Python Program on Prediction

October 24, 2024

[11]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
```

[16]:
```
# Load the housing dataset
df = pd.read_csv('USA_HOUSING_DATA.csv')
df
```

[16]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.45857	5.682861	7.009188	4.09	23086.80050	1.059034e+06
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1.505891e+06
2	61287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05
…	…	…	…	…	…	…
4995	60567.94414	7.830362	6.137356	3.46	22837.36103	1.060194e+06
4996	78491.27543	6.999135	6.576763	4.02	25616.11549	1.482618e+06
4997	63390.68689	7.250591	4.805081	2.13	33266.14549	1.030730e+06
4998	68001.33124	5.534388	7.130144	5.44	42625.62016	1.198657e+06
4999	65510.58180	5.992305	6.792336	4.07	46501.28380	1.298950e+06

```
                                                Address
0     208 Michael Ferry Apt. 674\nLaurabury, NE 3701…
1     188 Johnson Views Suite 079\nLake Kathleen, CA…
2     9127 Elizabeth Stravenue\nDanieltown, WI 06482…
3                           USS Barnett\nFPO AP 44820
4                          USNS Raymond\nFPO AE 09386
…                                                 …
4995                 USNS Williams\nFPO AP 30153-7653
4996            PSC 9258, Box 8489\nAPO AA 42991-3352
4997  4215 Tracy Garden Suite 076\nJoshualand, VA 01…
4998                          USS Wallace\nFPO AE 73316
4999  37778 George Ridges Apt. 509\nEast Holly, NV 2…

[5000 rows x 7 columns]
```

[17]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

[18]:
```
print("Column names:")
print(df.columns.tolist())
```

```
Column names:
['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg.
Area Number of Bedrooms', 'Area Population', 'Price', 'Address']
```

[19]:
```
# Display the first few rows of the dataset
print("Dataset head:\n", df.head())
```

```
Dataset head:
    Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
0       79545.45857             5.682861                   7.009188
1       79248.64245             6.002900                   6.730821
2       61287.06718             5.865890                   8.512727
3       63345.24005             7.188236                   5.586729
4       59982.19723             5.040555                   7.839388
```

```
   Avg. Area Number of Bedrooms  Area Population         Price  \
0                          4.09      23086.80050  1.059034e+06
1                          3.09      40173.07217  1.505891e+06
2                          5.13      36882.15940  1.058988e+06
3                          3.26      34310.24283  1.260617e+06
4                          4.23      26354.10947  6.309435e+05

                                          Address
0  208 Michael Ferry Apt. 674\nLaurabury, NE 3701…
1  188 Johnson Views Suite 079\nLake Kathleen, CA…
2  9127 Elizabeth Stravenue\nDanieltown, WI 06482…
3                          USS Barnett\nFPO AP 44820
4                         USNS Raymond\nFPO AE 09386
```

[20]:
```
# Check for missing values
print("\nMissing values in each column:\n", df.isnull().sum())
```

```
Missing values in each column:
 Avg. Area Income                0
Avg. Area House Age              0
Avg. Area Number of Rooms        0
Avg. Area Number of Bedrooms     0
Area Population                  0
Price                            0
Address                          0
dtype: int64
```

[21]:
```
df = df.drop(columns=['Address'])
```

[22]:
```
df.head(5)
```

[22]:
```
   Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
0       79545.45857             5.682861                   7.009188
1       79248.64245             6.002900                   6.730821
2       61287.06718             5.865890                   8.512727
3       63345.24005             7.188236                   5.586729
4       59982.19723             5.040555                   7.839388

   Avg. Area Number of Bedrooms  Area Population         Price
0                          4.09      23086.80050  1.059034e+06
1                          3.09      40173.07217  1.505891e+06
2                          5.13      36882.15940  1.058988e+06
3                          3.26      34310.24283  1.260617e+06
4                          4.23      26354.10947  6.309435e+05
```

[23]:
```
# Handle missing values if any (e.g., fill with mean)
df.fillna(df.mean(), inplace=True)
```

[24]:
```
# Scaling numeric features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.drop(columns=['Price']))
df_scaled = pd.DataFrame(scaled_features, columns=df.columns[:-1])
```

[25]:
```
df_scaled
```

[25]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	\
0	1.028660	-0.296927	0.021274	
1	1.000808	0.025902	-0.255506	
2	-0.684629	-0.112303	1.516243	
3	-0.491499	1.221572	-1.393077	
4	-0.807073	-0.944834	0.846742	
…	…	…	…	
4995	-0.752109	1.869297	-0.845588	
4996	0.929740	1.030822	-0.408686	
4997	-0.487235	1.284470	-2.170269	
4998	-0.054592	-0.446694	0.141541	
4999	-0.288313	0.015215	-0.194342	

	Avg. Area Number of Bedrooms	Area Population
0	0.088062	-1.317599
1	-0.722301	0.403999
2	0.930840	0.072410
3	-0.584540	-0.186734
4	0.201513	-0.988387
…	…	…
4995	-0.422467	-1.342732
4996	0.031337	-1.062747
4997	-1.500251	-0.291937
4998	1.182053	0.651116
4999	0.071855	1.041625

[5000 rows x 5 columns]

[26]:
```
# Adding the target variable back
df_scaled['Price'] = df['Price']
```

[27]:
```
X = df_scaled.drop(columns=['Price'])
y = df_scaled['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[28]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

[28]:
```
LinearRegression()
```

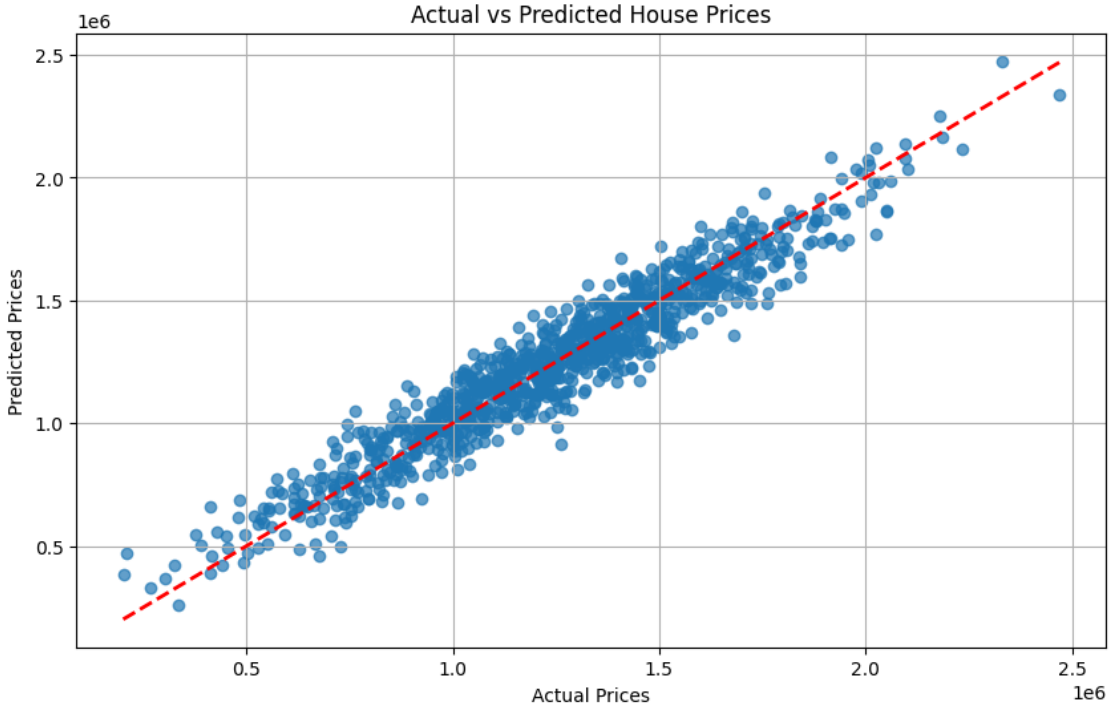
[29]:
```
y_pred = model.predict(X_test)
```

[30]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"\nMean Squared Error: {mse:.2f}")
print(f"R^2 Score: {r2:.2f}")
```

```
Mean Squared Error: 10089009299.50
R^2 Score: 0.92
```

[31]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
↪lw=2)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted House Prices')
plt.grid()
plt.show()
```



[ ]:

[ ]:

[ ]:

[ ]: