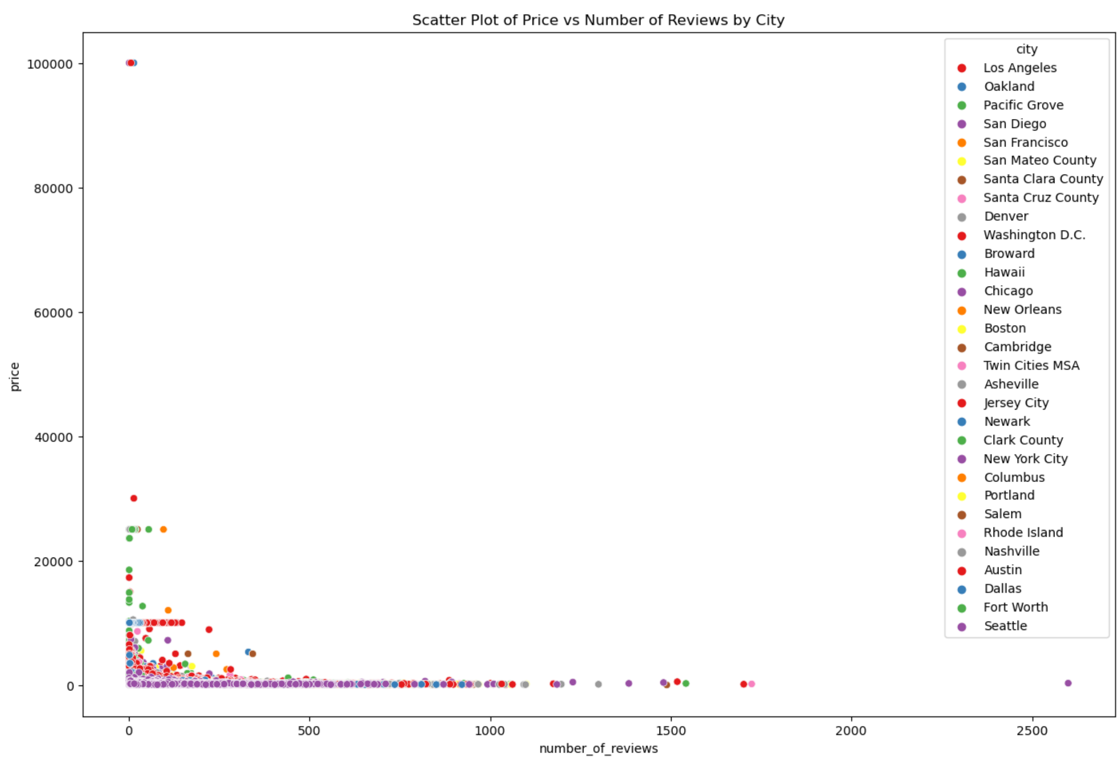
[11]:
```
#identify outliers in the data

import numpy as np



# Define numeric features for outlier detection
numeric_features = ['price', 'minimum_nights', 'number_of_reviews',
                    'reviews_per_month', 'calculated_host_listings_count',
                    'availability_365']

# Function to detect outliers using IQR within each city
def detect_outliers_iqr(df, features):
    outliers = pd.DataFrame()
    non_outliers = pd.DataFrame()
    for city in df['city'].unique():
        city_df = df[df['city'] == city]
        Q1 = city_df[features].quantile(0.25)
        Q3 = city_df[features].quantile(0.75)
        IQR = Q3 - Q1
        city_outliers = ((city_df[features] < (Q1 - 1.5 * IQR)) |␣
↪(city_df[features] > (Q3 + 1.5 * IQR))).any(axis=1)
        outliers = pd.concat([outliers, city_df[city_outliers]])
        non_outliers = pd.concat([non_outliers, city_df[~city_outliers]])
    return non_outliers, outliers

# Identify and remove outliers
df_no_outliers, df_outliers = detect_outliers_iqr(df, numeric_features)
print(f'Number of records before removing outliers: {len(df)}')
print(f'Number of outliers identified and removed: {df_outliers.shape[0]}')

print(f'Number of records after removing outliers: {df_no_outliers.shape[0]}')



import matplotlib.pyplot as plt
import seaborn as sns

# Plot box plots for each city
plt.figure(figsize=(15, 10))
```

```
sns.boxplot(x='city', y='price', data=df)
plt.xticks(rotation=45)
plt.title('Box Plot of Prices for Each City')
plt.show()

# Scatter plot to identify outliers in price vs. other features within each city
plt.figure(figsize=(15, 10))
sns.scatterplot(x='minimum_nights', y='price', hue='city', data=df,␣
↪palette='Set1')
plt.title('Scatter Plot of Price vs Minimum Nights by City')
plt.show()

plt.figure(figsize=(15, 10))
sns.scatterplot(x='number_of_reviews', y='price', hue='city', data=df,␣
↪palette='Set1')
plt.title('Scatter Plot of Price vs Number of Reviews by City')
plt.show()
```

```
Number of records before removing outliers: 262040
Number of outliers identified and removed: 96708
Number of records after removing outliers: 165332
```

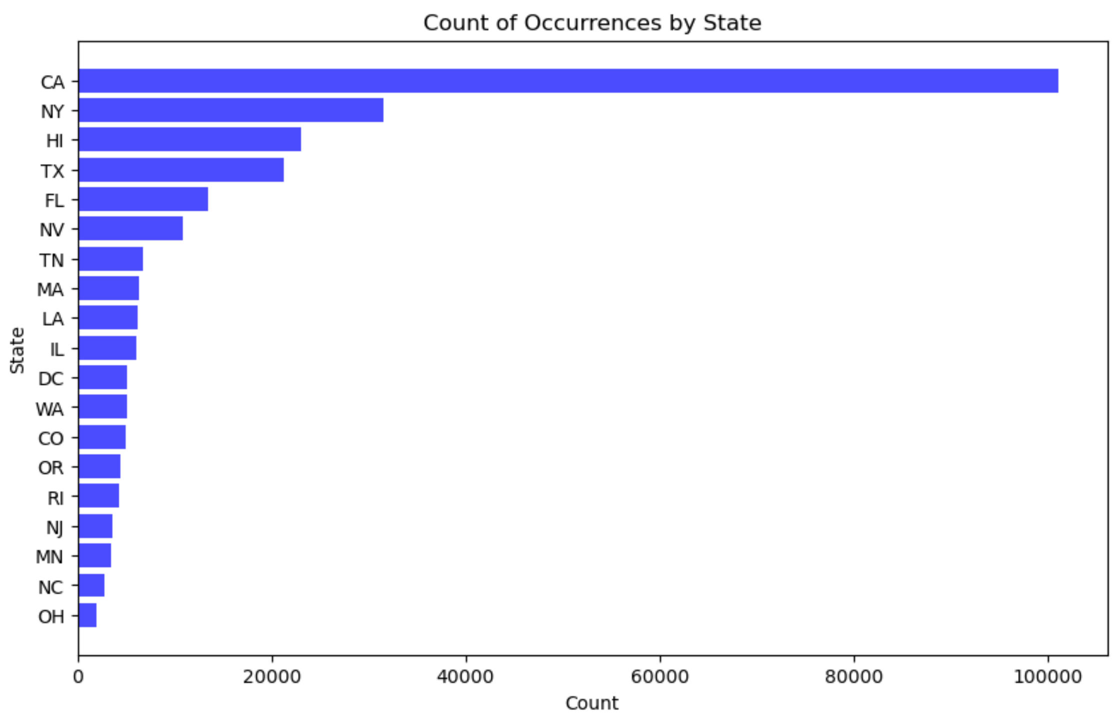



Scatter Plot of Price vs Minimum Nights by City
price
minimum_nights
0
20000
40000
60000
80000
100000
0
200
400
600
800
1000
1200
city
Los Angeles
Oakland
Pacific Grove
San Diego
San Francisco
San Mateo County
Santa Clara County
Santa Cruz County
Denver
Washington D.C.
Broward
Hawaii
Chicago
New Orleans
Boston
Cambridge
Twin Cities MSA
Asheville
Jersey City
Newark
Clark County
New York City
Columbus
Portland
Salem
Rhode Island
Nashville
Austin
Dallas
Fort Worth
Seattle


Scatter Plot of Price vs Number of Reviews by City
price
number_of_reviews
0
20000
40000
60000
80000
100000
0
500
1000
1500
2000
2500
city
Los Angeles
Oakland
Pacific Grove
San Diego
San Francisco
San Mateo County
Santa Clara County
Santa Cruz County
Denver
Washington D.C.
Broward
Hawaii
Chicago
New Orleans
Boston
Cambridge
Twin Cities MSA
Asheville
Jersey City
Newark
Clark County
New York City
Columbus
Portland
Salem
Rhode Island
Nashville
Austin
Dallas
Fort Worth
Seattle

2.1 Findings

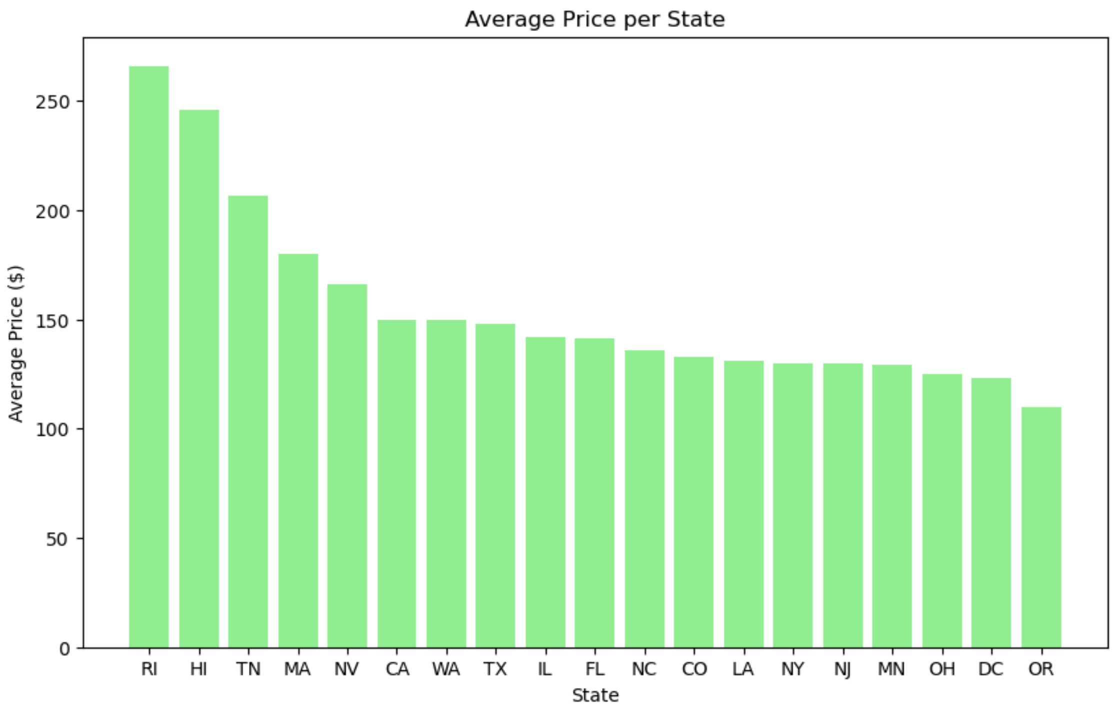
[12]:
```
#figure showing distribution by state
state_counts = df.groupby('state').size().reset_index(name='count').
 ↪sort_values('count')

plt.figure(figsize=(10, 6))
plt.barh(state_counts['state'], state_counts['count'], color='blue', alpha=0.7)
plt.xlabel('Count')
plt.ylabel('State')
plt.title('Count of Occurrences by State')
plt.show()
```



As you can see from the graphs above, majority of the listing on Airbnb in this dataset are located in California. The top 5 states with listings are California, New York, Hawaii, and Texas. California outnumbers all other states by almost 3 times.

