HOUSE PRICE PRIDICTION

Introduction

This project uses the `kc_house_data.csv` dataset to predict house prices. Two models are used:

- Linear Regression
- Gradient Boosting Regressor

Prerequisites

- Python 3.x installed
- Libraries: numpy, pandas, matplotlib, seaborn, scikit-learn

Setup

1. Clone or download the project repository.
2. Install the necessary packages:

   ```
   pip install numpy pandas matplotlib seaborn scikit-learn
   ```

Data Exploration

Load and explore the dataset to understand its structure and content.

```
import pandas as pd

# Load data
data = pd.read_csv("kc_house_data.csv")

# Data exploration
print(data.head())
print(data.describe())
```

Data Visualization

Visualize the dataset to identify patterns and relationships between features and the target variable (price).

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

# Plotting the number of bedrooms
data['bedrooms'].value_counts().plot(kind='bar')
plt.title('Number of Bedrooms')
plt.xlabel('Bedrooms')
plt.ylabel('Count')
plt.show()

# Joint plot of latitude and longitude
```

```
plt.figure(figsize=(10, 10))
sns.jointplot(x=data.lat.values, y=data.long.values, size=10)
plt.ylabel('Longitude', fontsize=12)
plt.xlabel('Latitude', fontsize=12)
plt.show()

# Scatter plots for various features against price
plt.scatter(data.price, data.sqft_living)
plt.title("Price vs Square Feet")
plt.show()

plt.scatter(data.price, data.long)
plt.title("Price vs Location of the area")
plt.show()

plt.scatter(data.price, data.lat)
plt.xlabel("Price")
plt.ylabel('Latitude')
plt.title("Latitude vs Price")
plt.show()

plt.scatter(data.bedrooms, data.price)
plt.title("Bedroom and Price ")
plt.xlabel("Bedrooms")
plt.ylabel("Price")
plt.show()

plt.scatter((data['sqft_living'] + data['sqft_basement']), data['price'])
plt.show()

plt.scatter(data.waterfront, data.price)
plt.title("Waterfront vs Price (0= no waterfront)")
plt.show()

plt.scatter(data.floors, data.price)
plt.show()

plt.scatter(data.condition, data.price)
plt.show()

plt.scatter(data.zipcode, data.price)
plt.title("Which is the pricey location by zipcode?")
plt.show()
```

Modeling

Train and evaluate Linear Regression and Gradient Boosting Regressor models.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor

# Prepare data for modeling
labels = data['price']
conv_dates = [1 if values == 2014 else 0 for values in data.date]
data['date'] = conv_dates
train1 = data.drop(['id', 'price'], axis=1)
x_train, x_test, y_train, y_test = train_test_split(train1, labels,
test_size=0.10, random_state=2)
```

```
# Linear Regression model
reg = LinearRegression()
reg.fit(x_train, y_train)
print("Linear Regression Score:", reg.score(x_test, y_test))

# Gradient Boosting Regressor model
clf = GradientBoostingRegressor(n_estimators=400, max_depth=5,
min_samples_split=2, learning_rate=0.1, loss='ls')
clf.fit(x_train, y_train)
print("Gradient Boosting Regressor Score:", clf.score(x_test, y_test))
```

Principal Component Analysis (PCA)

Apply PCA to reduce dimensionality and visualize the data.

```
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA

# PCA
pca = PCA()
pca_data = pca.fit_transform(scale(train1))
print(pca_data)
```

Results

The scores of the models on the test data are printed to the console.

- Linear Regression Score: This score indicates how well the Linear Regression model performs on the test set.
- Gradient Boosting Regressor Score: This score indicates how well the Gradient Boosting Regressor model performs on the test set.

Results of sample:
Linear Regression Score: 0.7320342760357805
Gradient Boosting Regressor Score: 0.9198120041384449

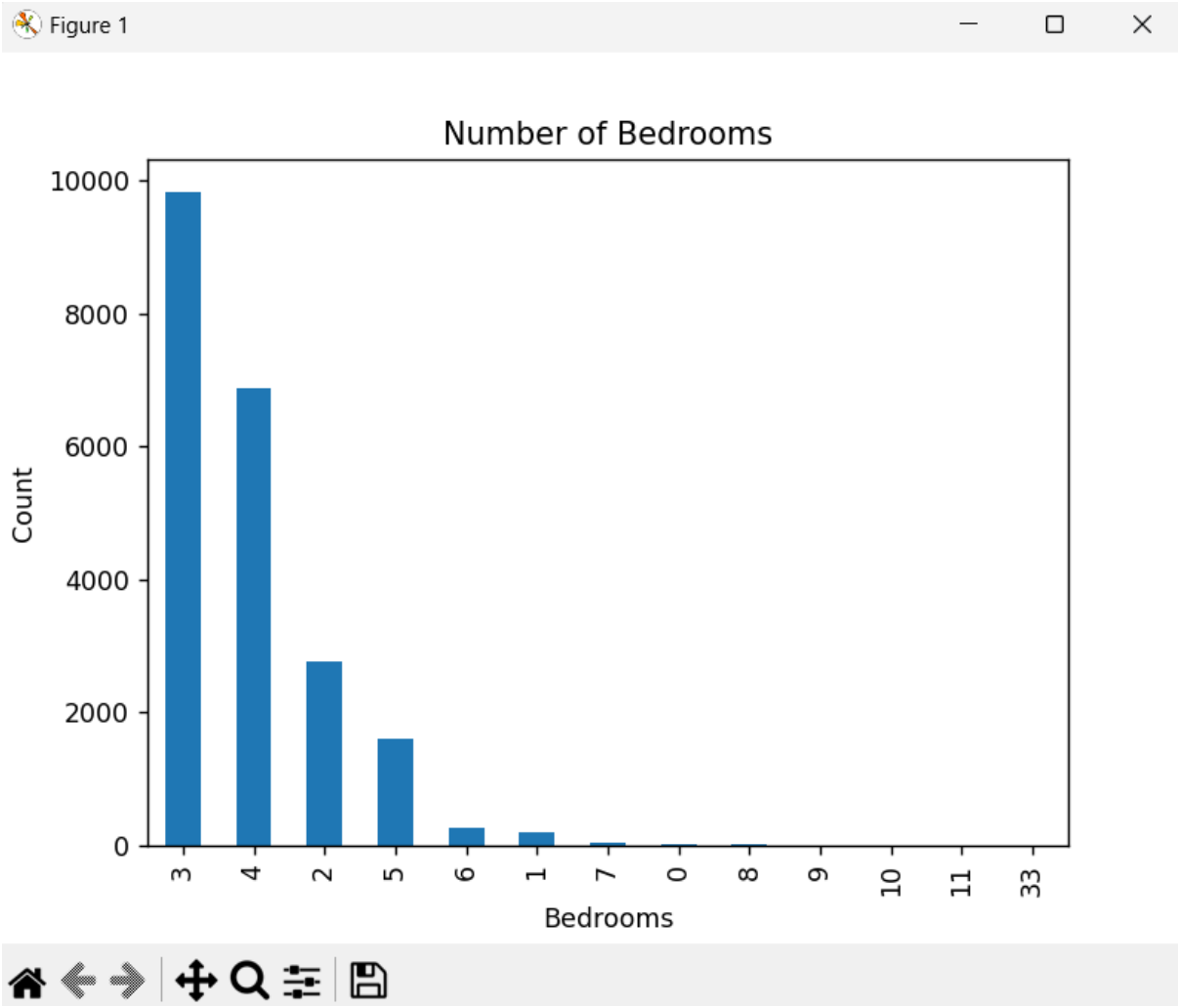


Figure 1
Number of Bedrooms
10000
8000
6000
4000
2000
0
Count
3
4
2
5
6
1
7
0
8
9
10
11
33
Bedrooms

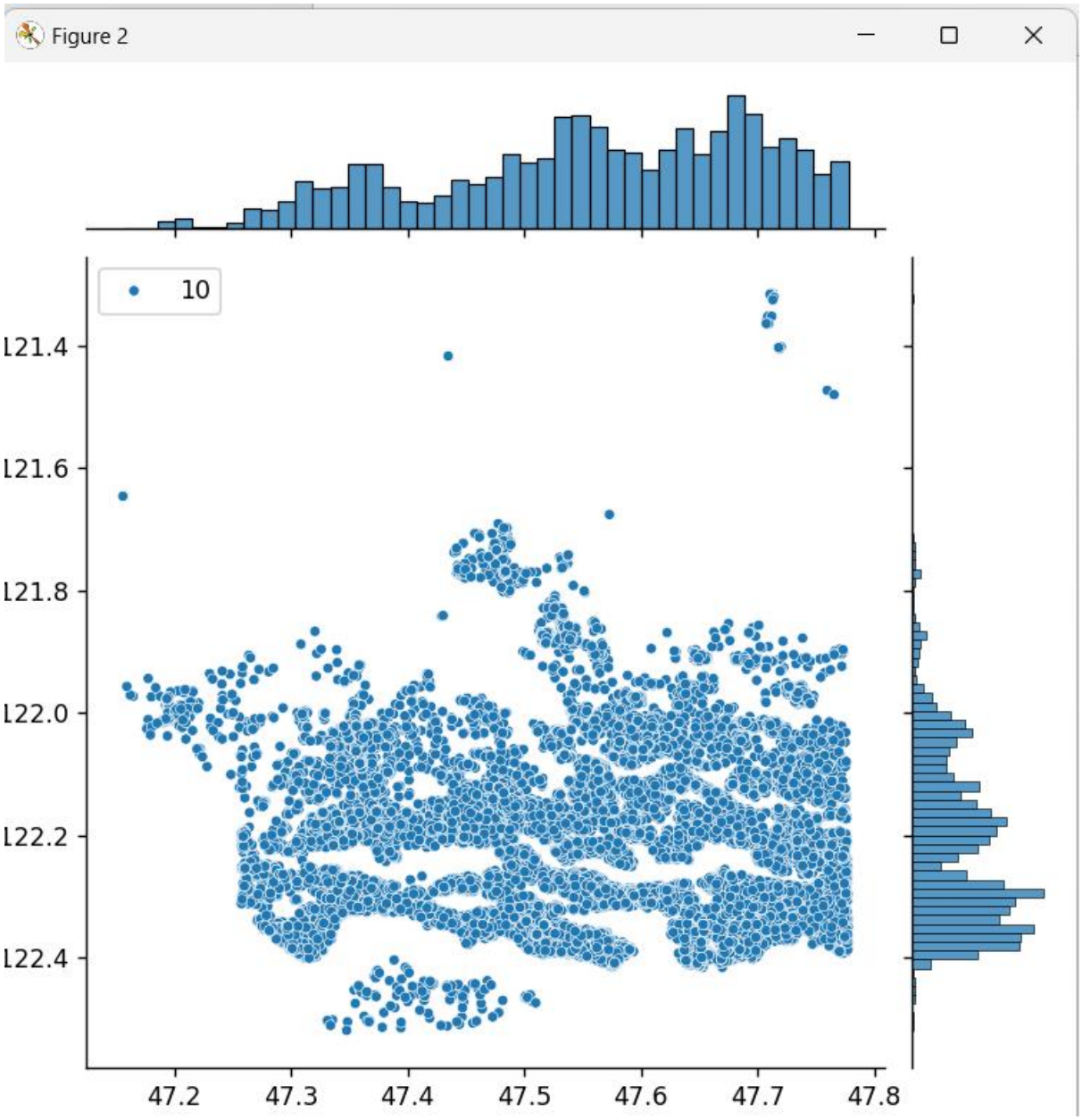


Figure 2
10
121.4
121.6
121.8
122.0
122.2
122.4
47.2
47.3
47.4
47.5
47.6
47.7
47.8

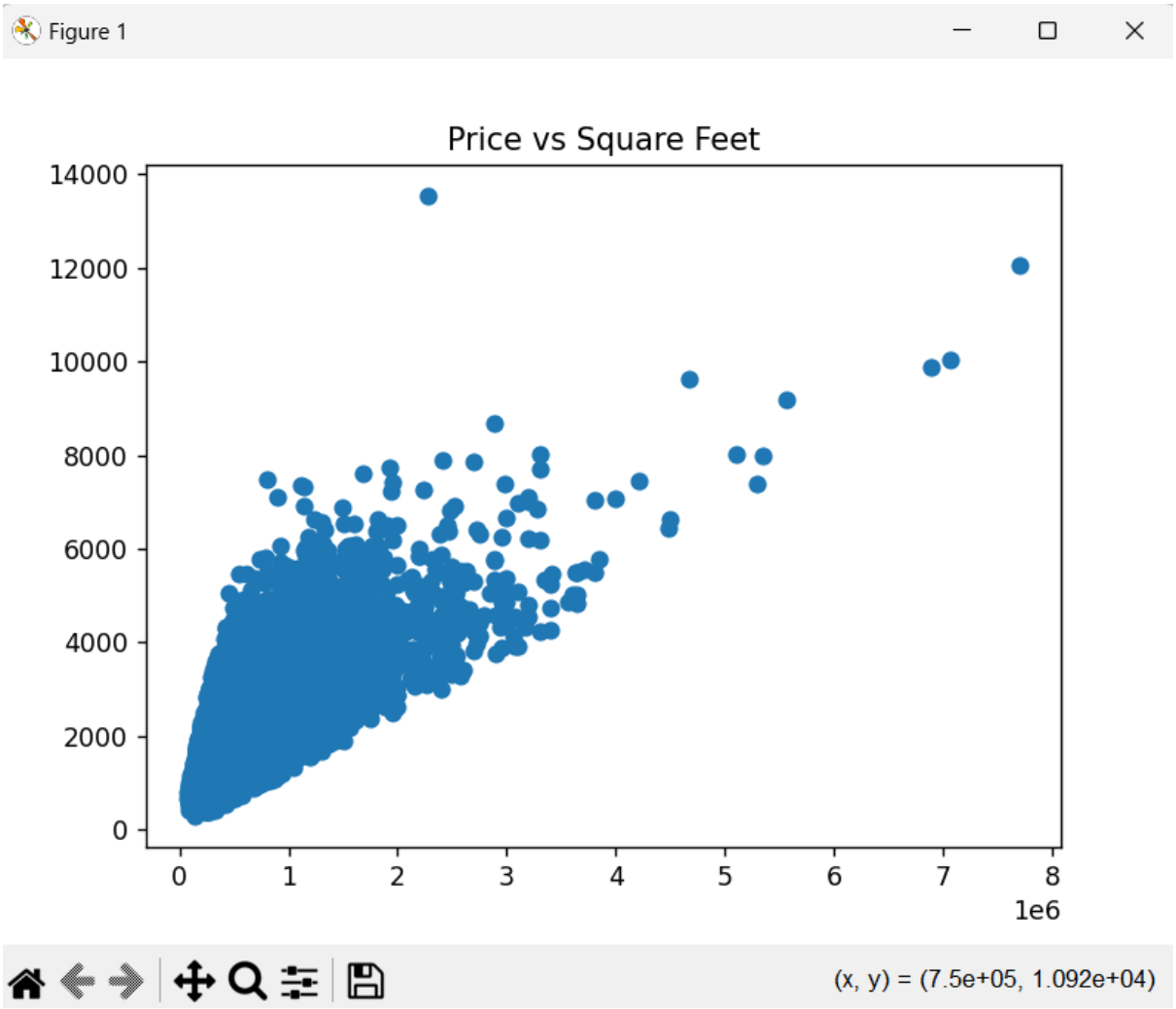


Figure 1
Price vs Square Feet
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
6
7
8
1e6
(x, y) = (7.5e+05, 1.092e+04)

