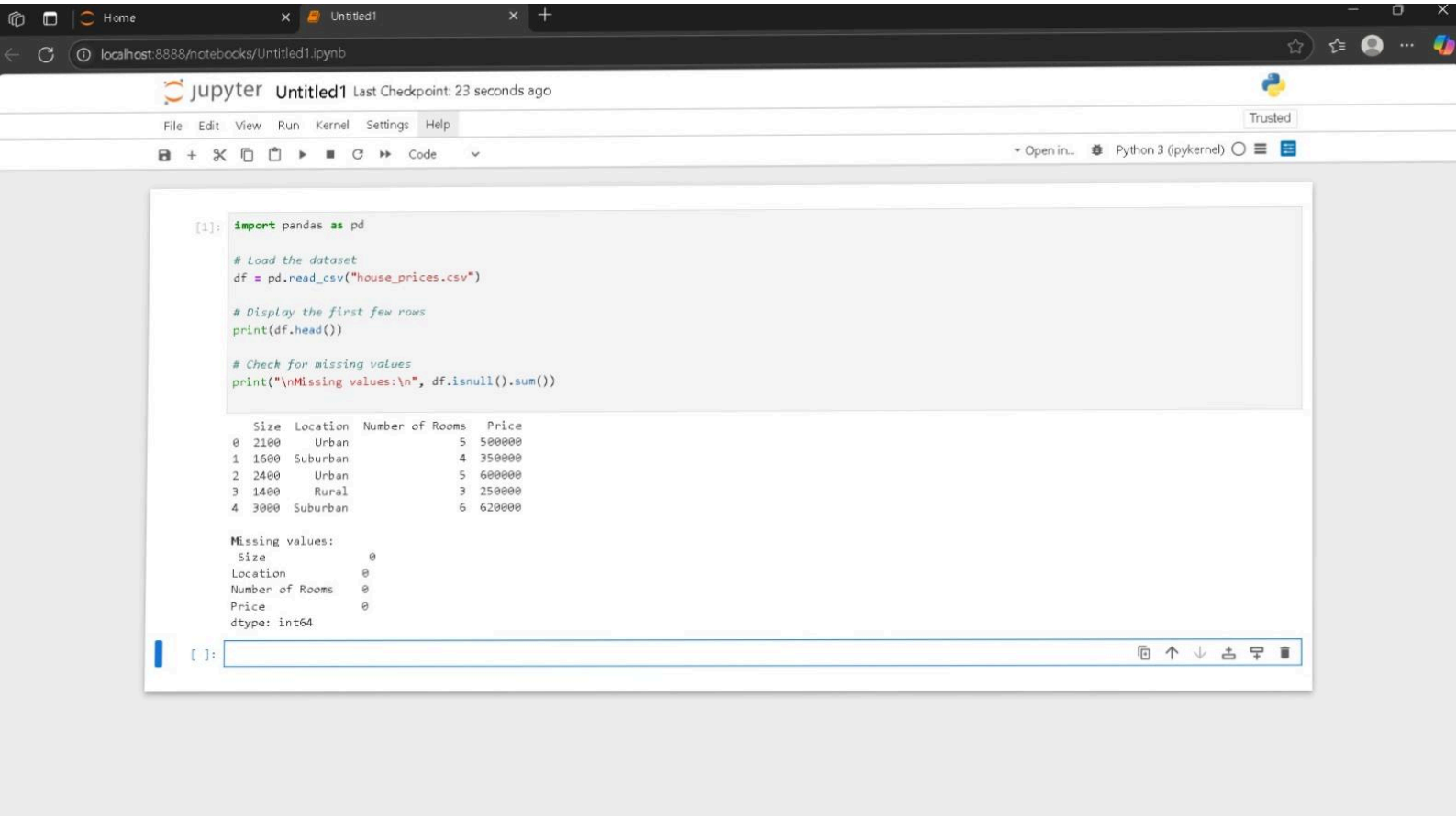

Home
Untitled1
localhost:8888/notebooks/Untitled1.ipynb
jupyter Untitled1 Last Checkpoint: 23 seconds ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)
[1]:
import pandas as pd
# Load the dataset
df = pd.read_csv("house_prices.csv")
# Display the first few rows
print(df.head())
# Check for missing values
print("\nMissing values:\n", df.isnull().sum())
Size Location Number of Rooms Price
0 2100 Urban 5 500000
1 1600 Suburban 4 350000
2 2400 Urban 5 600000
3 1400 Rural 3 250000
4 3000 Suburban 6 620000
Missing values:
Size 0
Location 0
Number of Rooms 0
Price 0
dtype: int64
[ ]:

Home | Untitled1

localhost:8888/notebooks/Untitled1.ipynb

jupyter **Untitled1** Last Checkpoint: 2 minutes ago

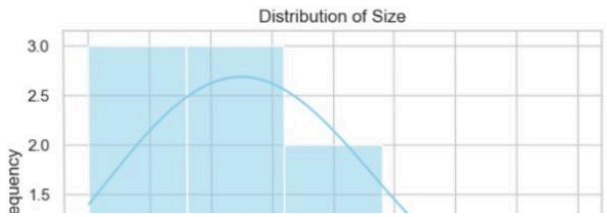
File Edit View Run Kernel Settings Help — Trusted

Code — Open in... — Python 3 (ipykernel)

```
Location          0
Number of Rooms   0
Price             0
dtype: int64
```

[2]:

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

# Set style
sns.set(style="whitegrid")

# Plot histograms for numerical features
numerical_cols = ["Size", "Number of Rooms", "Price"]

for col in numerical_cols:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], kde=True, color='skyblue')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.tight_layout()
    plt.show()

# Display basic statistics
print(df[numerical_cols].describe())
```



Home | Untitled1

localhost:8888/notebooks/Untitled1.ipynb

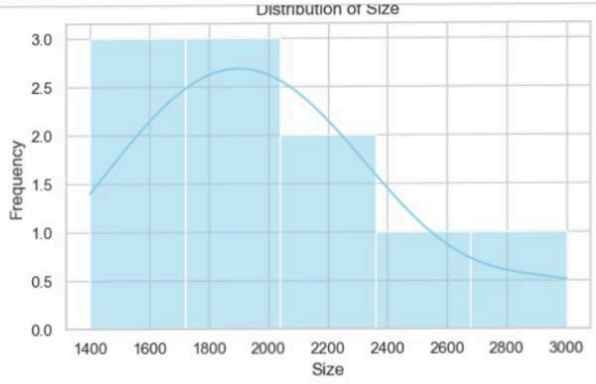
jupyter Untitled1 Last Checkpoint: 2 minutes ago

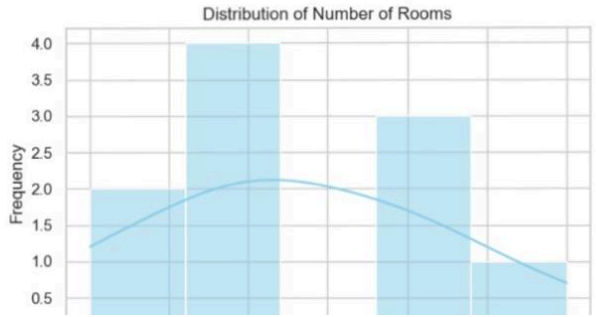
File Edit View Run Kernel Settings Help

Code

Trusted

Open in... Python 3 (ipykernel)





localhost:8888/notebooks/Untitled1.ipynb

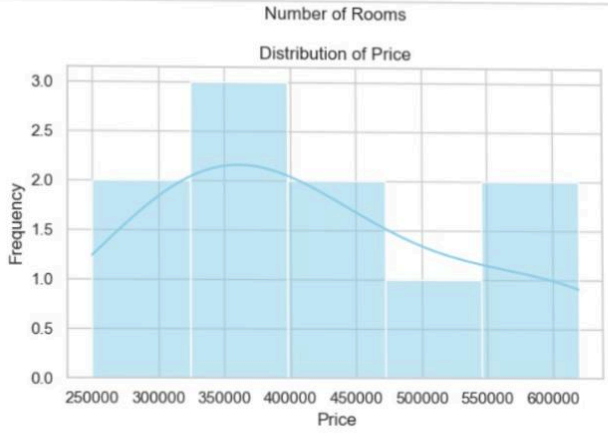
jupyter Untitled1 Last Checkpoint: 2 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

Open in... Python 3 (ipykernel)



	Size	Number of Rooms	Price
count	10.000000	10.000000	10.00000
mean	2010.000000	4.300000	418000.00000
std	455.704583	0.948683	123180.08497
min	1400.000000	3.000000	250000.00000
25%	1725.000000	4.000000	335000.00000
50%	1950.000000	4.000000	385000.00000
75%	2175.000000	5.000000	487500.00000
max	3000.000000	6.000000	620000.00000

[ ]:

Home | Untitled1

localhost:8888/notebooks/Untitled1.ipynb

jupyter Untitled1 Last Checkpoint: 5 minutes ago

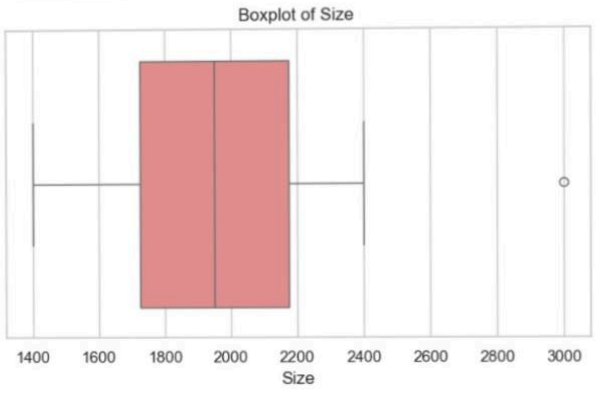
Trusted

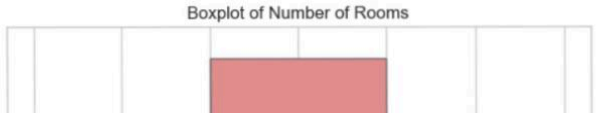
File Edit View Run Kernel Settings Help

Code | Open in... | Python 3 (ipykernel)