[13]:
```
state = df.groupby('state')['price'].agg(['count','mean', 'median', 'max',␣
 ↪'min']).sort_values(by=['count','mean','median'], ascending =␣
 ↪[False,False,False])
state = state.sort_values('median', ascending = False)

# #plot average price per state
plt.figure(figsize = (10,6))
```

```python
plt.bar(state.index,state['median'],color = 'lightgreen')
plt.title('Average Price per State')
plt.xlabel('State')
plt.ylabel('Average Price ($)')
plt.show()
```



From the following you can see Rhode Island has the highest median price of listings. Following Rhode Island, the top 5 rounds out with Hawaii , Tennessee, Massachusents, and Nevada. As a potential investor considering Texas, you can see that the median price is on the lower side, and the state is top 5 in listings. I would consider this a good investment area because Texas is vast and large with many major areas, spreading out the competition. However we want to get a better understanding of what city is best to inveset into.

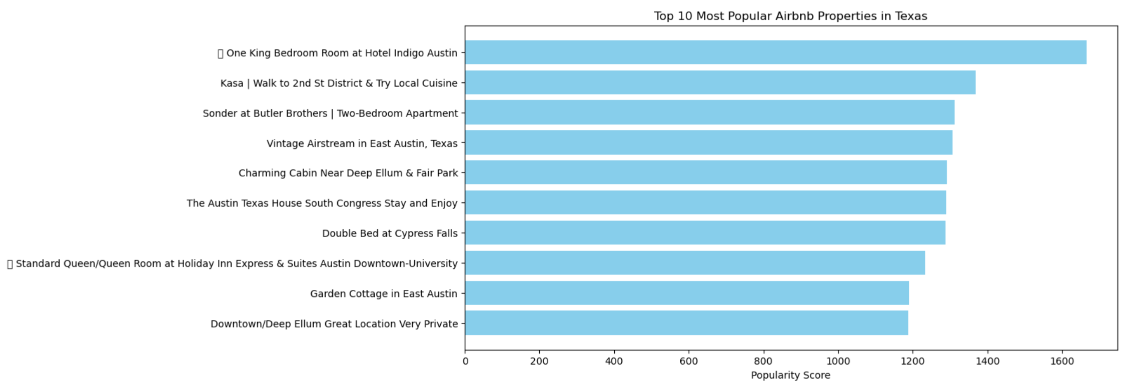
[14]:
```python
# Ensure we are working with a fresh copy
texas = df[df['state'] == 'TX'].copy()

# Create a popularity score using .loc
texas.loc[:, 'popularity_score'] = texas['number_of_reviews'] +␣
↪(texas['reviews_per_month'] * 30) - (texas['availability_365'] / 10)

# Sort properties by popularity score
most_popular_properties = texas.sort_values(by='popularity_score',␣
↪ascending=False).reset_index()
```

```python
# Select the top 10 most popular properties
top_10_most_popular = most_popular_properties.head(10)


print(top_10_most_popular[['name','state','city']])

# Plot the top 10 most popular properties without special characters
plt.figure(figsize=(12, 6))
plt.barh(top_10_most_popular['name'], top_10_most_popular['popularity_score'],
    color='skyblue')
plt.xlabel('Popularity Score')
plt.title('Top 10 Most Popular Airbnb Properties in Texas')
plt.gca().invert_yaxis()
plt.show()


# Ensure we are working with a fresh copy
california = df[df['state'] == 'CA'].copy()

# Create a popularity score using .loc
california.loc[:, 'popularity_score'] = california['number_of_reviews'] +
    (california['reviews_per_month'] * 30) - (california['availability_365'] /
    10)

# Sort properties by popularity score
most_popular_properties = california.sort_values(by='popularity_score',
    ascending=False).reset_index()

# Select the top 10 most popular properties
top_10_most_popular = most_popular_properties.head(15)


print(top_10_most_popular[['name','state','city']])

# Plot the top 10 most popular properties without special characters
plt.figure(figsize=(12, 6))
plt.barh(top_10_most_popular['name'], top_10_most_popular['popularity_score'],
    color='skyblue')
plt.xlabel('Popularity Score')
plt.title('Top 10 Most Popular Airbnb Properties in California')
plt.gca().invert_yaxis()
plt.show()
```

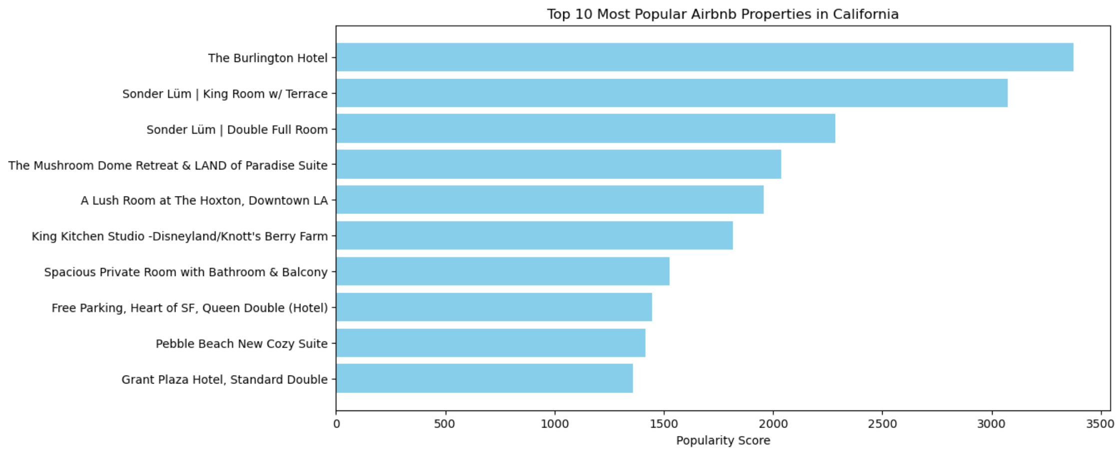
```
                                              name state    city
0       One King Bedroom Room at Hotel Indigo Austin    TX  Austin
1   Kasa | Walk to 2nd St District & Try Local Cui…    TX  Austin
2   Sonder at Butler Brothers | Two-Bedroom Apartment    TX  Dallas
```

```
3            Vintage Airstream in East Austin, Texas    TX  Austin
4         Charming Cabin Near Deep Ellum & Fair Park    TX  Dallas
5  The Austin Texas House South Congress Stay and…    TX  Austin
6                         Double Bed at Cypress Falls    TX  Austin
7    Standard Queen/Queen Room at Holiday Inn Exp…    TX  Austin
8                       Garden Cottage in East Austin    TX  Austin
9    Downtown/Deep Ellum Great Location Very Private    TX  Dallas
```

```
/Users/brianhonea/anaconda3/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: UserWarning: Glyph 10024 (\N{SPARKLES})
missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
                                              name state              city
0                             The Burlington Hotel    CA       Los Angeles
1                             The Burlington Hotel    CA       Los Angeles
2                Sonder Lüm | King Room w/ Terrace    CA       Los Angeles
3                Sonder Lüm | King Room w/ Terrace    CA       Los Angeles
4                    Sonder Lüm | Double Full Room    CA       Los Angeles
5                    Sonder Lüm | Double Full Room    CA       Los Angeles
6   The Mushroom Dome Retreat & LAND of Paradise S…    CA  Santa Cruz County
7           A Lush Room at The Hoxton, Downtown LA    CA       Los Angeles
8           A Lush Room at The Hoxton, Downtown LA    CA       Los Angeles
9   King Kitchen Studio -Disneyland/Knott's Berry …    CA       Los Angeles
10  King Kitchen Studio -Disneyland/Knott's Berry …    CA       Los Angeles
11        Spacious Private Room with Bathroom & Balcony    CA     San Francisco
12     Free Parking, Heart of SF, Queen Double (Hotel)    CA     San Francisco
13                       Pebble Beach New Cozy Suite    CA     Pacific Grove
14               Grant Plaza Hotel, Standard Double    CA     San Francisco
```



2.1.1 Findings on the Most Popular Airbnb Properties in Texas

Based on our analysis of the Airbnb dataset, we created a popularity score for properties in Texas by considering the number of reviews, reviews per month, and availability. This score was calculated as follows:

popularity_score = number_of_reviews + (reviews_per_month * 30) - (availability_365 / 10)

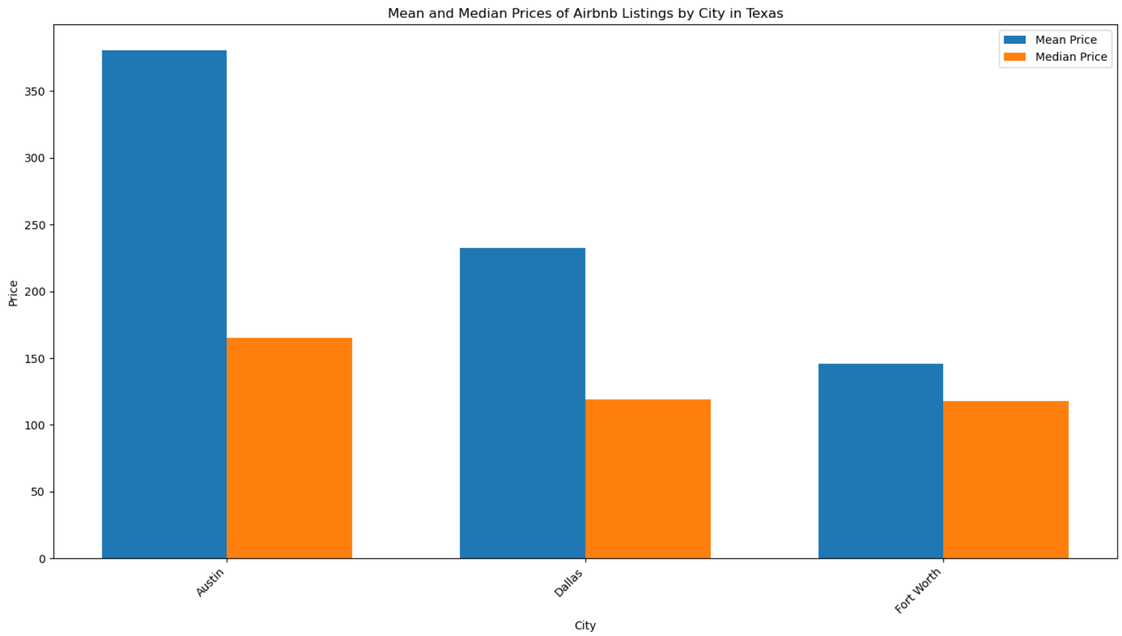
After sorting the properties by their popularity scores, we identified the top 10 most popular Airbnb properties in Texas. Here are the key findings:

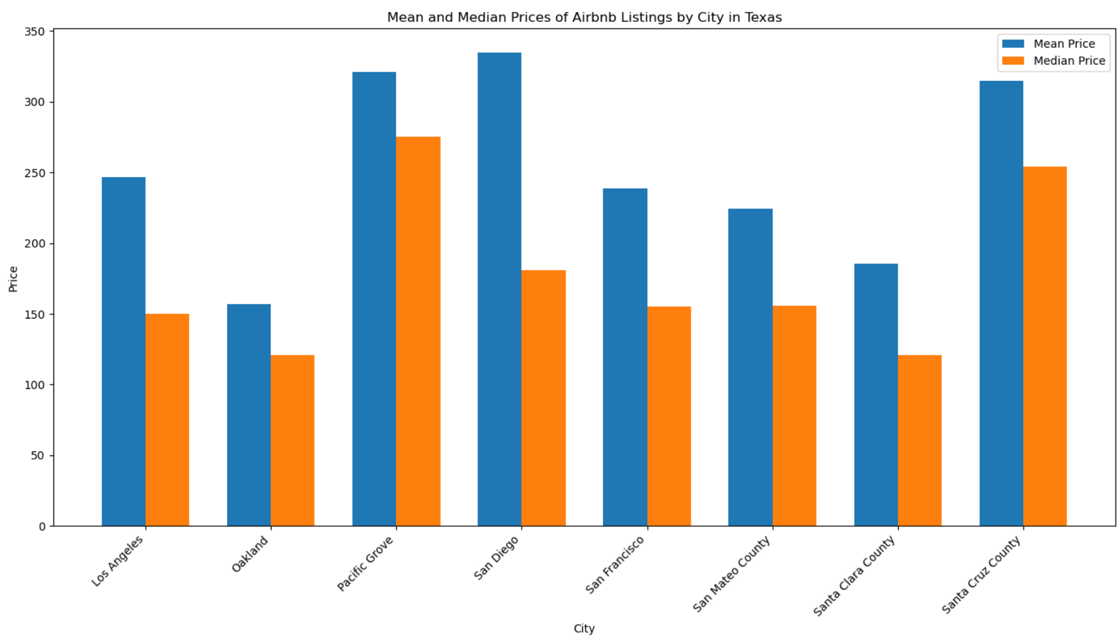
1. **Top Properties**: The most popular properties are located in various cities across Texas, highlighting the widespread interest in Airbnb accommodations throughout the state.
2. **High Review Count**: These properties have a high number of reviews, indicating that they are frequently booked and reviewed by guests.
3. **Active Engagement**: Properties with higher reviews per month are consistently attracting guests, suggesting active engagement and positive experiences that encourage reviews.
4. **Low Availability**: Many of the top properties have lower availability throughout the year, implying they are frequently booked and in high demand.

This analysis provides valuable insights into the factors contributing to the popularity of Airbnb properties. Hosts and property managers can leverage this information to enhance their listings, improve guest experiences, and increase bookings by focusing on generating positive reviews and maintaining high occupancy rates.

[15]:
```
# Group by city and calculate mean and median prices
texas_cities = texas.groupby('city')['price'].agg(['mean', 'median']).
 ↪reset_index()

# Define the positions for the bars
x = np.arange(len(texas_cities['city']))
width = 0.35
```

```python
# Create the plot
plt.figure(figsize=(14, 8))
plt.bar(x - width/2, texas_cities['mean'], width, label='Mean Price')
plt.bar(x + width/2, texas_cities['median'], width, label='Median Price')

# Add labels, title, and legend
plt.xlabel('City')
plt.ylabel('Price')
plt.title('Mean and Median Prices of Airbnb Listings by City in Texas')
plt.xticks(ticks=x, labels=texas_cities['city'], rotation=45, ha='right')
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()




# Group by city and calculate mean and median prices
california_cities = california.groupby('city')['price'].agg(['mean', 'median']).
 ↪reset_index()

# Define the positions for the bars
x = np.arange(len(california_cities['city']))
width = 0.35

# Create the plot
plt.figure(figsize=(14, 8))
plt.bar(x - width/2, california_cities['mean'], width, label='Mean Price')
plt.bar(x + width/2, california_cities['median'], width, label='Median Price')

# Add labels, title, and legend
plt.xlabel('City')
plt.ylabel('Price')
plt.title('Mean and Median Prices of Airbnb Listings by City in Texas')
plt.xticks(ticks=x, labels=california_cities['city'], rotation=45, ha='right')
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```





2.1.2 Visualizations

[16]:
```
import folium
from folium.plugins import MarkerCluster
```

```python
# Filter the DataFrame for listings in Texas
texas = df[df['state'] == 'TX']

# Create a base map centered around DFW (Dallas-Fort Worth)
m = folium.Map(location=[32.7079, -96.9209], zoom_start=9)

# Create a marker cluster
marker_cluster = MarkerCluster().add_to(m)

# Add markers to the map
for idx, row in texas.iterrows():
    folium.Marker(
        location=[row['latitude'], row['longitude']],
        popup=folium.Popup(
            f"<strong>Price:</strong> ${row['price']}<br>"
            f"<strong>Name:</strong> {row['name']}<br>"
            f"<strong>Min Nights:</strong> {row['minimum_nights']}",
            max_width=300
        ),
        icon=folium.Icon(color='blue', icon='info-sign')
    ).add_to(marker_cluster)

# Add zip code boundaries
# Replace 'path_to_zip_codes.geojson' with the path to your GeoJSON file
folium.GeoJson(
    "texas-zip-codes-_1613.geojson",
    name='Zip Codes',
    style_function=lambda x: {'fillColor': 'transparent', 'color': 'black',␣
 ↪'weight': 1}
).add_to(m)

# Add layer control to toggle GeoJson overlay
folium.LayerControl().add_to(m)

m
```

[16]: <folium.folium.Map at 0x142d5de10>

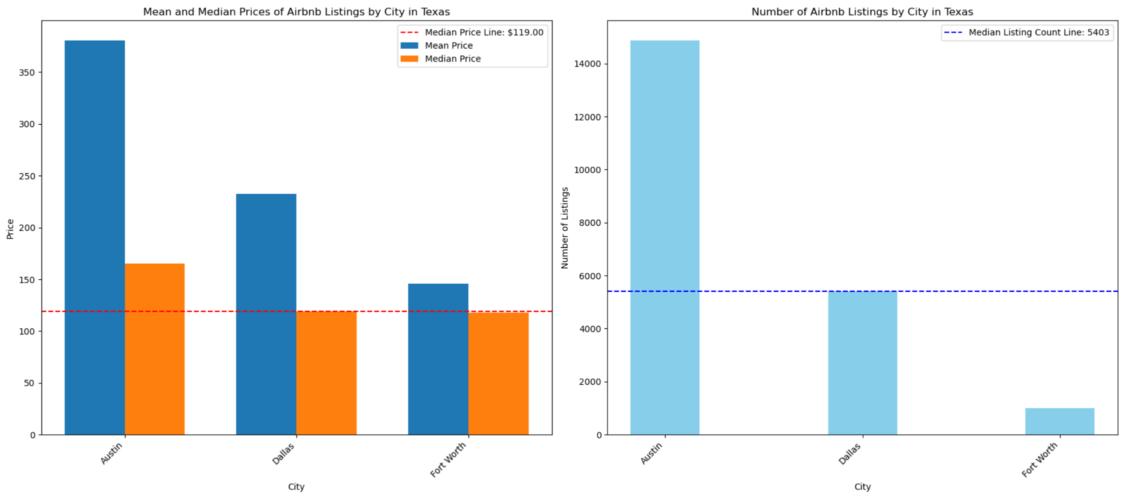
The interactive map above allows you to look across the map at all Airbnb listings, when you click on a listing it will display the name, price per night, and minimum nights required to book. This gives us a great idea of the competition in certain areas. For someone wanting to invest in an Airbnb property it would be great to look at competetion in the area and see also what commonalities the competetion has in terms of price, minimum nights.

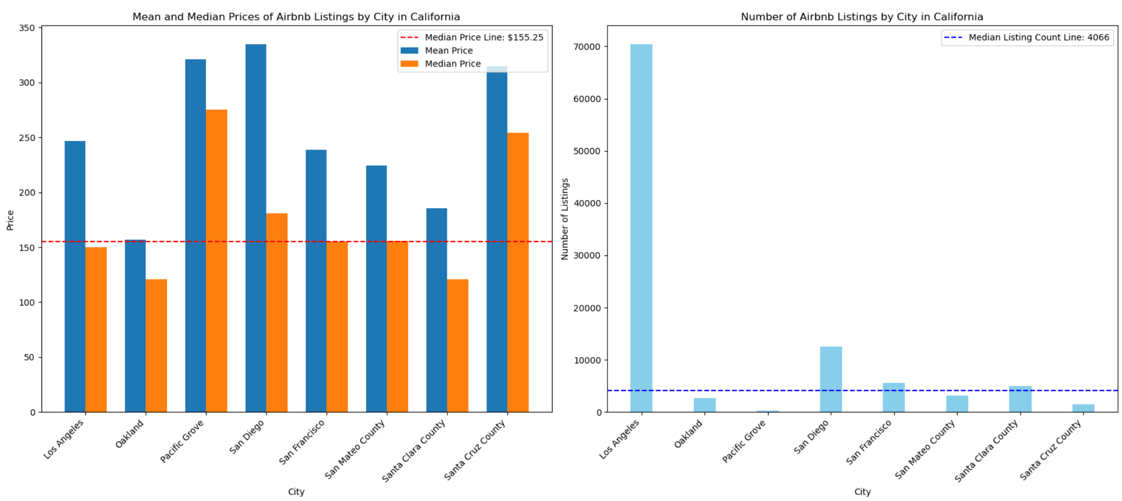
[17]:
```python
# Ensure we are working with a fresh copy of the Texas data
texas = df[df['state'] == 'TX'].copy()

# Group by city and calculate mean, median, and count of prices
```

```
texas_cities = texas.groupby('city')['price'].agg(['mean', 'median', 'count']).
    ↪reset_index()

# Define the positions for the bars
x = np.arange(len(texas_cities['city']))
width = 0.35

# Calculate median listing count for Texas
median_listing_count_tx = texas_cities['count'].median()

# Create the plots
plt.figure(figsize=(18, 8))

# Plot mean and median prices for Texas
plt.subplot(1, 2, 1)
bars1 = plt.bar(x - width/2, texas_cities['mean'], width, label='Mean Price')
bars2 = plt.bar(x + width/2, texas_cities['median'], width, label='Median␣
    ↪Price')

# Highlight the median price with a horizontal line
median_price_value_tx = texas_cities['median'].median()
plt.axhline(median_price_value_tx, color='red', linestyle='--', label=f'Median␣
    ↪Price Line: ${median_price_value_tx:.2f}')

plt.xlabel('City')
plt.ylabel('Price')
plt.title('Mean and Median Prices of Airbnb Listings by City in Texas')
plt.xticks(ticks=x, labels=texas_cities['city'], rotation=45, ha='right')
plt.legend()

# Plot count of listings for Texas
plt.subplot(1, 2, 2)
plt.bar(x, texas_cities['count'], width, color='skyblue')
plt.xlabel('City')
plt.ylabel('Number of Listings')
plt.title('Number of Airbnb Listings by City in Texas')
plt.xticks(ticks=x, labels=texas_cities['city'], rotation=45, ha='right')

# Add a horizontal line for the median number of listings
plt.axhline(median_listing_count_tx, color='blue', linestyle='--',␣
    ↪label=f'Median Listing Count Line: {median_listing_count_tx:.0f}')
plt.legend()

plt.tight_layout()
plt.show()

# Ensure we are working with a fresh copy of the California data
```

```python
california = df[df['state'] == 'CA'].copy()

# Group by city and calculate mean, median, and count of prices
california_cities = california.groupby('city')['price'].agg(['mean', 'median', 'count']).reset_index()

# Define the positions for the bars
x = np.arange(len(california_cities['city']))
width = 0.35

# Calculate median listing count for California
median_listing_count_ca = california_cities['count'].median()

# Create the plots
plt.figure(figsize=(18, 8))

# Plot mean and median prices for California
plt.subplot(1, 2, 1)
bars1 = plt.bar(x - width/2, california_cities['mean'], width, label='Mean Price')
bars2 = plt.bar(x + width/2, california_cities['median'], width, label='Median Price')

# Highlight the median price with a horizontal line
median_price_value_ca = california_cities['median'].median()
plt.axhline(median_price_value_ca, color='red', linestyle='--', label=f'Median Price Line: ${median_price_value_ca:.2f}')

plt.xlabel('City')
plt.ylabel('Price')
plt.title('Mean and Median Prices of Airbnb Listings by City in California')
plt.xticks(ticks=x, labels=california_cities['city'], rotation=45, ha='right')
plt.legend()

# Plot count of listings for California
plt.subplot(1, 2, 2)
plt.bar(x, california_cities['count'], width, color='skyblue')
plt.xlabel('City')
plt.ylabel('Number of Listings')
plt.title('Number of Airbnb Listings by City in California')
plt.xticks(ticks=x, labels=california_cities['city'], rotation=45, ha='right')

# Add a horizontal line for the median number of listings
plt.axhline(median_listing_count_ca, color='blue', linestyle='--', label=f'Median Listing Count Line: {median_listing_count_ca:.0f}')
plt.legend()
```

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





2.1.3 Findings on the Most Popular Airbnb Properties in Texas

Based on our analysis of the Airbnb dataset, we created a popularity score for properties in Texas by considering the number of reviews, reviews per month, and availability. This score was calculated as follows:

popularity_score = number_of_reviews + (reviews_per_month * 30) - (availability_365 / 10)

After sorting the properties by their popularity scores, we identified the top 10 most popular Airbnb properties in Texas. Here are the key findings:

1. **Top Properties**: The most popular properties are located in various cities across Texas,

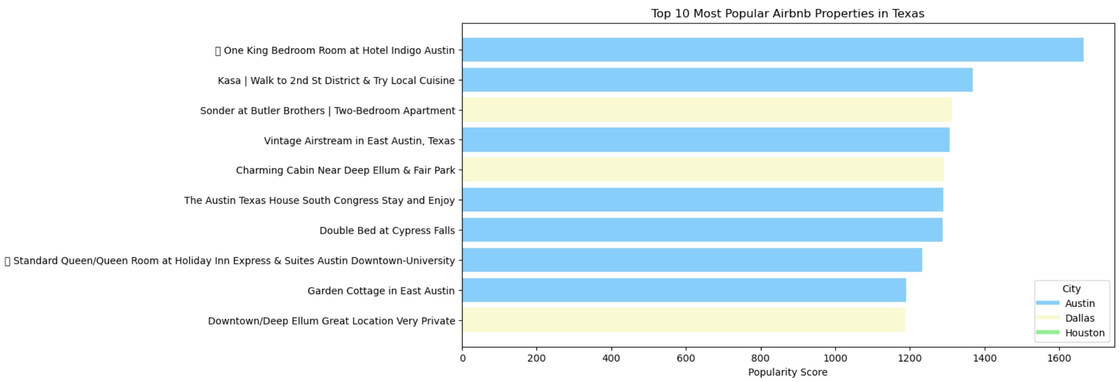
highlighting the widespread interest in Airbnb accommodations throughout the state.
2. **High Review Count**: These properties have a high number of reviews, indicating that they are frequently booked and reviewed by guests.
3. **Active Engagement**: Properties with higher reviews per month are consistently attracting guests, suggesting active engagement and positive experiences that encourage reviews.
4. **Low Availability**: Many of the top properties have lower availability throughout the year, implying they are frequently booked and in high demand.

This analysis provides valuable insights into the factors contributing to the popularity of Airbnb properties. Hosts and property managers can leverage this information to enhance their listings, improve guest experiences, and increase bookings by focusing on generating positive reviews and maintaining high occupancy rates.

[18]:
```
# Ensure we are working with a fresh copy
texas = df[df['state'] == 'TX'].copy()

# Create a popularity score using .loc
texas.loc[:, 'popularity_score'] = texas['number_of_reviews'] +
 ↪(texas['reviews_per_month'] * 30) - (texas['availability_365'] / 10)

# Sort properties by popularity score
most_popular_properties_texas = texas.sort_values(by='popularity_score',
 ↪ascending=False).reset_index()

# Select the top 10 most popular properties
top_10_most_popular_texas = most_popular_properties_texas.head(10)

# Define colors for different cities in Texas
city_colors_texas = {
    'Austin': 'lightskyblue',
    'Dallas': 'lightgoldenrodyellow',
    'Houston': 'lightgreen',
    # Add more cities if needed
}

# Plot the top 10 most popular properties in Texas
plt.figure(figsize=(12, 6))
colors_texas = top_10_most_popular_texas['city'].map(city_colors_texas).
 ↪fillna('grey')  # Handle missing colors
bars_texas = plt.barh(top_10_most_popular_texas['name'],
 ↪top_10_most_popular_texas['popularity_score'], color=colors_texas)

# Create a legend
handles_texas = [plt.Line2D([0], [0], color=color, lw=4) for color in
 ↪city_colors_texas.values()]
labels_texas = city_colors_texas.keys()
plt.legend(handles_texas, labels_texas, title='City', loc='best')
```

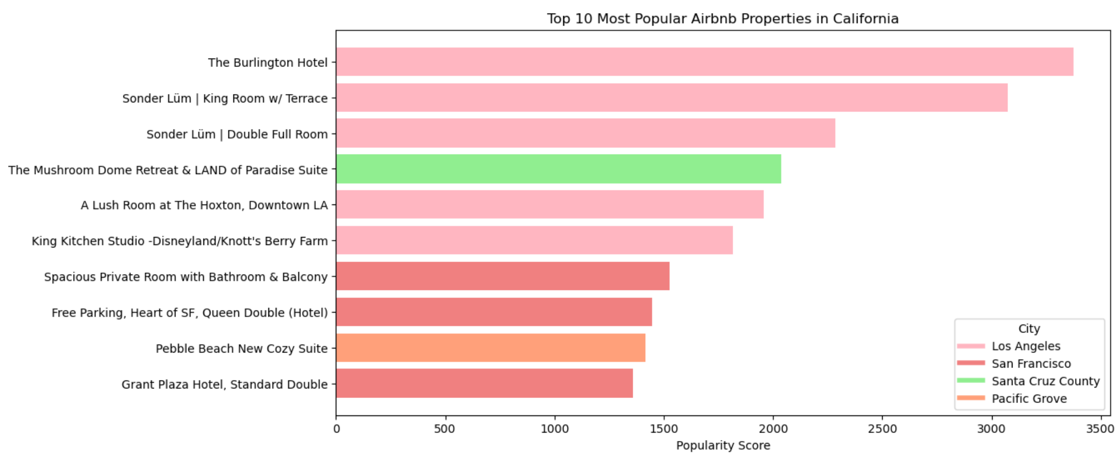
```
plt.xlabel('Popularity Score')
plt.title('Top 10 Most Popular Airbnb Properties in Texas')
plt.gca().invert_yaxis()
plt.show()
```

```
/Users/brianhonea/anaconda3/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: UserWarning: Glyph 10024 (\N{SPARKLES})
missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



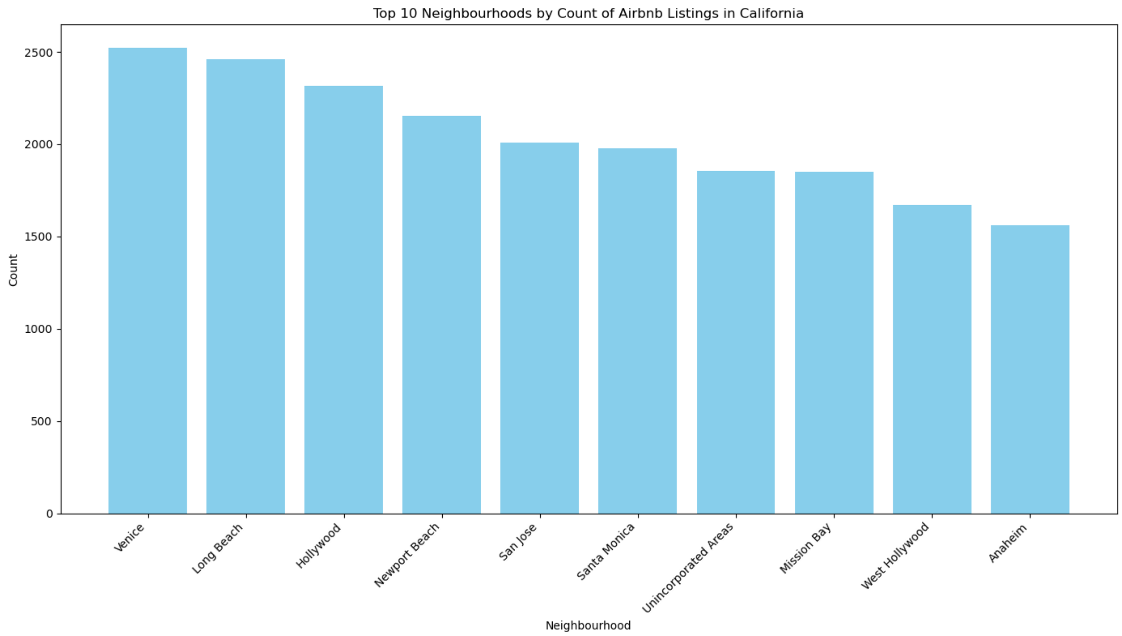
[19]:
```
# Ensure we are working with a fresh copy
california = df[df['state'] == 'CA'].copy()

# Create a popularity score using .loc
california.loc[:, 'popularity_score'] = california['number_of_reviews'] + (california['reviews_per_month'] * 30) - (california['availability_365'] / 10)

# Sort properties by popularity score
most_popular_properties_california = california.sort_values(by='popularity_score', ascending=False).reset_index()

# Select the top 10 most popular properties
top_10_most_popular_california = most_popular_properties_california.head(15)

# Define colors for different cities in California
city_colors_california = {
    'Los Angeles': 'lightpink',
    'San Francisco': 'lightcoral',
    'Santa Cruz County': 'lightgreen',
    'Pacific Grove': 'lightsalmon',
    # Add more cities if needed
```

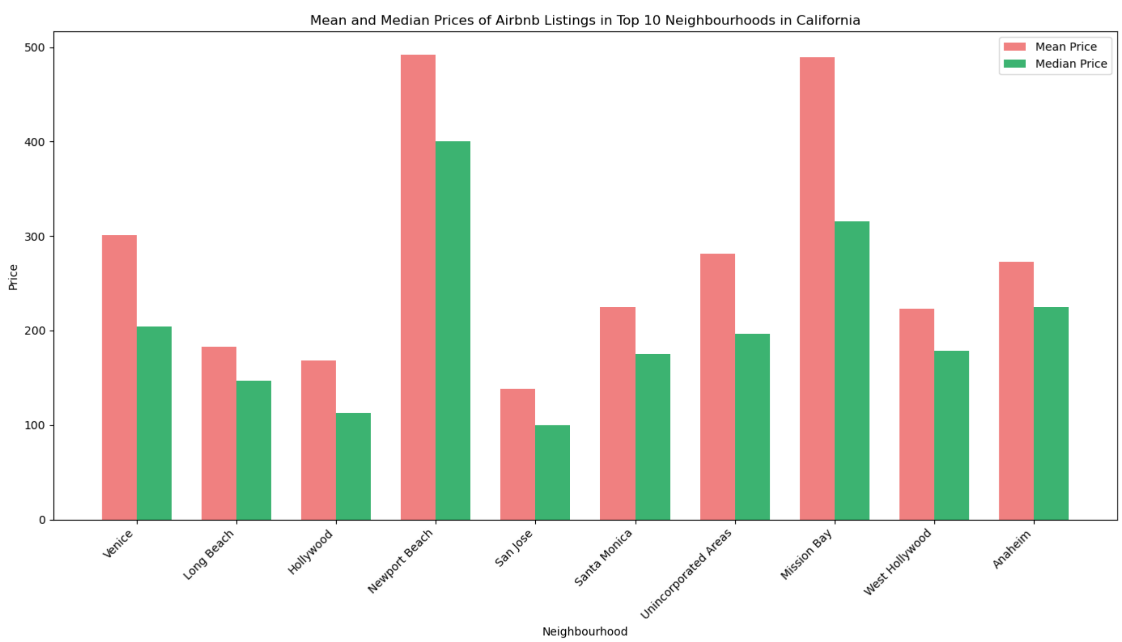
```
}

# Plot the top 10 most popular properties in California
plt.figure(figsize=(12, 6))
colors_california = top_10_most_popular_california['city'].
    ↪map(city_colors_california).fillna('grey')  # Handle missing colors
bars_california = plt.barh(top_10_most_popular_california['name'],␣
    ↪top_10_most_popular_california['popularity_score'], color=colors_california)

# Create a legend
handles_california = [plt.Line2D([0], [0], color=color, lw=4) for color in␣
    ↪city_colors_california.values()]
labels_california = city_colors_california.keys()
plt.legend(handles_california, labels_california, title='City', loc='best')

plt.xlabel('Popularity Score')
plt.title('Top 10 Most Popular Airbnb Properties in California')
plt.gca().invert_yaxis()
plt.show()
```



[20]:
```
# Calculate the count of listings per neighborhood
neighbourhood_counts = california['neighbourhood'].value_counts().reset_index()
neighbourhood_counts.columns = ['neighbourhood', 'count']

# Select the top 10 neighborhoods by count
top_10_neighbourhoods = neighbourhood_counts.head(10)

# Filter the original DataFrame to include only the top 10 neighborhoods
top_10_data = california[california['neighbourhood'].
    ↪isin(top_10_neighbourhoods['neighbourhood'])]
```

```
# Group by neighborhood and calculate mean and median prices
top_10_prices = top_10_data.groupby('neighbourhood')['price'].agg(['mean', 'median']).reset_index()

# Sort the mean and median prices DataFrame to match the order of the top 10 neighborhoods by count
top_10_prices = top_10_prices.set_index('neighbourhood').loc[top_10_neighbourhoods['neighbourhood']].reset_index()

# Create the first plot for the top 10 neighborhoods by count
plt.figure(figsize=(14, 8))
plt.bar(top_10_neighbourhoods['neighbourhood'], top_10_neighbourhoods['count'], color='skyblue')
plt.xlabel('Neighbourhood')
plt.ylabel('Count')
plt.title('Top 10 Neighbourhoods by Count of Airbnb Listings in California')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Create the second plot for the mean and median prices of the top 10 neighborhoods
x = np.arange(len(top_10_prices['neighbourhood']))
width = 0.35

plt.figure(figsize=(14, 8))
plt.bar(x - width/2, top_10_prices['mean'], width, label='Mean Price', color='lightcoral')
plt.bar(x + width/2, top_10_prices['median'], width, label='Median Price', color='mediumseagreen')
plt.xlabel('Neighbourhood')
plt.ylabel('Price')
plt.title('Mean and Median Prices of Airbnb Listings in Top 10 Neighbourhoods in California')
plt.xticks(ticks=x, labels=top_10_prices['neighbourhood'], rotation=45, ha='right')
plt.legend()
plt.tight_layout()
plt.show()
```

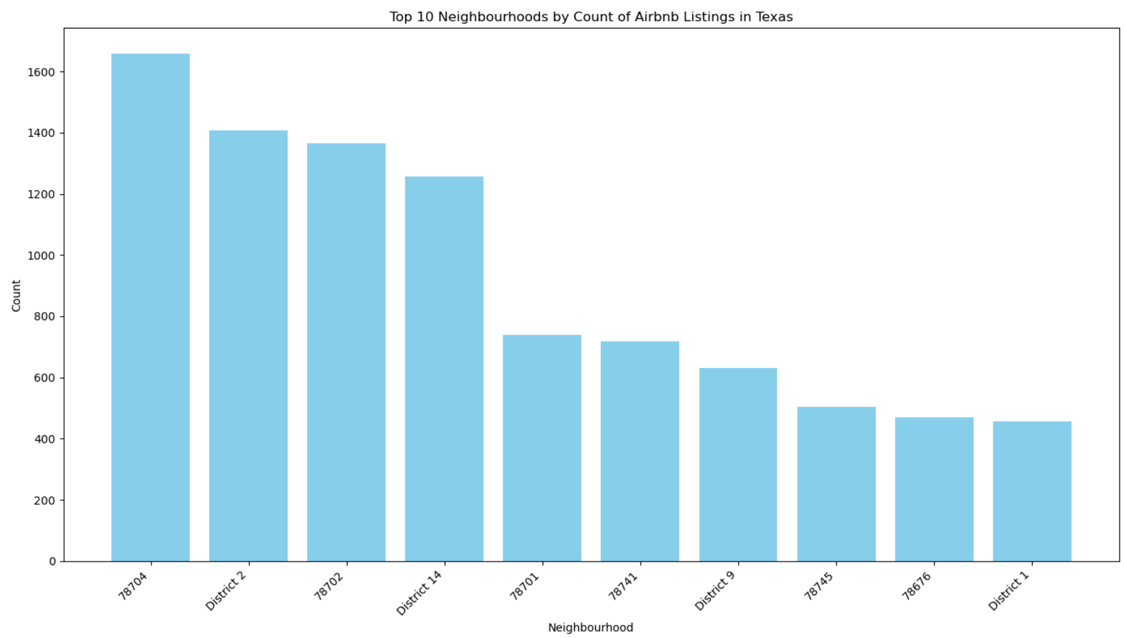


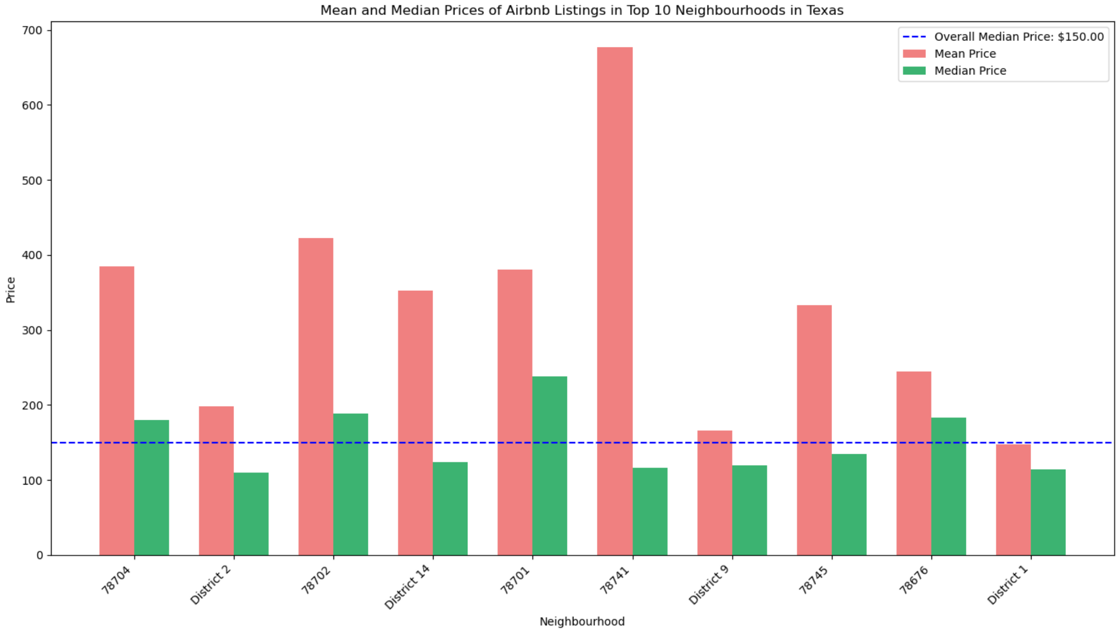


[21]:
```
# Calculate the count of listings per neighborhood
neighbourhood_counts = texas['neighbourhood'].value_counts().reset_index()
neighbourhood_counts.columns = ['neighbourhood', 'count']

# Select the top 10 neighborhoods by count
```

```python
top_10_neighbourhoods = neighbourhood_counts.head(10)

# Filter the original DataFrame to include only the top 10 neighborhoods
top_10_data = texas[texas['neighbourhood'].isin(top_10_neighbourhoods['neighbourhood'])]

# Group by neighborhood and calculate mean and median prices
top_10_prices = top_10_data.groupby('neighbourhood')['price'].agg(['mean', 'median']).reset_index()

# Sort the mean and median prices DataFrame to match the order of the top 10 neighborhoods by count
top_10_prices = top_10_prices.set_index('neighbourhood').loc[top_10_neighbourhoods['neighbourhood']].reset_index()

# Calculate the overall median price for the top 10 neighborhoods
overall_median_price = top_10_data['price'].median()

# Create the first plot for the top 10 neighborhoods by count
plt.figure(figsize=(14, 8))
plt.bar(top_10_neighbourhoods['neighbourhood'], top_10_neighbourhoods['count'], color='skyblue')
plt.xlabel('Neighbourhood')
plt.ylabel('Count')
plt.title('Top 10 Neighbourhoods by Count of Airbnb Listings in Texas')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Create the second plot for the mean and median prices of the top 10 neighborhoods
x = np.arange(len(top_10_prices['neighbourhood']))
width = 0.35

plt.figure(figsize=(14, 8))
plt.bar(x - width/2, top_10_prices['mean'], width, label='Mean Price', color='lightcoral')
plt.bar(x + width/2, top_10_prices['median'], width, label='Median Price', color='mediumseagreen')

# Add a horizontal line for the overall median price
plt.axhline(y=overall_median_price, color='blue', linestyle='--', linewidth=1.5, label=f'Overall Median Price: ${overall_median_price:.2f}')

# Add labels, title, and legend
plt.xlabel('Neighbourhood')
```

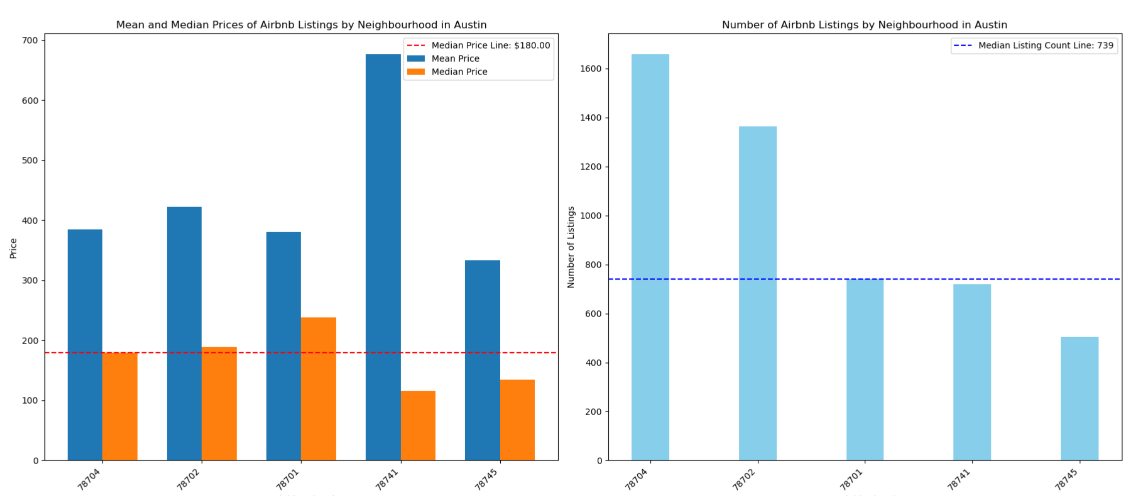
```
plt.ylabel('Price')
plt.title('Mean and Median Prices of Airbnb Listings in Top 10 Neighbourhoods␣
↪in Texas')
plt.xticks(ticks=x, labels=top_10_prices['neighbourhood'], rotation=45,␣
↪ha='right')
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```





[22]:
```
# Filter for Austin
austin_data = texas[texas['city'] == 'Austin']

# Group by neighborhood and calculate mean, median, and count of prices
austin_neighborhoods = austin_data.groupby('neighbourhood')['price'].
 ↪agg(['mean', 'median', 'count']).reset_index()

# Get the top 5 neighborhoods by number of listings
top_5_neighborhoods = austin_neighborhoods.nlargest(5, 'count')

# Define the positions for the bars
x = np.arange(len(top_5_neighborhoods['neighbourhood']))
width = 0.35

# Create the plots
plt.figure(figsize=(18, 8))

# Plot mean and median prices
plt.subplot(1, 2, 1)
bars1 = plt.bar(x - width/2, top_5_neighborhoods['mean'], width, label='Mean␣
 ↪Price')
bars2 = plt.bar(x + width/2, top_5_neighborhoods['median'], width,␣
 ↪label='Median Price')

# Highlight the median price with a horizontal line
median_price_value = top_5_neighborhoods['median'].median()
```

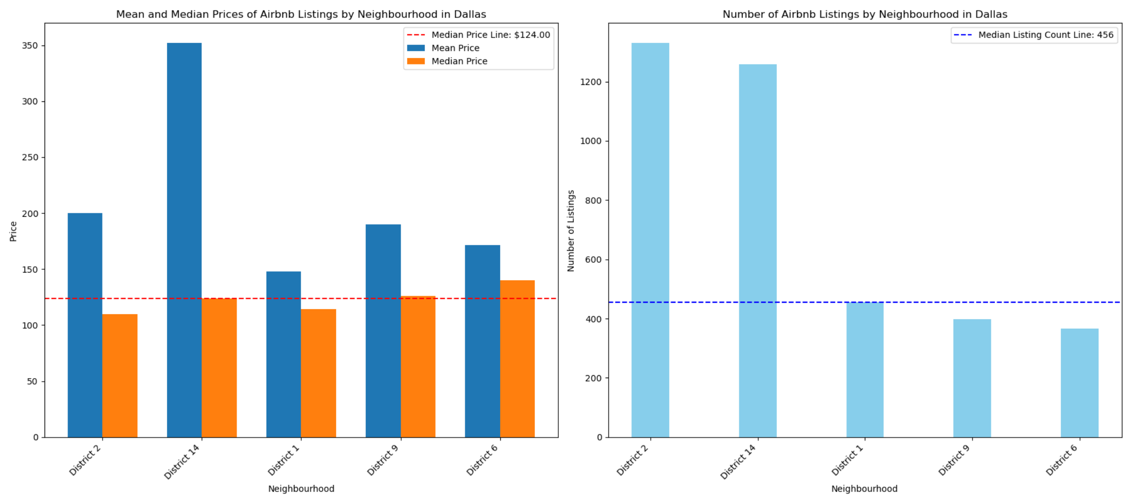
```python
plt.axhline(median_price_value, color='red', linestyle='--', label=f'Median Price Line: ${median_price_value:.2f}')

plt.xlabel('Neighbourhood')
plt.ylabel('Price')
plt.title('Mean and Median Prices of Airbnb Listings by Neighbourhood in Austin')
plt.xticks(ticks=x, labels=top_5_neighborhoods['neighbourhood'], rotation=45, ha='right')
plt.legend()

# Plot count of listings
plt.subplot(1, 2, 2)
plt.bar(x, top_5_neighborhoods['count'], width, color='skyblue')
plt.xlabel('Neighborhood')
plt.ylabel('Number of Listings')
plt.title('Number of Airbnb Listings by Neighbourhood in Austin')
plt.xticks(ticks=x, labels=top_5_neighborhoods['neighbourhood'], rotation=45, ha='right')

# Highlight the median listing count with a horizontal line
median_listing_count_austin = top_5_neighborhoods['count'].median()

# Add a horizontal line for the median number of listings
plt.axhline(median_listing_count_austin, color='blue', linestyle='--', label=f'Median Listing Count Line: {median_listing_count_austin:.0f}')
plt.legend()

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



[23]:

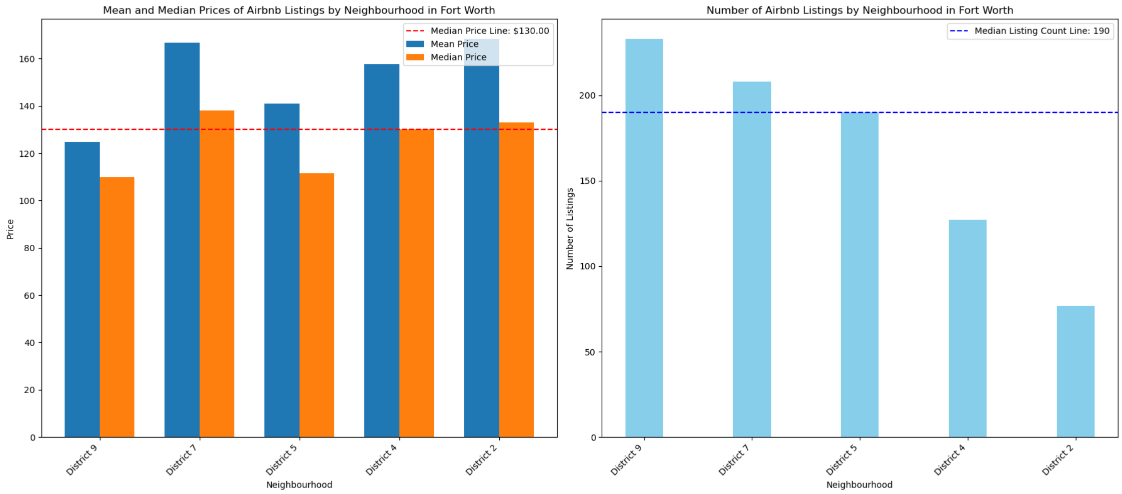
```
# Filter for Dallas
dallas_data = texas[texas['city'] == 'Dallas']


# Group by neighbourhood and calculate mean, median, and count of prices
dallas_neighbourhoods = dallas_data.groupby('neighbourhood')['price'].
↪agg(['mean', 'median', 'count']).reset_index()

# Get the top 5 neighbourhoods by number of listings
top_5_neighbourhoods_dallas = dallas_neighbourhoods.nlargest(5, 'count')

# Define the positions for the bars
x = np.arange(len(top_5_neighbourhoods_dallas['neighbourhood']))
width = 0.35

# Create the plots
plt.figure(figsize=(18, 8))

# Plot mean and median prices
plt.subplot(1, 2, 1)
bars1 = plt.bar(x - width/2, top_5_neighbourhoods_dallas['mean'], width,␣
↪label='Mean Price')
bars2 = plt.bar(x + width/2, top_5_neighbourhoods_dallas['median'], width,␣
↪label='Median Price')

# Highlight the median price with a horizontal line
median_price_value_dallas = top_5_neighbourhoods_dallas['median'].median()
plt.axhline(median_price_value_dallas, color='red', linestyle='--',␣
↪label=f'Median Price Line: ${median_price_value_dallas:.2f}')

plt.xlabel('Neighbourhood')
plt.ylabel('Price')
plt.title('Mean and Median Prices of Airbnb Listings by Neighbourhood in␣
↪Dallas')
plt.xticks(ticks=x, labels=top_5_neighbourhoods_dallas['neighbourhood'],␣
↪rotation=45, ha='right')
plt.legend()

# Plot count of listings
plt.subplot(1, 2, 2)
plt.bar(x, top_5_neighbourhoods_dallas['count'], width, color='skyblue')
plt.xlabel('Neighbourhood')
plt.ylabel('Number of Listings')
plt.title('Number of Airbnb Listings by Neighbourhood in Dallas')
```

```
plt.xticks(ticks=x, labels=top_5_neighbourhoods_dallas['neighbourhood'],␣
 ↪rotation=45, ha='right')

# Highlight the median listing count with a horizontal line
median_listing_count_dallas = top_5_neighbourhoods_dallas['count'].median()

# Add a horizontal line for the median number of listings
plt.axhline(median_listing_count_dallas, color='blue', linestyle='--',␣
 ↪label=f'Median Listing Count Line: {median_listing_count_dallas:.0f}')
plt.legend()

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



[24]:
```
# Filter for Fort Worth
fort_worth_data = texas[texas['city'] == 'Fort Worth']

# Group by neighbourhood and calculate mean, median, and count of prices
fort_worth_neighbourhoods = fort_worth_data.groupby('neighbourhood')['price'].
 ↪agg(['mean', 'median', 'count']).reset_index()

# Get the top 5 neighbourhoods by number of listings
top_5_neighbourhoods_fort_worth = fort_worth_neighbourhoods.nlargest(5, 'count')

# Define the positions for the bars
x = np.arange(len(top_5_neighbourhoods_fort_worth['neighbourhood']))
width = 0.35

# Create the plots
plt.figure(figsize=(18, 8))
```

```python
# Plot mean and median prices
plt.subplot(1, 2, 1)
bars1 = plt.bar(x - width/2, top_5_neighbourhoods_fort_worth['mean'], width, label='Mean Price')
bars2 = plt.bar(x + width/2, top_5_neighbourhoods_fort_worth['median'], width, label='Median Price')

# Highlight the median price with a horizontal line
median_price_value_fort_worth = top_5_neighbourhoods_fort_worth['median'].median()
plt.axhline(median_price_value_fort_worth, color='red', linestyle='--', label=f'Median Price Line: ${median_price_value_fort_worth:.2f}')


plt.xlabel('Neighbourhood')
plt.ylabel('Price')
plt.title('Mean and Median Prices of Airbnb Listings by Neighbourhood in Fort Worth')
plt.xticks(ticks=x, labels=top_5_neighbourhoods_fort_worth['neighbourhood'], rotation=45, ha='right')
plt.legend()

# Plot count of listings
plt.subplot(1, 2, 2)
plt.bar(x, top_5_neighbourhoods_fort_worth['count'], width, color='skyblue')
plt.xlabel('Neighbourhood')
plt.ylabel('Number of Listings')
plt.title('Number of Airbnb Listings by Neighbourhood in Fort Worth')
plt.xticks(ticks=x, labels=top_5_neighbourhoods_fort_worth['neighbourhood'], rotation=45, ha='right')

# Highlight the median listing count with a horizontal line
median_listing_count_fort_worth = top_5_neighbourhoods_fort_worth['count'].median()

# Add a horizontal line for the median number of listings
plt.axhline(median_listing_count_fort_worth, color='blue', linestyle='--', label=f'Median Listing Count Line: {median_listing_count_fort_worth:.0f}')
plt.legend()

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



2.2 Predictive Modeling

At this point in the analysis we have leveraged our visual and honed in on Texas as a viable investment market. Specifically we have chosen the following:

Dallas and Fort Worth

[25]:
```
# Calculate average price per city
texas['avg_price_city'] = texas.groupby('city')['price'].transform('mean')

# Calculate average number of reviews per city
texas['avg_reviews_city'] = texas.groupby('city')['number_of_reviews'].
  ↪transform('mean')

# Calculate availability score
texas['availability_score'] = texas['availability_365'] / 365

# Create feature matrix and target vector
features = ['latitude', 'longitude', 'room_type', 'minimum_nights',
            'avg_price_city', 'avg_reviews_city', 'availability_score', 'city']
target = 'price'

X = texas[features]
y = texas[target]
```

[26]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

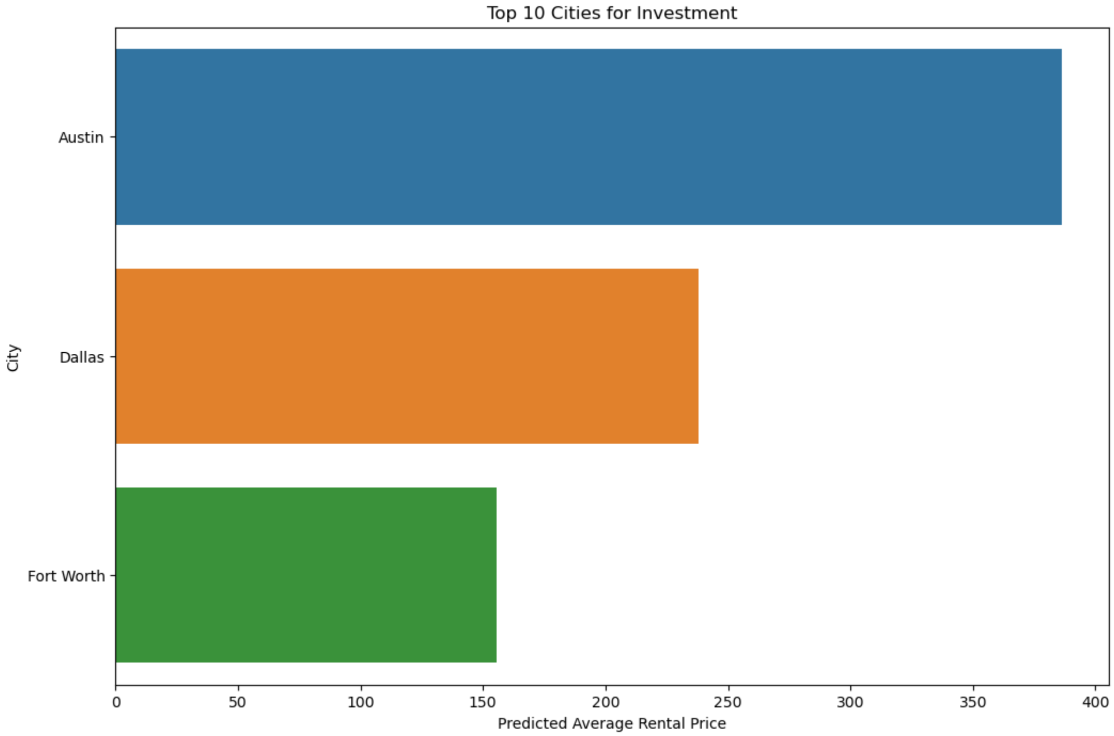
```python
# Define preprocessing pipelines
numeric_features = ['latitude', 'longitude', 'minimum_nights',
                    'avg_price_city', 'avg_reviews_city', 'availability_score']
categorical_features = ['room_type', 'city']

numeric_pipeline = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_pipeline = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_pipeline, numeric_features),
        ('cat', categorical_pipeline, categorical_features)
    ]
)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
 ↪random_state=42)

# Build and train the model
model_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(random_state=42))
])

model_pipeline.fit(X_train, y_train)

# Make predictions
y_pred = model_pipeline.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')
```

```
Mean Squared Error: 163683.01537288324
R^2 Score: 0.8692285285926519
```

[27]:
```
# Add predictions to the dataset
texas['predicted_price'] = model_pipeline.predict(X)

# Calculate average predicted price per city
avg_predicted_price_city = texas.groupby('city')['predicted_price'].mean().
 ↪reset_index()
avg_predicted_price_city = avg_predicted_price_city.
 ↪sort_values(by='predicted_price', ascending=False)

# Get top 10 cities
top_cities = avg_predicted_price_city.head(10)

print("Top 10 cities for investment:")
print(top_cities)
```

```
Top 10 cities for investment:
         city  predicted_price
0      Austin       386.279324
1      Dallas       237.854439
2  Fort Worth       155.381815
```

[28]:
```
# Plot top 10 cities for investment
plt.figure(figsize=(12, 8))
sns.barplot(x='predicted_price', y='city', data=top_cities)
plt.title('Top 10 Cities for Investment')
plt.xlabel('Predicted Average Rental Price')
plt.ylabel('City')
plt.show()
```



[29]:

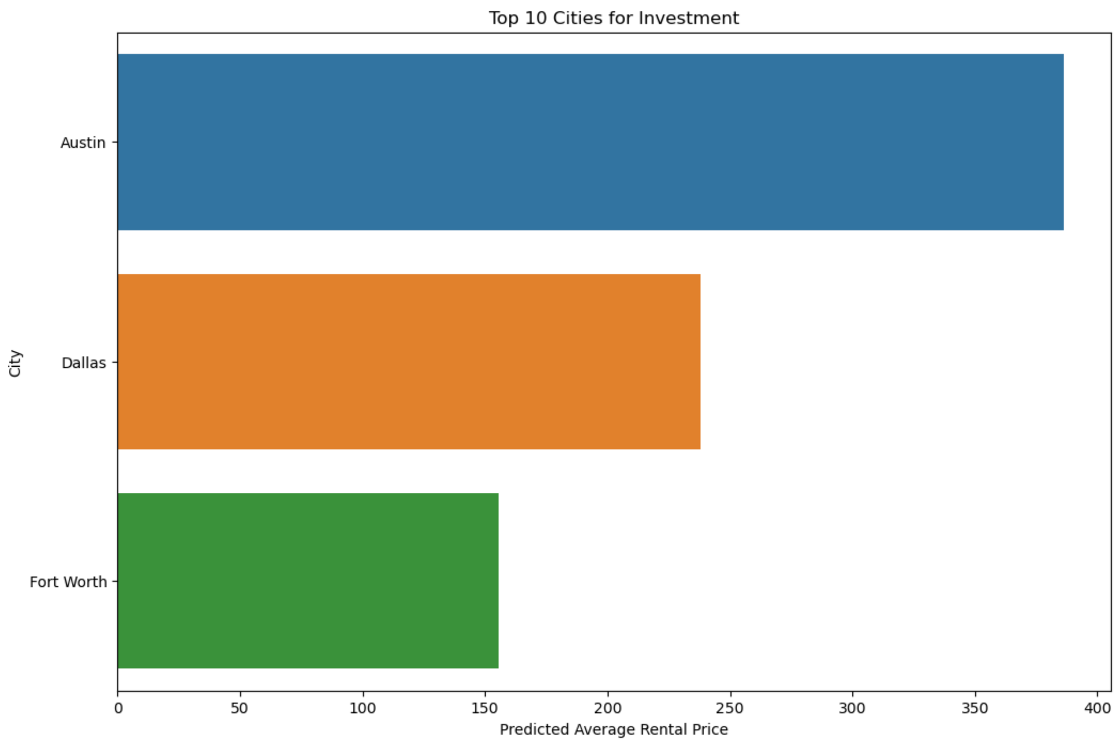
```
#Feature Engineering

# Calculate average price per neighborhood
texas['avg_price_neigh'] = texas.groupby('neighbourhood')['price'].
 ↪transform('mean')

# Calculate average number of reviews per neighborhood
texas['avg_reviews_neigh'] = texas.
 ↪groupby('neighbourhood')['number_of_reviews'].transform('mean')

# Calculate availability score
texas['availability_score'] = texas['availability_365'] / 365

# Create feature matrix and target vector
features = ['latitude', 'longitude', 'room_type', 'minimum_nights',
            'avg_price_neigh', 'avg_reviews_neigh', 'availability_score',␣
 ↪'city']
target = 'price'

X = texas[features]
y = texas[target]
```

[30]:
```
# Calculate average price per city
texas['avg_price_city'] = texas.groupby('city')['price'].transform('mean')

# Calculate average number of reviews per city
texas['avg_reviews_city'] = texas.groupby('city')['number_of_reviews'].
 ↪transform('mean')

# Calculate availability score
texas['availability_score'] = texas['availability_365'] / 365

# Create feature matrix and target vector
features = ['latitude', 'longitude', 'room_type', 'minimum_nights',
            'avg_price_city', 'avg_reviews_city', 'availability_score', 'city']
target = 'price'

X = texas[features]
y = texas[target]
```

[31]:
```
# Add predictions to the dataset
texas['predicted_price'] = model_pipeline.predict(X)

# Calculate average predicted price per city
avg_predicted_price_city = texas.groupby('city')['predicted_price'].mean().
 ↪reset_index()
avg_predicted_price_city = avg_predicted_price_city.
 ↪sort_values(by='predicted_price', ascending=False)

# Get top 10 cities
top_cities = avg_predicted_price_city.head(10)

print("Top 10 cities for investment:")
print(top_cities)
```

```
Top 10 cities for investment:
         city  predicted_price
0      Austin       386.279324
1      Dallas       237.854439
2  Fort Worth       155.381815
```

[32]:
```
# Plot top 10 cities for investment
plt.figure(figsize=(12, 8))
sns.barplot(x='predicted_price', y='city', data=top_cities)
plt.title('Top 10 Cities for Investment')
plt.xlabel('Predicted Average Rental Price')
plt.ylabel('City')
plt.show()
```



[33]:
```python
# Assuming top_cities DataFrame is available from the previous steps
top_city = top_cities['city'].iloc[0]  # Get the top city
df_city = texas[texas['city'] == top_city]

# Calculate average price per neighborhood
df_city['avg_price_neigh'] = df_city.groupby('neighbourhood')['price'].
    ↪transform('mean')

# Calculate average number of reviews per neighborhood
df_city['avg_reviews_neigh'] = df_city.
    ↪groupby('neighbourhood')['number_of_reviews'].transform('mean')

# Calculate availability score
df_city['availability_score'] = df_city['availability_365'] / 365

# Create feature matrix and target vector
features_neigh = ['latitude', 'longitude', 'room_type', 'minimum_nights',
                  'avg_price_neigh', 'avg_reviews_neigh', 'availability_score']
target_neigh = 'price'

X_neigh = df_city[features_neigh]
y_neigh = df_city[target_neigh]
```

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Define preprocessing pipelines
numeric_features_neigh = ['latitude', 'longitude', 'minimum_nights',
                          'avg_price_neigh', 'avg_reviews_neigh',␣
 ↪'availability_score']
categorical_features_neigh = ['room_type']

numeric_pipeline_neigh = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_pipeline_neigh = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor_neigh = ColumnTransformer(
    transformers=[
        ('num', numeric_pipeline_neigh, numeric_features_neigh),
        ('cat', categorical_pipeline_neigh, categorical_features_neigh)
    ]
)

# Train-test split
X_train_neigh, X_test_neigh, y_train_neigh, y_test_neigh =␣
 ↪train_test_split(X_neigh, y_neigh, test_size=0.3, random_state=42)

# Build and train the model
model_pipeline_neigh = Pipeline(steps=[
    ('preprocessor', preprocessor_neigh),
    ('regressor', RandomForestRegressor(random_state=42))
])

model_pipeline_neigh.fit(X_train_neigh, y_train_neigh)

# Make predictions
y_pred_neigh = model_pipeline_neigh.predict(X_test_neigh)

# Evaluate the model
mse_neigh = mean_squared_error(y_test_neigh, y_pred_neigh)
r2_neigh = r2_score(y_test_neigh, y_pred_neigh)
```

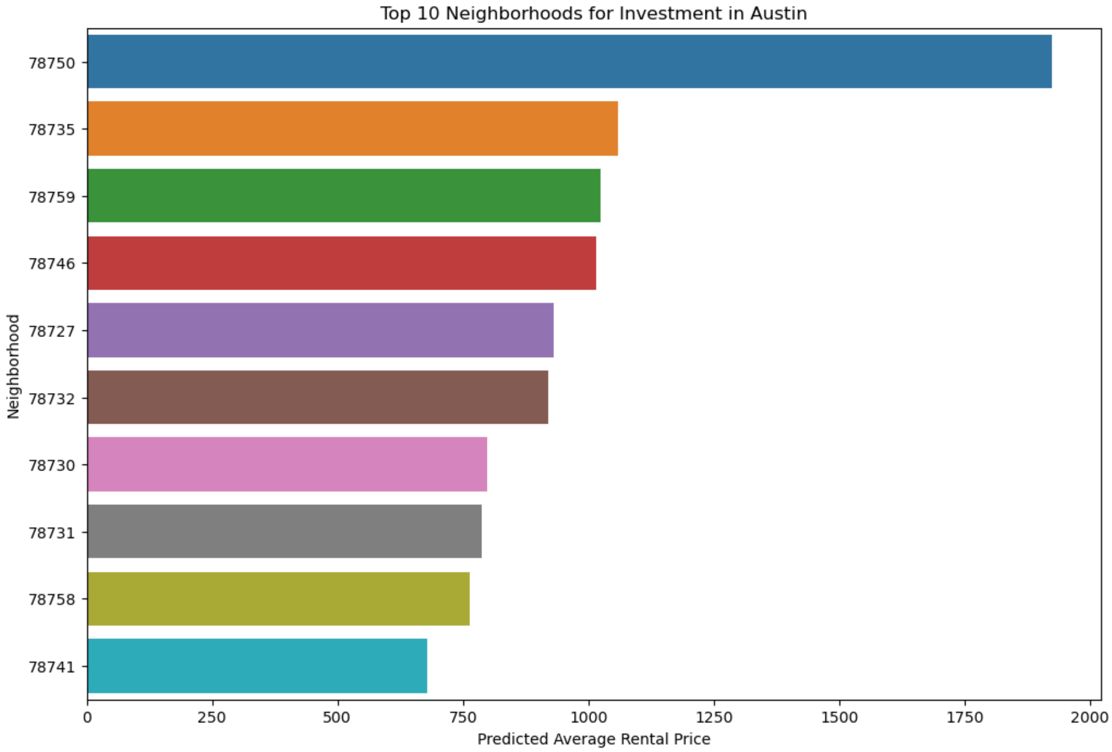
```
print(f'Mean Squared Error: {mse_neigh}')
print(f'R^2 Score: {r2_neigh}')

# Add predictions to the dataset
df_city['predicted_price'] = model_pipeline_neigh.predict(X_neigh)

# Calculate average predicted price per neighborhood
avg_predicted_price_neigh = df_city.groupby('neighbourhood')['predicted_price'].
 ↪mean().reset_index()
avg_predicted_price_neigh = avg_predicted_price_neigh.
 ↪sort_values(by='predicted_price', ascending=False)

# Get top 10 neighborhoods
top_neigh = avg_predicted_price_neigh.head(10)

print(f"Top 10 neighborhoods for investment in {top_city}:")
print(top_neigh)


# Plot top 10 neighborhoods for investment in the selected city
plt.figure(figsize=(12, 8))
sns.barplot(x='predicted_price', y='neighbourhood', data=top_neigh)
plt.title(f'Top 10 Neighborhoods for Investment in {top_city}')
plt.xlabel('Predicted Average Rental Price')
plt.ylabel('Neighborhood')
plt.show()
```

```
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/62496316.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_city['avg_price_neigh'] =
df_city.groupby('neighbourhood')['price'].transform('mean')
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/62496316.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_city['avg_reviews_neigh'] =
df_city.groupby('neighbourhood')['number_of_reviews'].transform('mean')
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/62496316.py:12:
```

```
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_city['availability_score'] = df_city['availability_365'] / 365
```

```
Mean Squared Error: 141546.71300135995
R^2 Score: 0.8973139425732697
Top 10 neighborhoods for investment in Austin:
   neighbourhood  predicted_price
70         78750      1924.693800
57         78735      1058.856417
78         78759      1024.308203
66         78746      1015.013500
49         78727       931.417881
54         78732       919.099000
52         78730       797.261154
53         78731       788.439840
77         78758       763.950419
62         78741       679.123314
```

```
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/62496316.py:72:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_city['predicted_price'] = model_pipeline_neigh.predict(X_neigh)
```



[34]:
```
#Do the same for dallas and fort worth

# Filter for state = TX and specific cities
texas = df[df['state'] == 'TX']
df_dallas = texas[texas['city'] == 'Dallas']
df_fortworth = texas[texas['city'] == 'Fort Worth']

# For Dallas
df_dallas['avg_price_neigh'] = df_dallas.groupby('neighbourhood')['price'].
↪transform('mean')
df_dallas['avg_reviews_neigh'] = df_dallas.
↪groupby('neighbourhood')['number_of_reviews'].transform('mean')
df_dallas['availability_score'] = df_dallas['availability_365'] / 365

# For Fort Worth
df_fortworth['avg_price_neigh'] = df_fortworth.
↪groupby('neighbourhood')['price'].transform('mean')
df_fortworth['avg_reviews_neigh'] = df_fortworth.
↪groupby('neighbourhood')['number_of_reviews'].transform('mean')
df_fortworth['availability_score'] = df_fortworth['availability_365'] / 365
```

```python
# Features and target for Dallas
features_neigh = ['latitude', 'longitude', 'room_type', 'minimum_nights',
                  'avg_price_neigh', 'avg_reviews_neigh', 'availability_score']
target_neigh = 'price'

X_dallas = df_dallas[features_neigh]
y_dallas = df_dallas[target_neigh]

# Features and target for Fort Worth
X_fortworth = df_fortworth[features_neigh]
y_fortworth = df_fortworth[target_neigh]

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Define preprocessing pipelines
numeric_features_neigh = ['latitude', 'longitude', 'minimum_nights',
                          'avg_price_neigh', 'avg_reviews_neigh',␣
 ↪'availability_score']
categorical_features_neigh = ['room_type']

numeric_pipeline_neigh = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_pipeline_neigh = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor_neigh = ColumnTransformer(
    transformers=[
        ('num', numeric_pipeline_neigh, numeric_features_neigh),
        ('cat', categorical_pipeline_neigh, categorical_features_neigh)
    ]
)

# Model training function
def train_model(X, y):
    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

    # Build and train the model
```

```
    model_pipeline = Pipeline(steps=[
        ('preprocessor', preprocessor_neigh),
        ('regressor', RandomForestRegressor(random_state=42))
    ])

    model_pipeline.fit(X_train, y_train)

    # Make predictions
    y_pred = model_pipeline.predict(X_test)

    # Evaluate the model
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f'Mean Squared Error: {mse}')
    print(f'R^2 Score: {r2}')

    return model_pipeline

# Train models for Dallas and Fort Worth
model_dallas = train_model(X_dallas, y_dallas)
model_fortworth = train_model(X_fortworth, y_fortworth)


# Add predictions to the datasets
df_dallas['predicted_price'] = model_dallas.predict(X_dallas)
df_fortworth['predicted_price'] = model_fortworth.predict(X_fortworth)

# Calculate average predicted price per neighborhood
avg_predicted_price_neigh_dallas = df_dallas.
 ↪groupby('neighbourhood')['predicted_price'].mean().reset_index()
avg_predicted_price_neigh_dallas = avg_predicted_price_neigh_dallas.
 ↪sort_values(by='predicted_price', ascending=False)

avg_predicted_price_neigh_fortworth = df_fortworth.
 ↪groupby('neighbourhood')['predicted_price'].mean().reset_index()
avg_predicted_price_neigh_fortworth = avg_predicted_price_neigh_fortworth.
 ↪sort_values(by='predicted_price', ascending=False)

# Get top 10 neighborhoods
top_neigh_dallas = avg_predicted_price_neigh_dallas.head(10)
top_neigh_fortworth = avg_predicted_price_neigh_fortworth.head(10)

print("Top 10 neighborhoods for investment in Dallas:")
print(top_neigh_dallas)

print("Top 10 neighborhoods for investment in Fort Worth:")
```

```python
print(top_neigh_fortworth)


# Plot top 10 neighborhoods for investment in Dallas
plt.figure(figsize=(12, 8))
sns.barplot(x='predicted_price', y='neighbourhood', data=top_neigh_dallas)
plt.title('Top 10 Neighborhoods for Investment in Dallas')
plt.xlabel('Predicted Average Rental Price')
plt.ylabel('Neighborhood')
plt.show()

# Plot top 10 neighborhoods for investment in Fort Worth
plt.figure(figsize=(12, 8))
sns.barplot(x='predicted_price', y='neighbourhood', data=top_neigh_fortworth)
plt.title('Top 10 Neighborhoods for Investment in Fort Worth')
plt.xlabel('Predicted Average Rental Price')
plt.ylabel('Neighborhood')
plt.show()
```

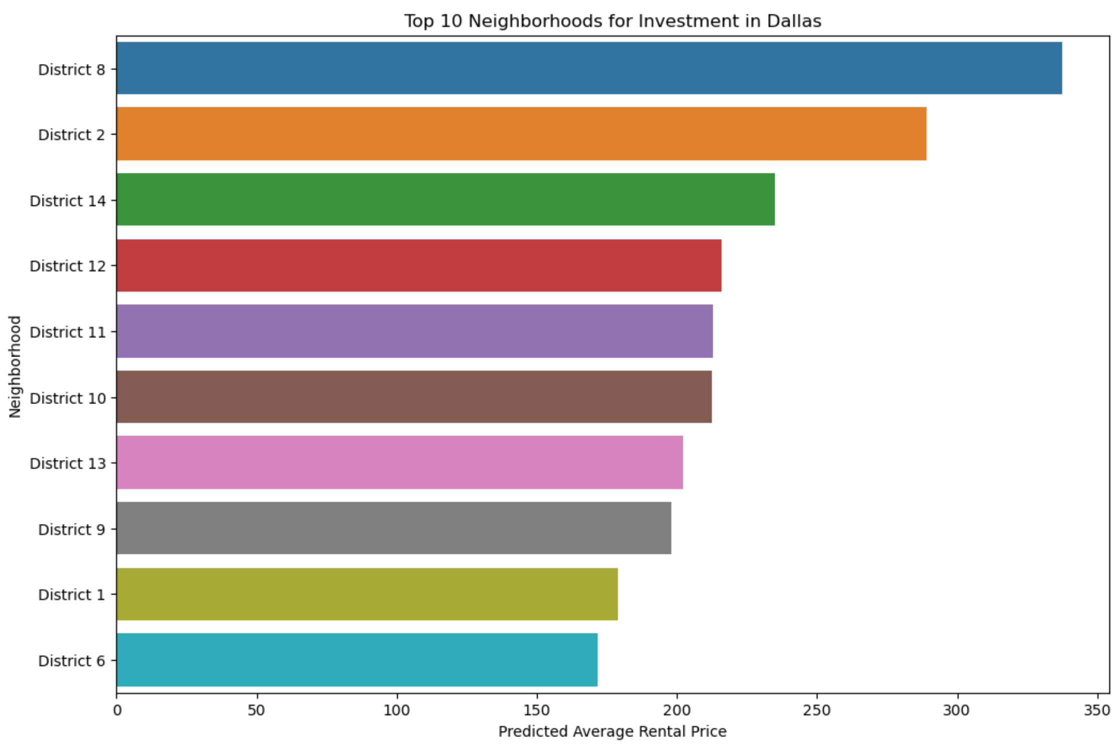
```
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/2612041528.py:9
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_dallas['avg_price_neigh'] =
df_dallas.groupby('neighbourhood')['price'].transform('mean')
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/2612041528.py:1
0: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_dallas['avg_reviews_neigh'] =
df_dallas.groupby('neighbourhood')['number_of_reviews'].transform('mean')
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/2612041528.py:1
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_dallas['availability_score'] = df_dallas['availability_365'] / 365
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/2612041528.py:1
4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

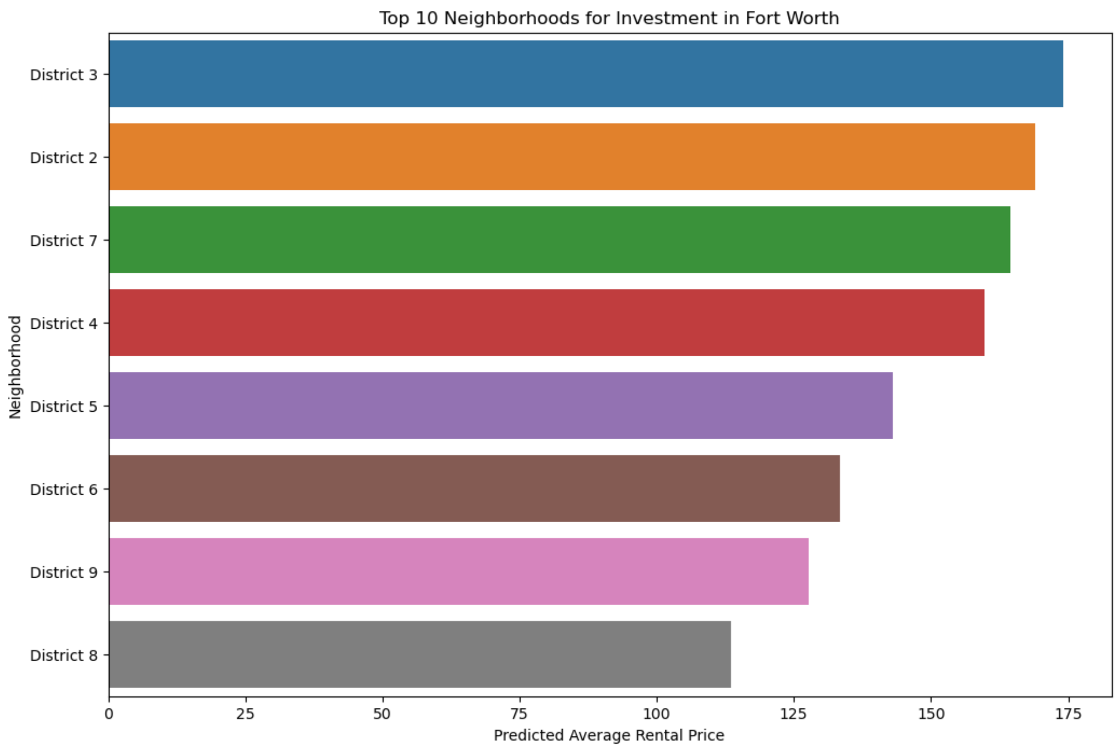
```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_fortworth['avg_price_neigh'] =
df_fortworth.groupby('neighbourhood')['price'].transform('mean')
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/2612041528.py:1
5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_fortworth['avg_reviews_neigh'] =
df_fortworth.groupby('neighbourhood')['number_of_reviews'].transform('mean')
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/2612041528.py:1
6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_fortworth['availability_score'] = df_fortworth['availability_365'] / 365
```

```
Mean Squared Error: 141619.07048061202
R^2 Score: 0.7730946576305356
Mean Squared Error: 8465.688019269104
R^2 Score: 0.21286680295048543
Top 10 neighborhoods for investment in Dallas:
   neighbourhood  predicted_price
12    District 8       337.544800
6     District 2       289.084598
5    District 14       235.023474
3    District 12       216.068317
2    District 11       213.012285
1    District 10       212.544745
4    District 13       202.136601
13    District 9       198.052663
0     District 1       179.120472
10    District 6       171.902486
Top 10 neighborhoods for investment in Fort Worth:
  neighbourhood  predicted_price
1    District 3       174.155172
0    District 2       168.901558
5    District 7       164.387788
2    District 4       159.688976
3    District 5       143.116947
4    District 6       133.353929
```

```
7    District 9       127.715966
6    District 8       113.474444
```

```
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/2612041528.py:8
9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_dallas['predicted_price'] = model_dallas.predict(X_dallas)
/var/folders/rm/wtgg2sx94jq9c87_mpvgkj700000gn/T/ipykernel_27801/2612041528.py:9
0: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_fortworth['predicted_price'] = model_fortworth.predict(X_fortworth)
```





Utilizing the predictive modeling, stakeholders will not only be reaffirmed in their decision to invest in Texas Real Estate, but they can also feel additionally confident in picking a neighborhood that poses a substantial return on investment.

This can be done by leveraging the above predicted rental price measures as well as the labeled top neighborhoods to invest in!