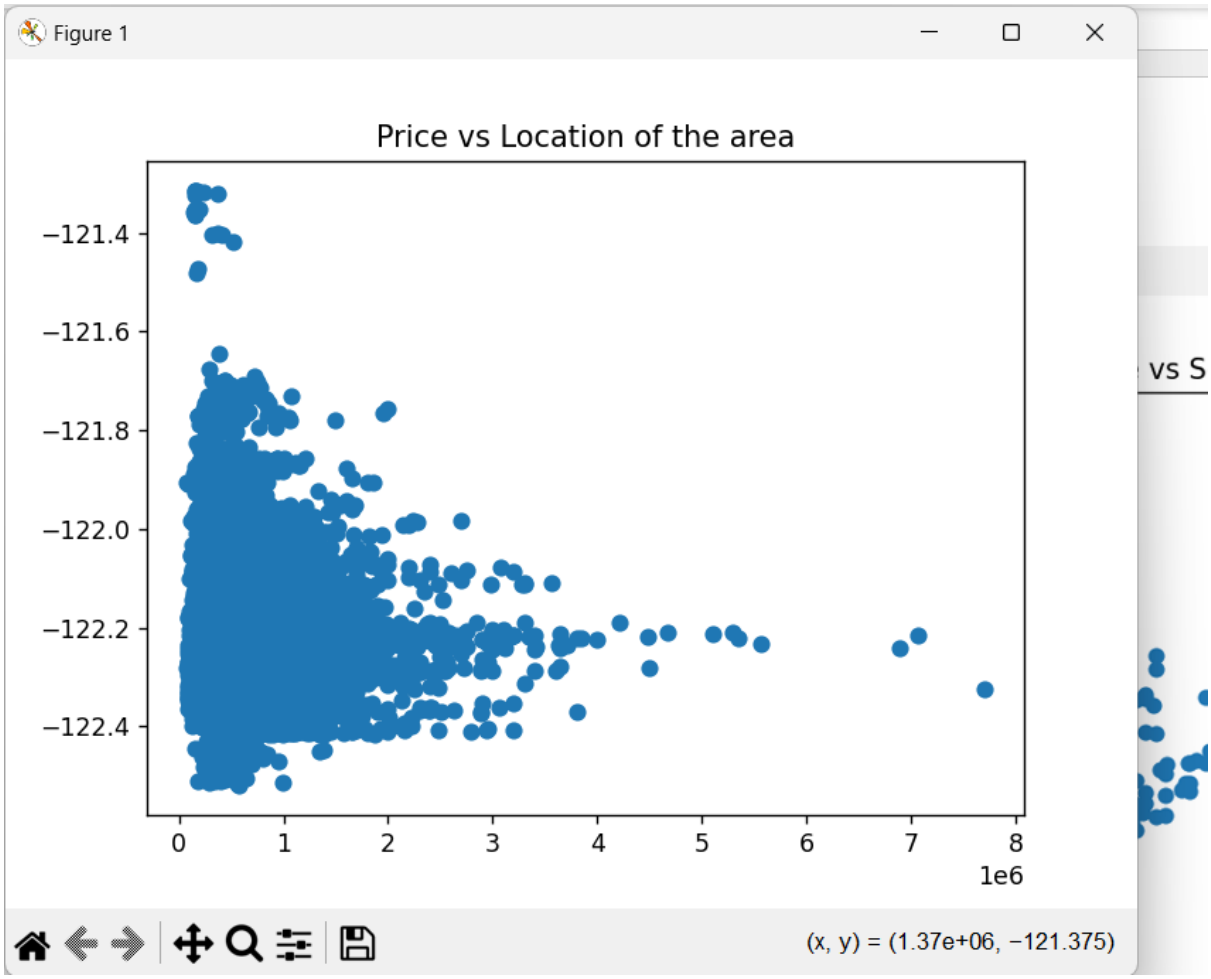


Figure 1
Price vs Location of the area
−121.4
−121.6
−121.8
−122.0
−122.2
−122.4
0
1
2
3
4
5
6
7
8
1e6
(x, y) = (1.37e+06, −121.375)

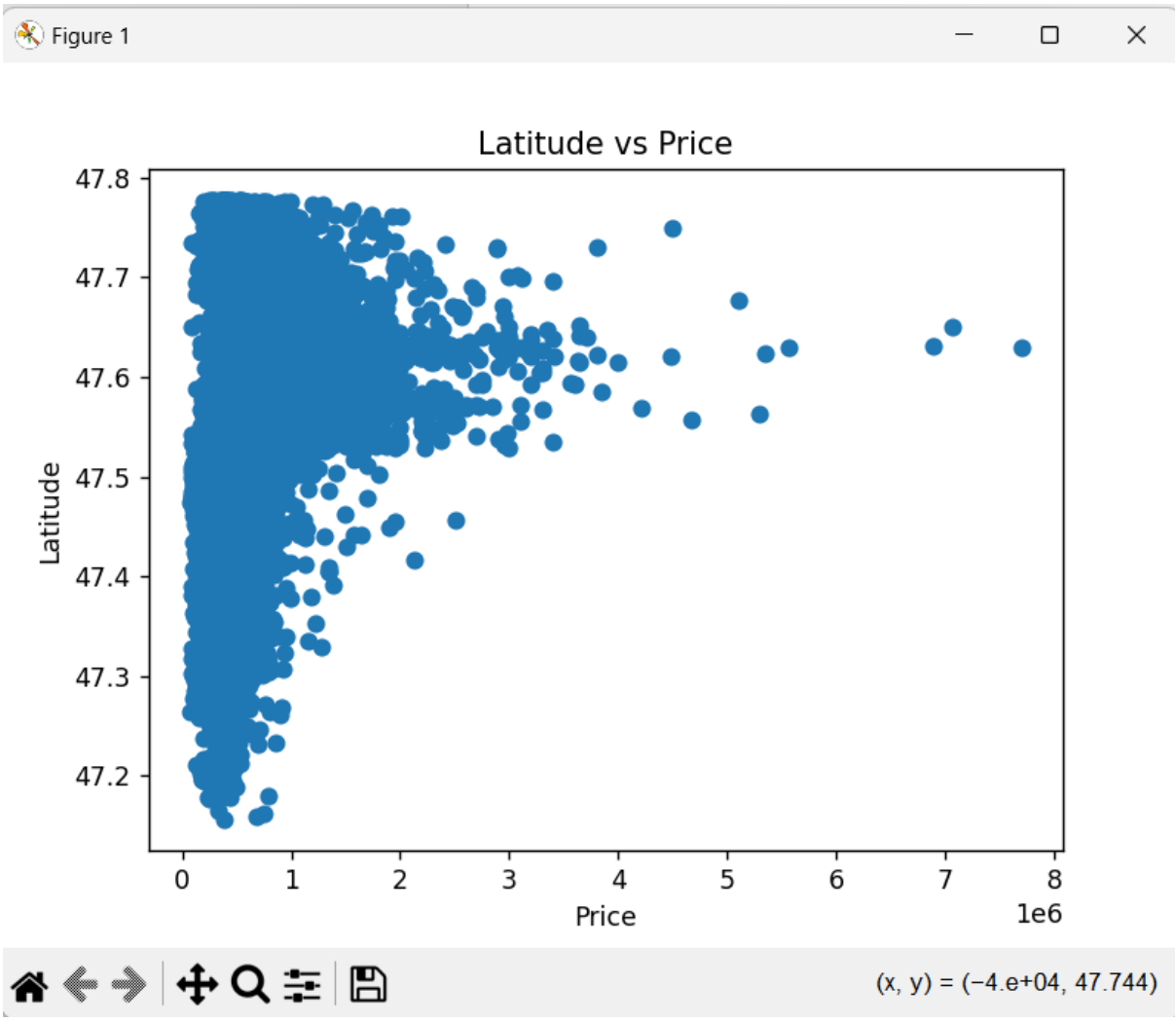


Figure 1
Latitude vs Price
47.8
47.7
47.6
47.5
47.4
47.3
47.2
Latitude
0
1
2
3
4
5
6
7
8
Price
1e6
(x, y) = (−4.e+04, 47.744)

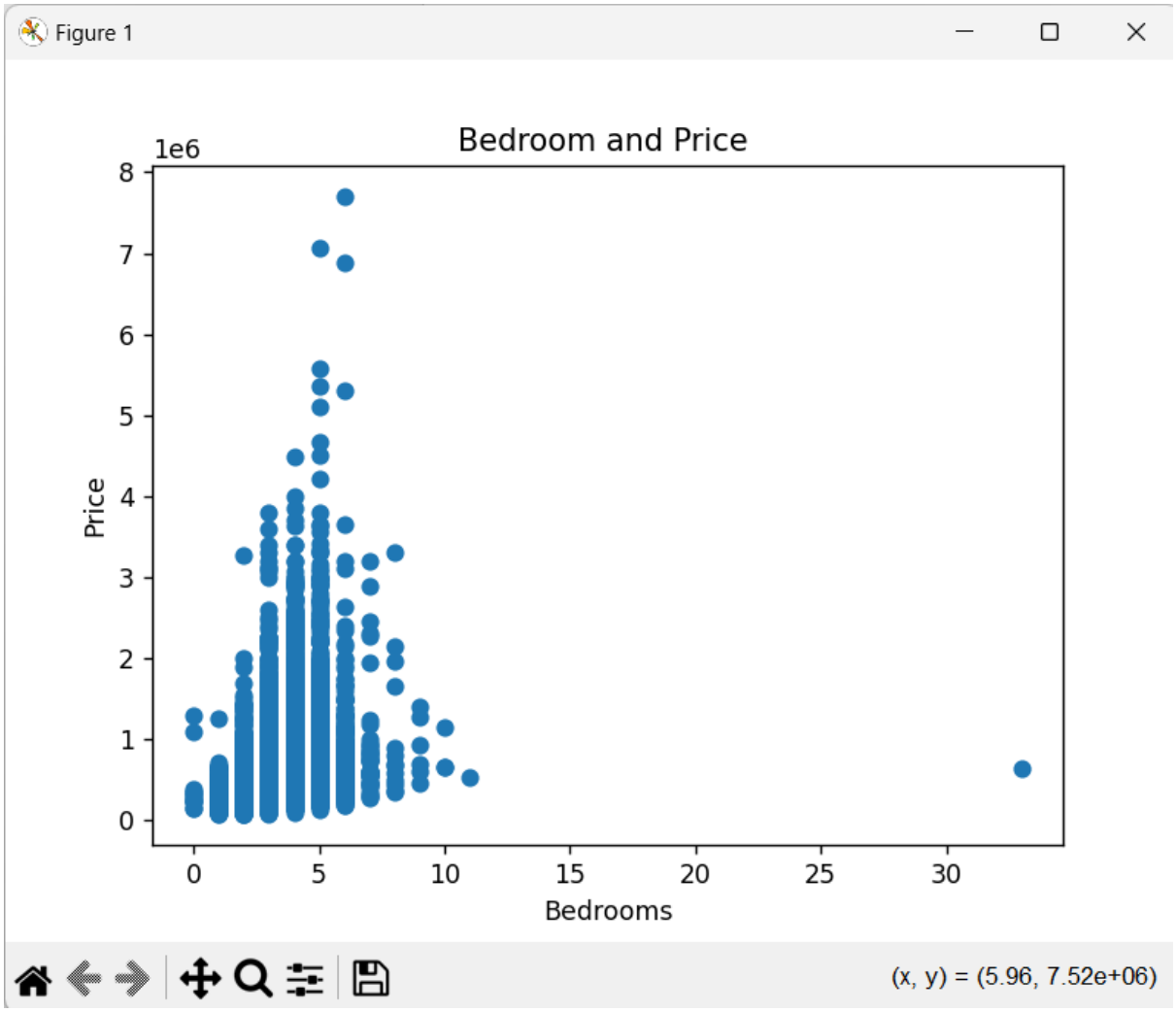


Figure 1
Bedroom and Price
1e6
8
7
6
5
4
3
2
1
0
Price
0
5
10
15
20
25
30
Bedrooms
(x, y) = (5.96, 7.52e+06)

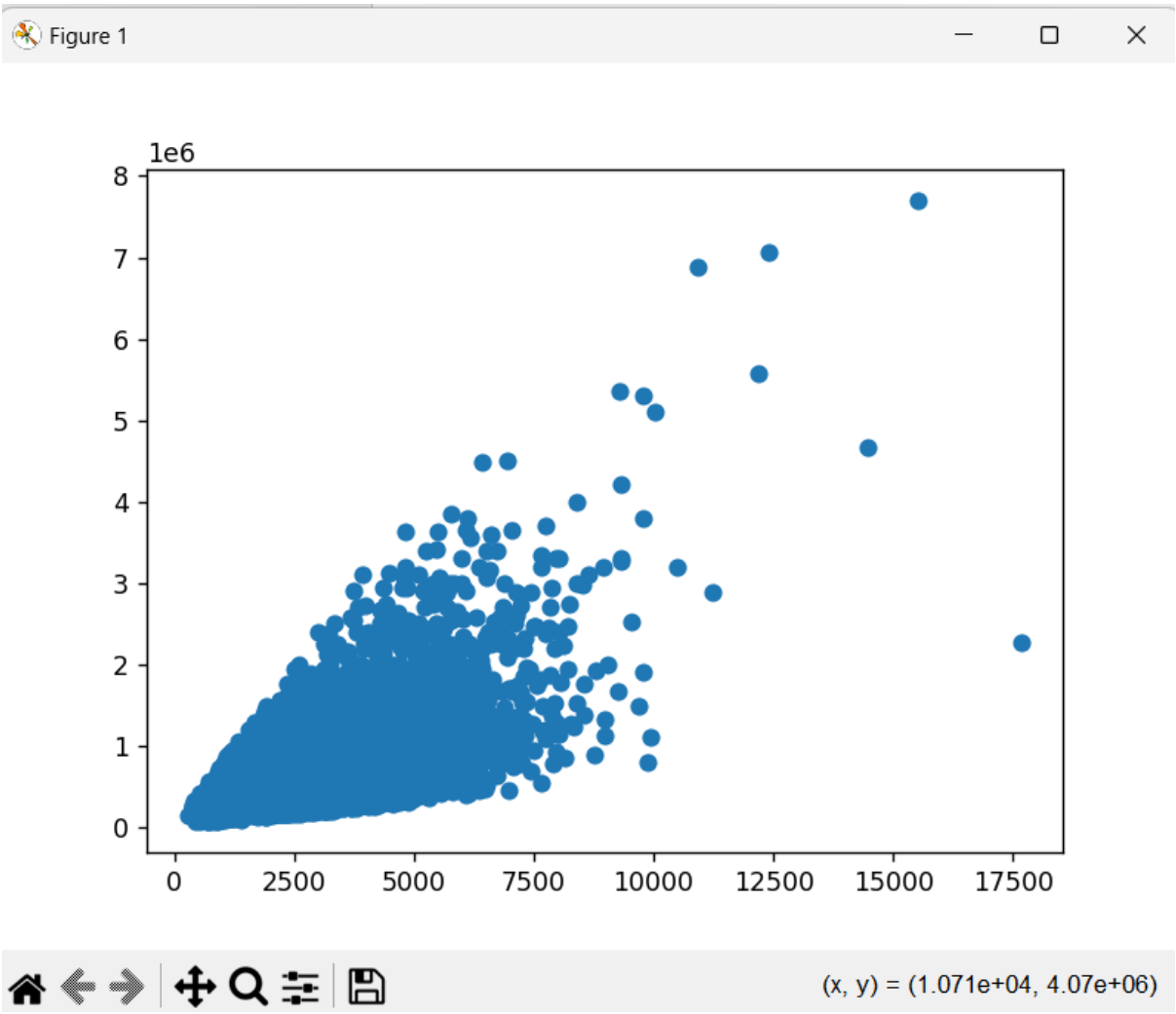


Figure 1
1e6
8
7
6
5
4
3
2
1
0
0
2500
5000
7500
10000
12500
15000
17500
(x, y) = (1.071e+04, 4.07e+06)

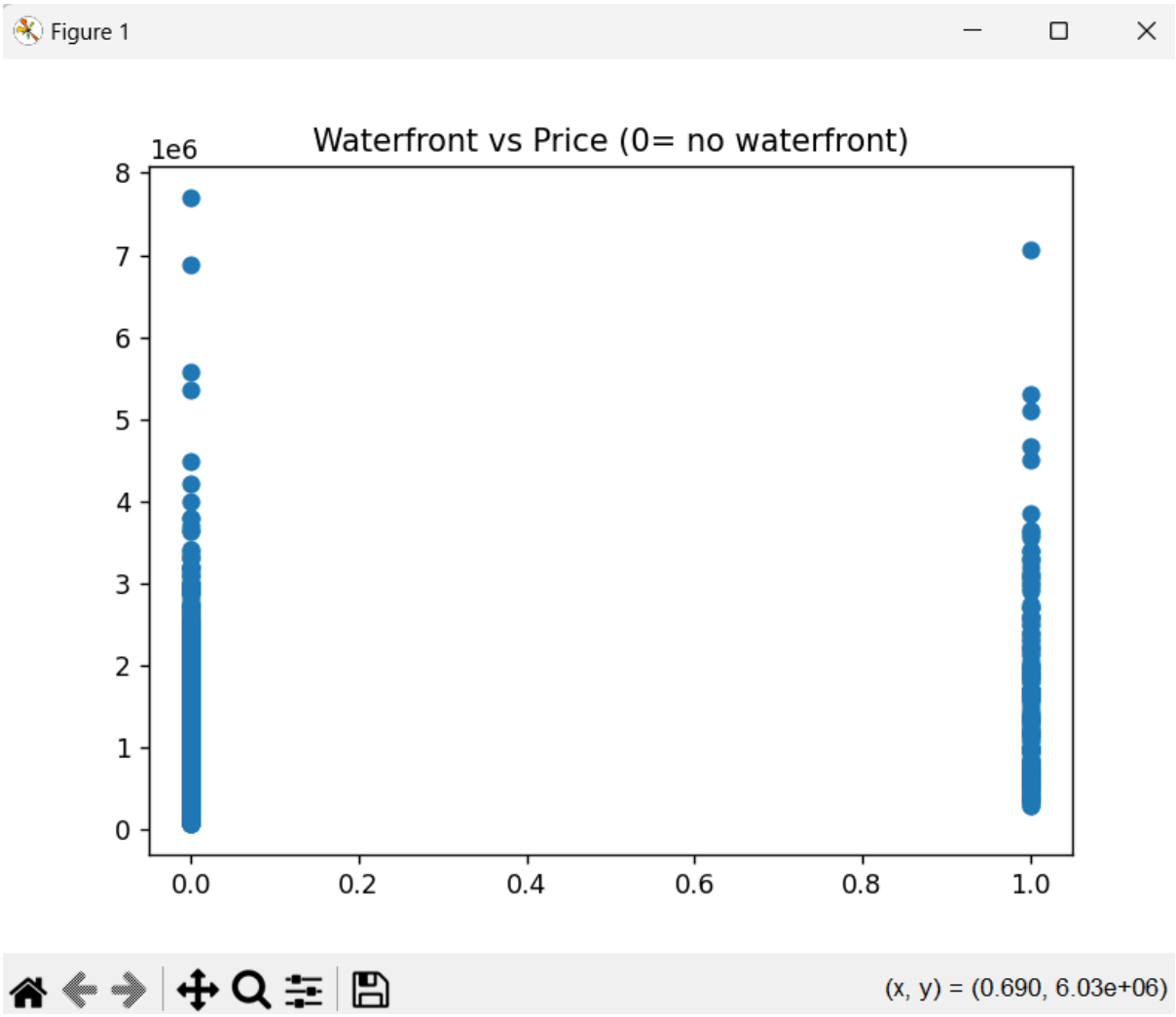


Figure 1
Waterfront vs Price (0= no waterfront)
1e6
8
7
6
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
(x, y) = (0.690, 6.03e+06)

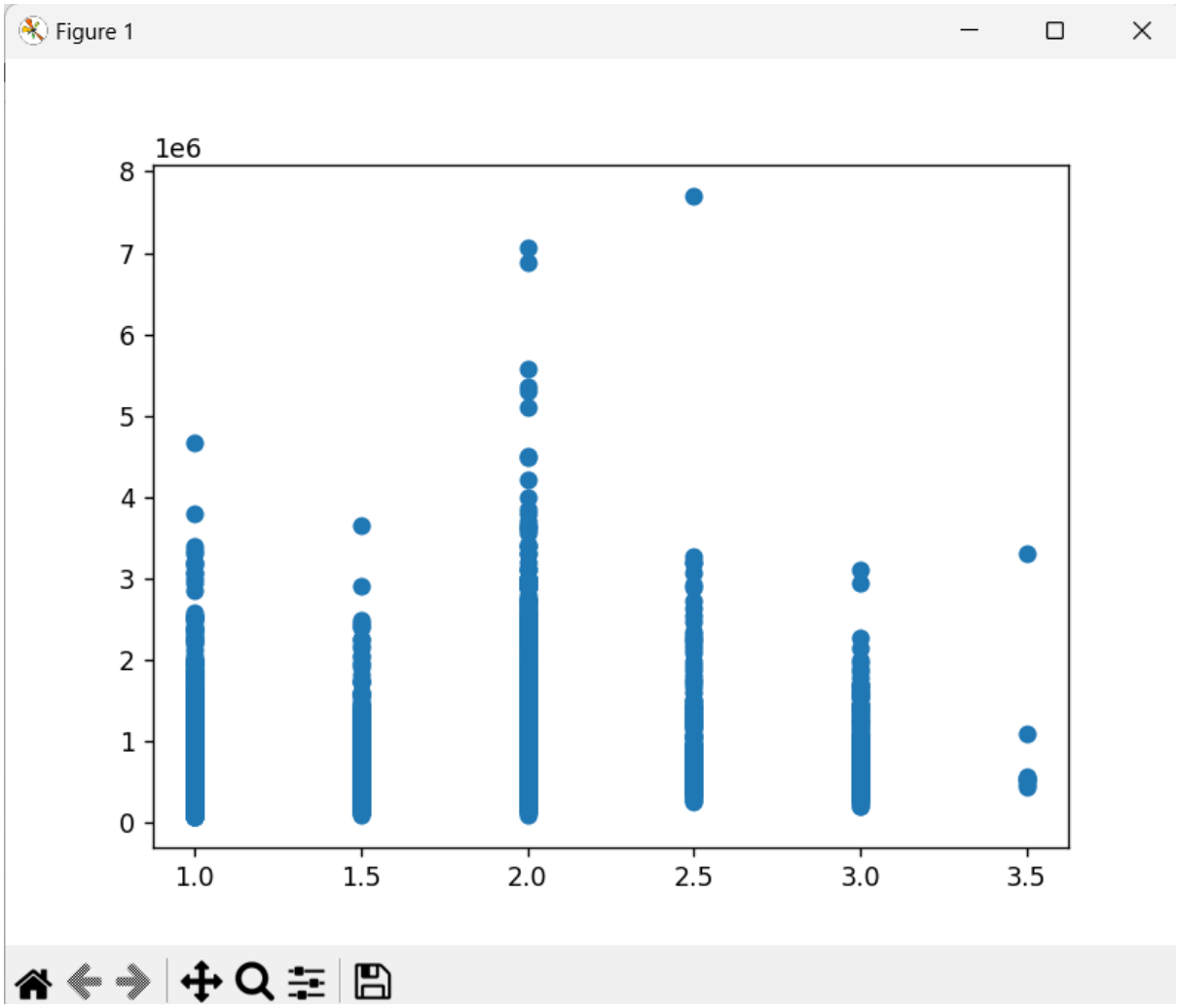


Figure 1
1e6
8
7
6
5
4
3
2
1
0
1.0
1.5
2.0
2.5
3.0
3.5

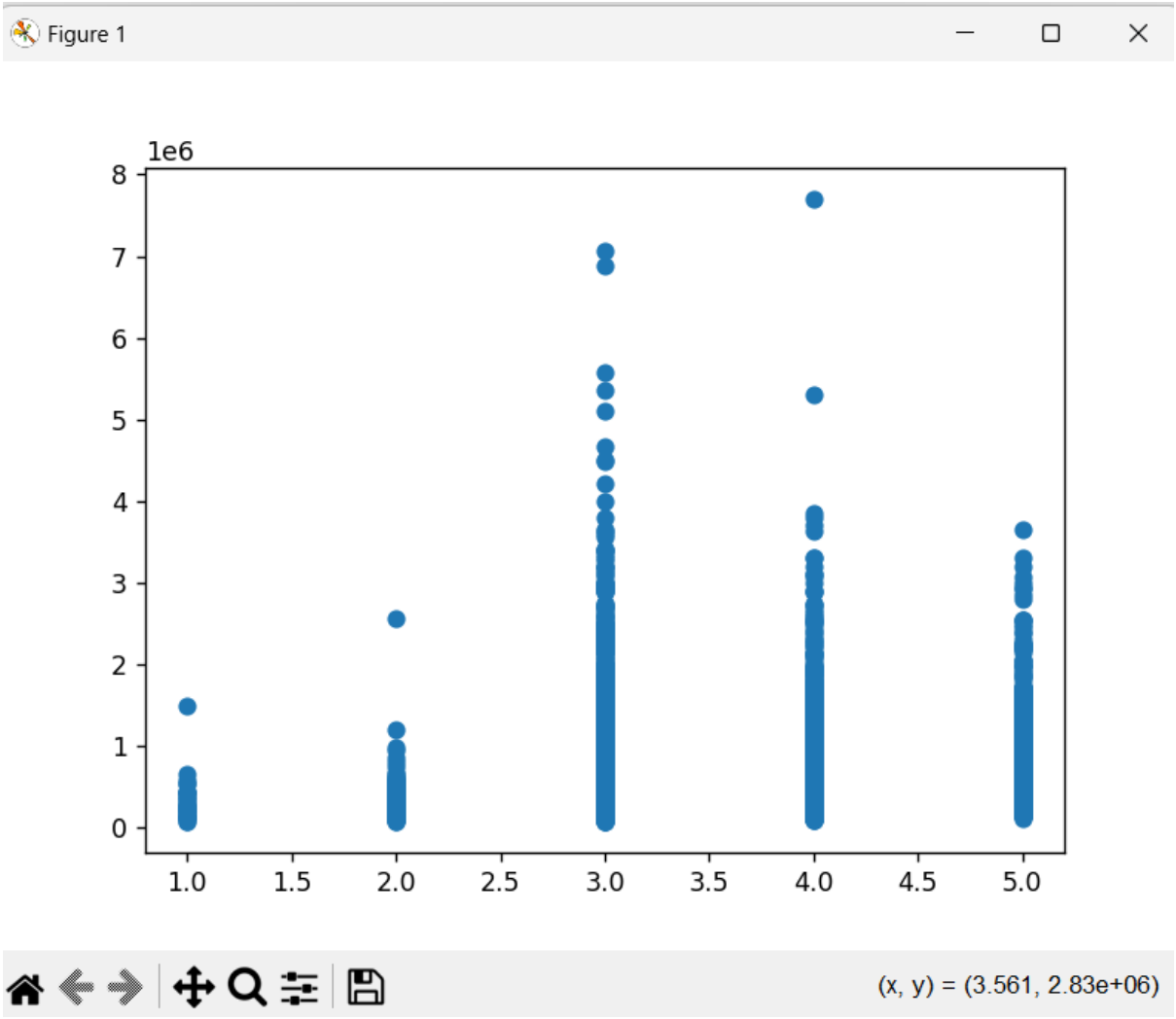
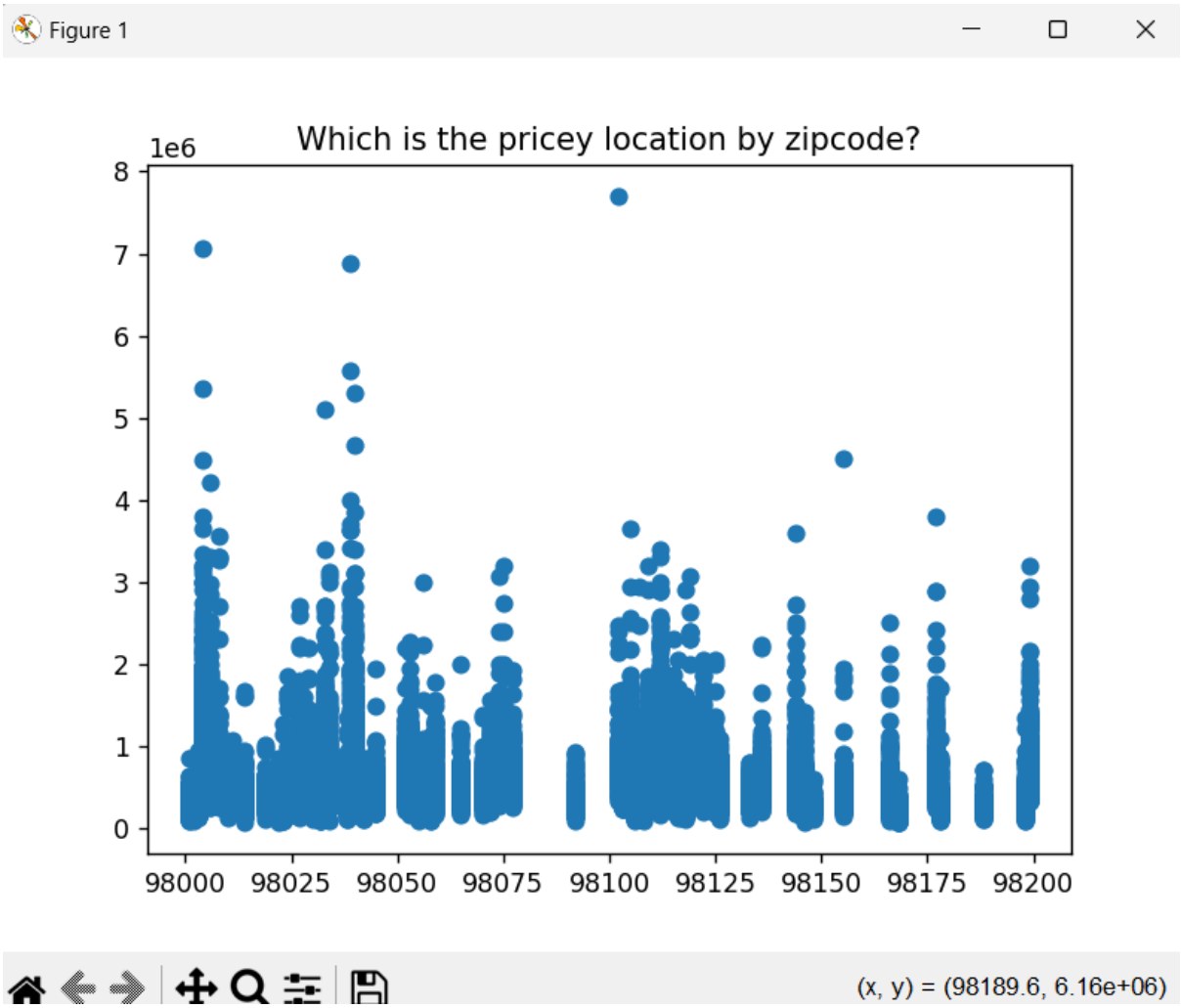


Figure 1
1e6
8
7
6
5
4
3
2
1
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
(x, y) = (3.561, 2.83e+06)



Troubleshooting

Common Issues

- **ModuleNotFoundError**: Ensure all required packages are installed using `pip install numpy pandas matplotlib seaborn scikit-learn`.
- **FileNotFoundError**: Ensure the `kc_house_data.csv` file is in the correct directory.
- **TypeError**: Ensure all function calls and parameters are correctly specified.

Additional Tips

- Refer to the tutorial for a more detailed walkthrough and explanation of the steps.
- Visualize the results and the data to understand the relationships and improve model performance.

By following these steps and tips, you should be able to successfully run the house price prediction project and interpret the results.