```
max    3000.000000          6.000000  620000.00000
```

[3]:
```
# Plot boxplots to identify outliers
for col in ["Size", "Number of Rooms", "Price"]:
    plt.figure(figsize=(6, 4))
    sns.boxplot(data=df, x=col, color='lightcoral')
    plt.title(f'Boxplot of {col}')
    plt.tight_layout()
    plt.show()
```





localhost:8888/notebooks/Untitled1.ipynb

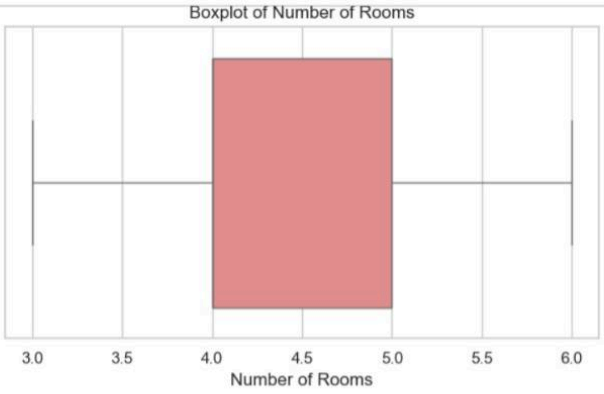
jupyter Untitled1 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

Open in... Python 3 (ipykernel)



Home | Untitled1

localhost:8888/notebooks/Untitled1.ipynb

jupyter Untitled1 Last Checkpoint: 6 minutes ago

File Edit View Run Kernel Settings Help

Trusted

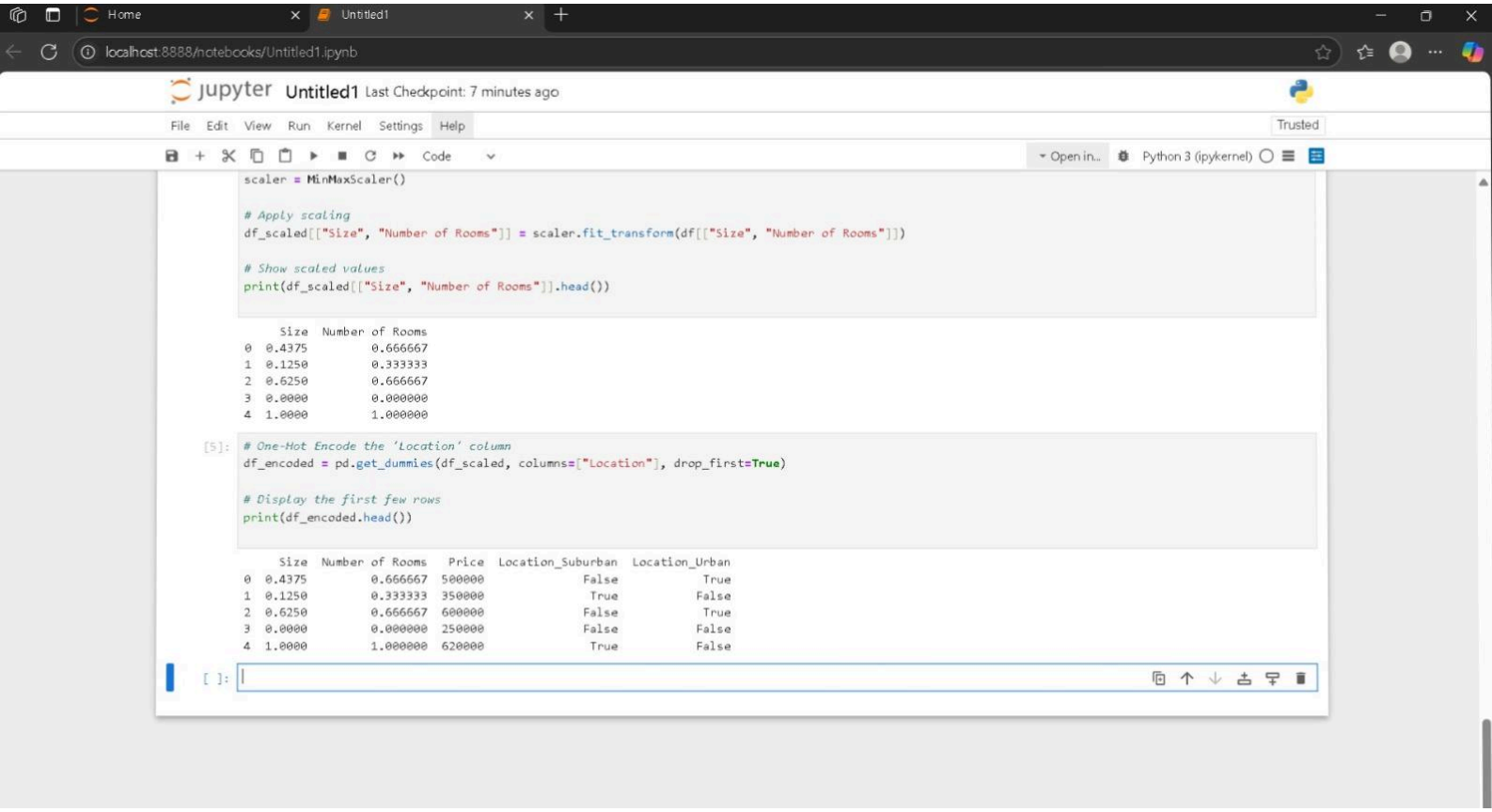
Code

Open in... Python 3 (ipykernel)

[4]:
```
from sklearn.preprocessing import MinMaxScaler

# Create a copy to preserve original data
df_scaled = df.copy()

# Initialize scaler
scaler = MinMaxScaler()

# Apply scaling
df_scaled[["Size", "Number of Rooms"]] = scaler.fit_transform(df[["Size", "Number of Rooms"]])

# Show scaled values
print(df_scaled[["Size", "Number of Rooms"]].head())
```

```
     Size  Number of Rooms
0  0.4375         0.666667
1  0.1250         0.333333
2  0.6250         0.666667
3  0.0000         0.000000
4  1.0000         1.000000
```

[ ]:


Home
Untitled1
localhost:8888/notebooks/Untitled1.ipynb
jupyter Untitled1 Last Checkpoint: 7 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)
scaler = MinMaxScaler()
# Apply scaling
df_scaled[["Size", "Number of Rooms"]] = scaler.fit_transform(df[["Size", "Number of Rooms"]])
# Show scaled values
print(df_scaled[["Size", "Number of Rooms"]].head())
Size Number of Rooms
0 0.4375 0.666667
1 0.1250 0.333333
2 0.6250 0.666667
3 0.0000 0.000000
4 1.0000 1.000000
[5]:
# One-Hot Encode the 'Location' column
df_encoded = pd.get_dummies(df_scaled, columns=["Location"], drop_first=True)
# Display the first few rows
print(df_encoded.head())
Size Number of Rooms Price Location_Suburban Location_Urban
0 0.4375 0.666667 500000 False True
1 0.1250 0.333333 350000 True False
2 0.6250 0.666667 600000 False True
3 0.0000 0.000000 250000 False False
4 1.0000 1.000000 620000 True False
[ ]:

Home | Untitled1

localhost:8888/notebooks/Untitled1.ipynb

jupyter Untitled1 Last Checkpoint: 8 minutes ago

File Edit View Run Kernel Settings Help

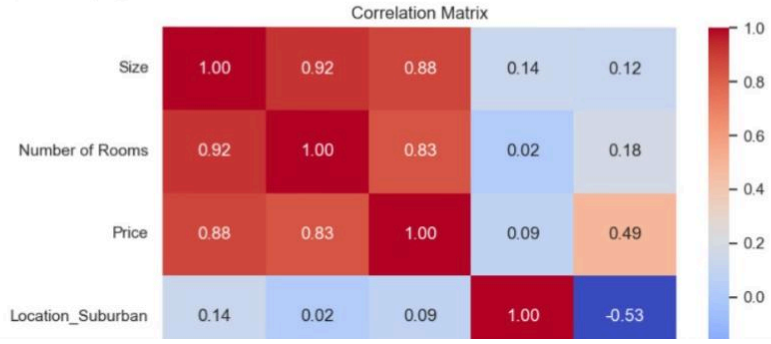
Trusted

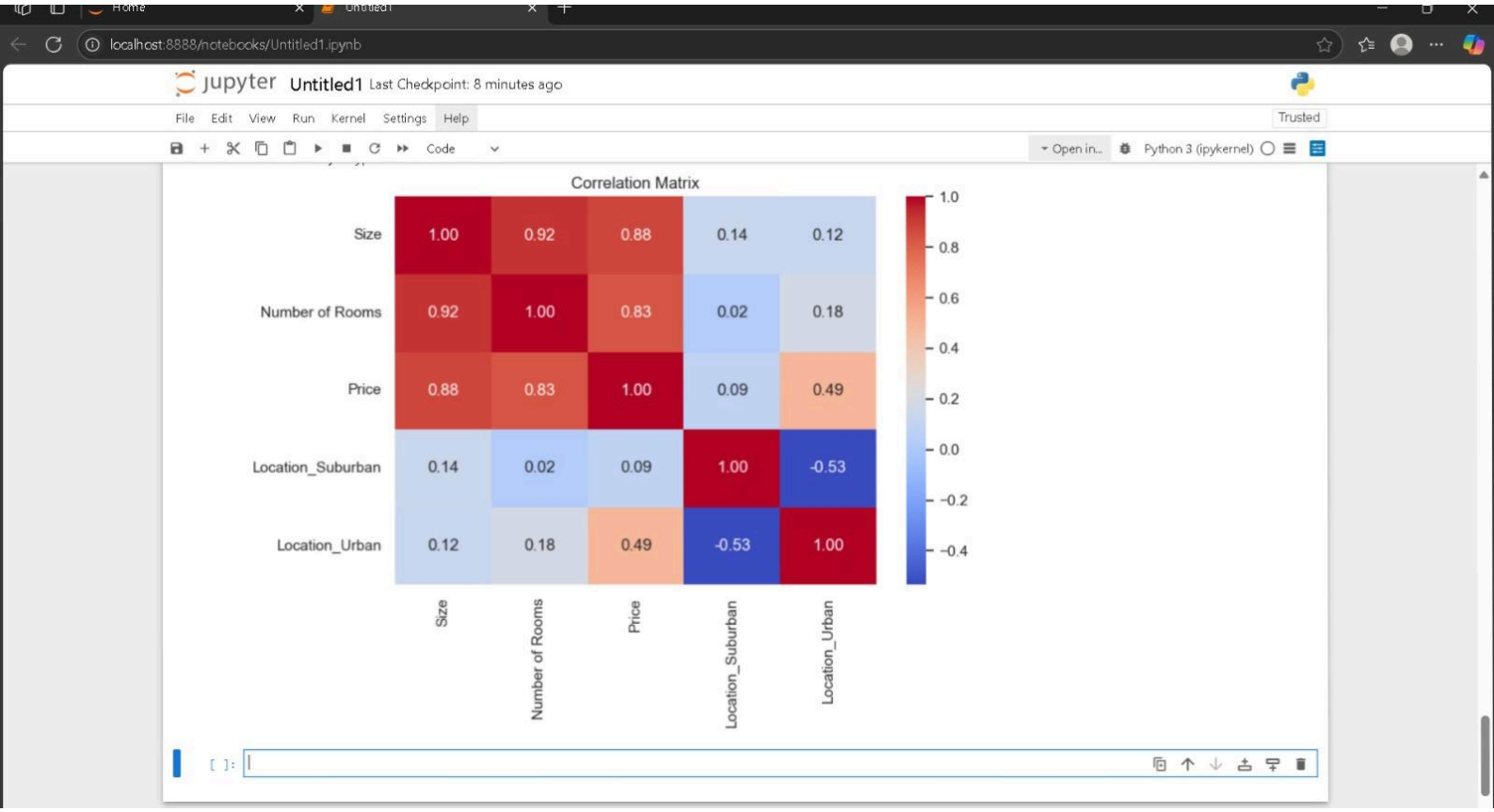
Code

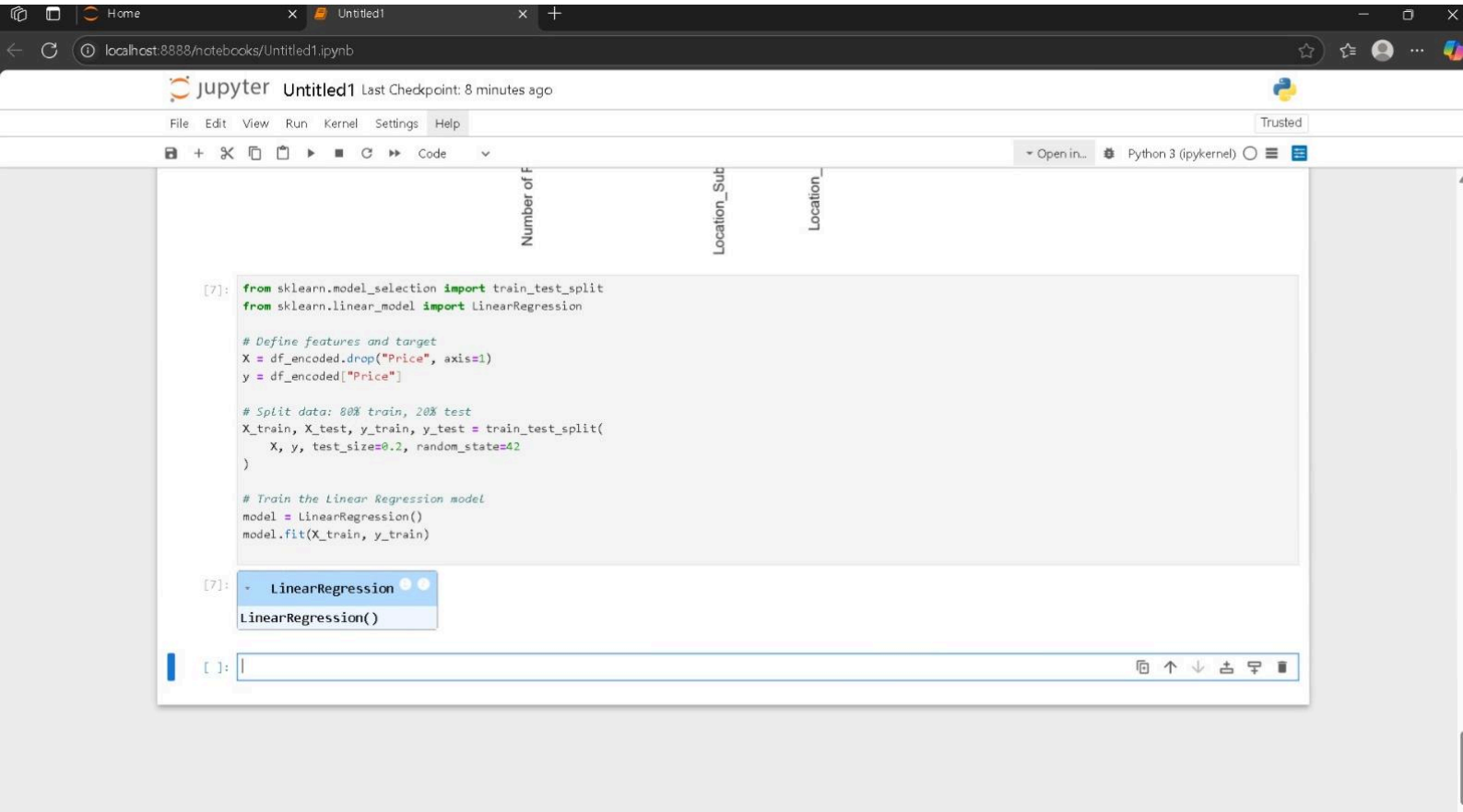
Open in... Python 3 (ipykernel)

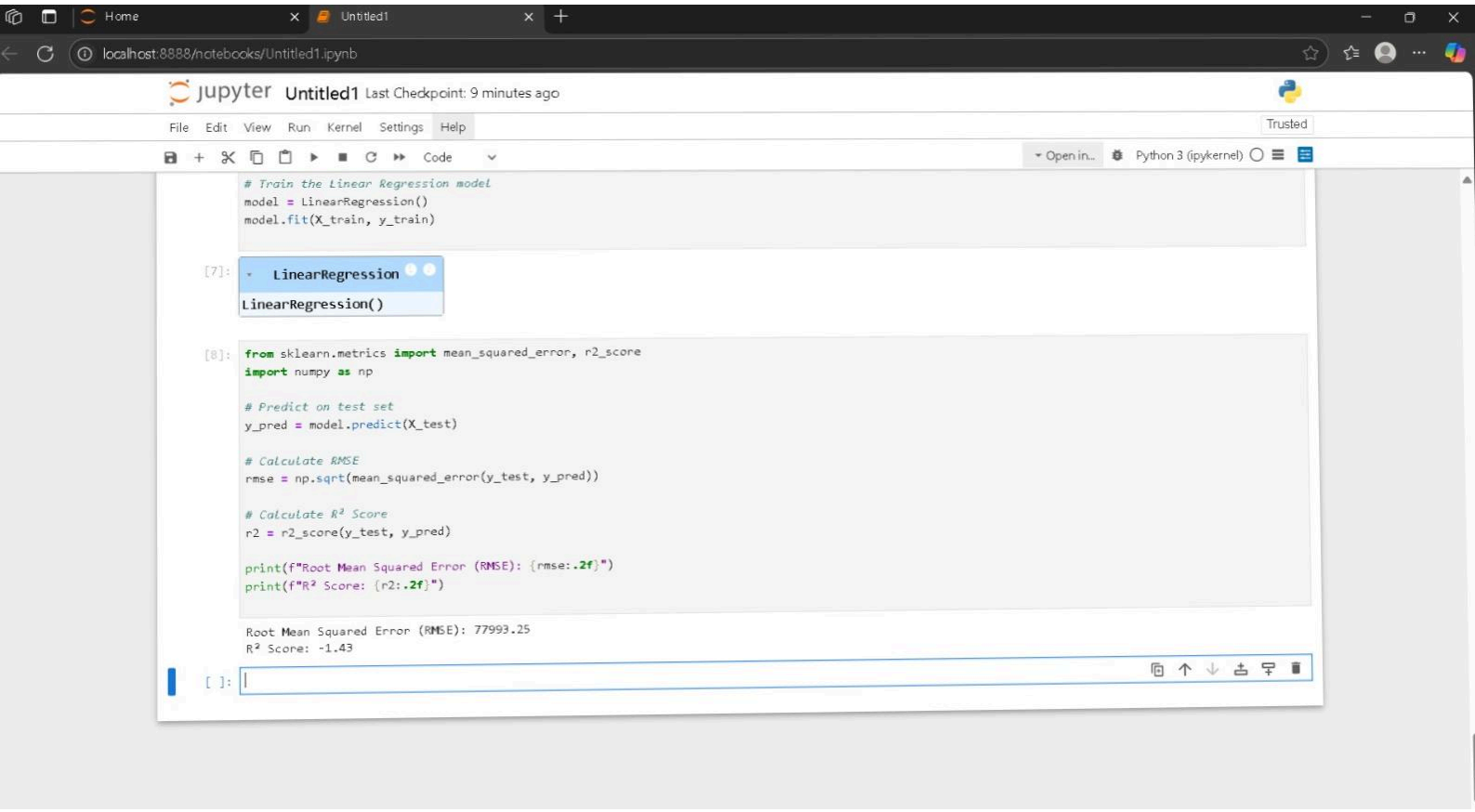
[6]:
```
# Compute correlation matrix
correlation_matrix = df_encoded.corr()

# Display correlation with respect to Price
print(correlation_matrix["Price"].sort_values(ascending=False))

# Plot heatmap for visual insight
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix")
plt.tight_layout()
plt.show()
```

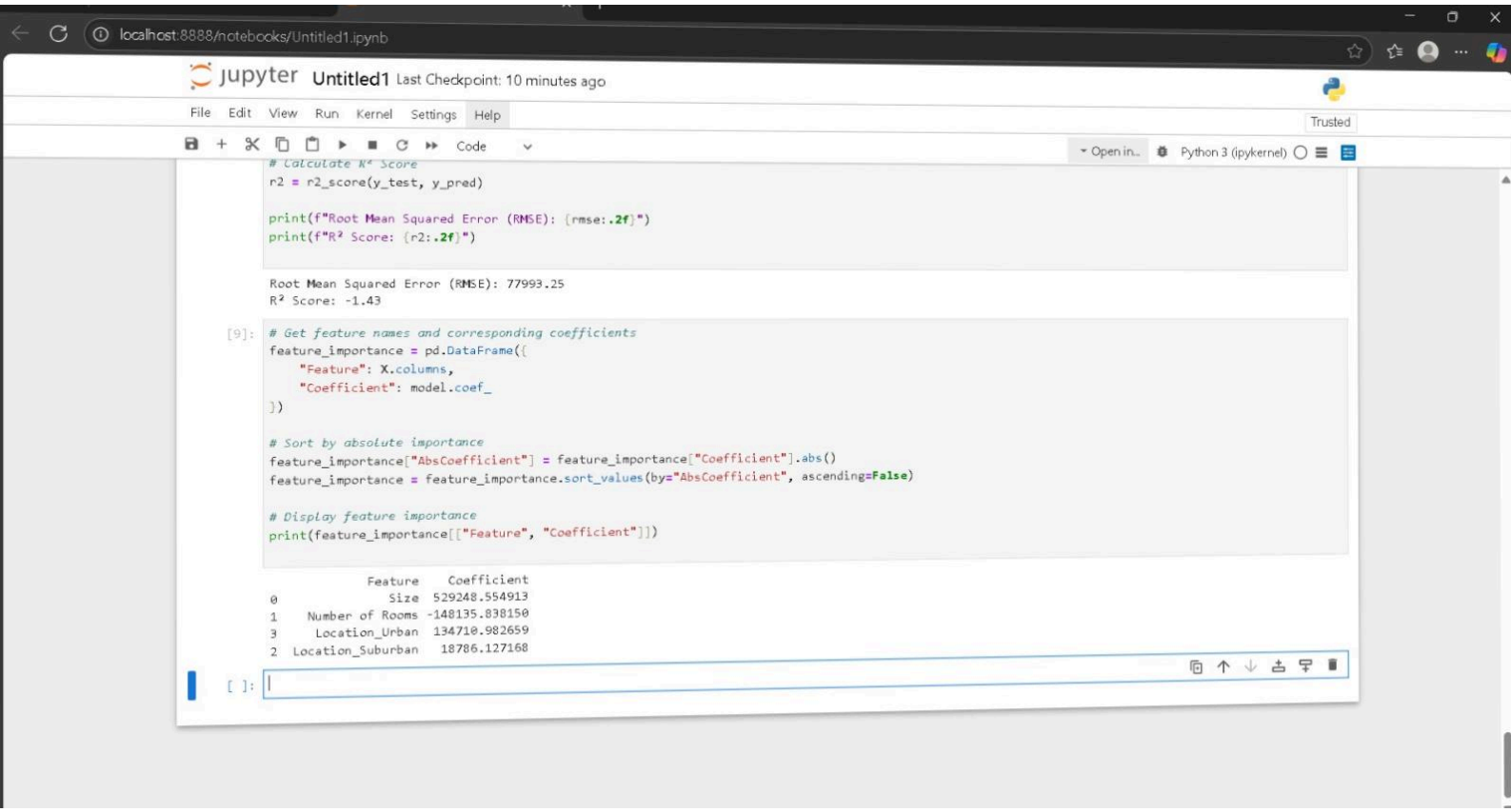
```
Price                1.000000
Size                 0.875291
Number of Rooms      0.832913
Location_Urban       0.485599
Location_Suburban    0.085899
Name: Price, dtype: float64
```




localhost:8888/notebooks/Untitled1.ipynb
jupyter Untitled1 Last Checkpoint: 8 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)
Correlation Matrix
Size 1.00 0.92 0.88 0.14 0.12
Number of Rooms 0.92 1.00 0.83 0.02 0.18
Price 0.88 0.83 1.00 0.09 0.49
Location_Suburban 0.14 0.02 0.09 1.00 -0.53
Location_Urban 0.12 0.18 0.49 -0.53 1.00
Size
Number of Rooms
Price
Location_Suburban
Location_Urban
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
[ ]:


Home
Untitled1
localhost:8888/notebooks/Untitled1.ipynb
jupyter Untitled1 Last Checkpoint: 8 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)
Number of F
Location_Sub
Location_
[7]:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# Define features and target
X = df_encoded.drop("Price", axis=1)
y = df_encoded["Price"]
# Split data: 80% train, 20% test
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.2, random_state=42
)
# Train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
[7]:
LinearRegression
LinearRegression()
[ ]:


Home
Untitled1
localhost:8888/notebooks/Untitled1.ipynb
jupyter Untitled1 Last Checkpoint: 9 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)
# Train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
[7]:
LinearRegression
LinearRegression()
[8]:
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
# Predict on test set
y_pred = model.predict(X_test)
# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
# Calculate R² Score
r2 = r2_score(y_test, y_pred)
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R² Score: {r2:.2f}")
Root Mean Squared Error (RMSE): 77993.25
R² Score: -1.43
[ ]:


localhost:8888/notebooks/Untitled1.ipynb
jupyter Untitled1 Last Checkpoint: 10 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)
# Calculate R² Score
r2 = r2_score(y_test, y_pred)
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R² Score: {r2:.2f}")
Root Mean Squared Error (RMSE): 77993.25
R² Score: -1.43
[9]:
# Get feature names and corresponding coefficients
feature_importance = pd.DataFrame({
"Feature": X.columns,
"Coefficient": model.coef_
})
Sort by absolute importance
feature_importance["AbsCoefficient"] = feature_importance["Coefficient"].abs()
feature_importance = feature_importance.sort_values(by="AbsCoefficient", ascending=False)
# Display feature importance
print(feature_importance[["Feature", "Coefficient"]])
Feature Coefficient
0 Size 529248.554913
1 Number of Rooms -148135.838150
3 Location_Urban 134710.982659
2 Location_Suburban 18786.127168
[ ]: