House Price Prediction using Linear Regression

Project Overview

This project implements a linear regression model to predict house prices based on various features of the properties. The dataset used for this project contains information about different houses, including attributes such as the number of bedrooms, bathrooms, square footage, and more. The goal is to develop a predictive model that can estimate house prices accurately.

1. Import Necessary Libraries

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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score , mean_squared_error, r2_score
```

2. Load and Explore the Dataset

Let's assume you are using a dataset with various house attributes and their corresponding prices.

```
data = pd.read_csv('/content/drive/MyDrive/Datasets/data.csv')
data
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	1340	0
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	3370	280
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	1930	0
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	1140	800
...	...	...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	1510	0
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	1460	0
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	3010	0
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	1070	1020
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	1490	0

4600 rows × 18 columns

Next steps: Generate code with `data` View recommended plots New interactive sheet

3. Data Preprocessing

Clean the dataset (handle missing values, remove outliers, etc.). For instance, if any columns have missing values, you can either fill them or drop those rows.

```
data.isna().sum()
```

	0
date	0
price	0
bedrooms	0
bathrooms	0
sqft_living	0
sqft_lot	0
floors	0
waterfront	0
view	0
condition	0
sqft_above	0
sqft_basement	0
yr_built	0
yr_renovated	0
street	0
city	0
statezip	0
country	0

dtype: int64

data.types - To check the datatypes of each field

```
data.dtypes
```

	0
date	object
price	float64
bedrooms	float64
bathrooms	float64
sqft_living	int64
sqft_lot	int64
floors	float64
waterfront	int64
view	int64
condition	int64
sqft_above	int64
sqft_basement	int64
yr_built	int64
yr_renovated	int64
street	object
city	object
statezip	object
country	object

dtype: object

data.info() - To check the information of the data

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           4600 non-null   object 
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64  
 5   sqft_lot       4600 non-null   int64  
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64  
 8   view           4600 non-null   int64  
 9   condition      4600 non-null   int64  
 10  sqft_above     4600 non-null   int64  
 11  sqft_basement  4600 non-null   int64  
 12  yr_built       4600 non-null   int64  
 13  yr_renovated   4600 non-null   int64  
 14  street         4600 non-null   object 
 15  city           4600 non-null   object 
 16  statezip       4600 non-null   object 
 17  country        4600 non-null   object 
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

```
from sklearn import preprocessing
labelencoder = preprocessing.LabelEncoder()

for i in data:
  data[i] = labelencoder.fit_transform(data[i])
```

```
data.head()
```

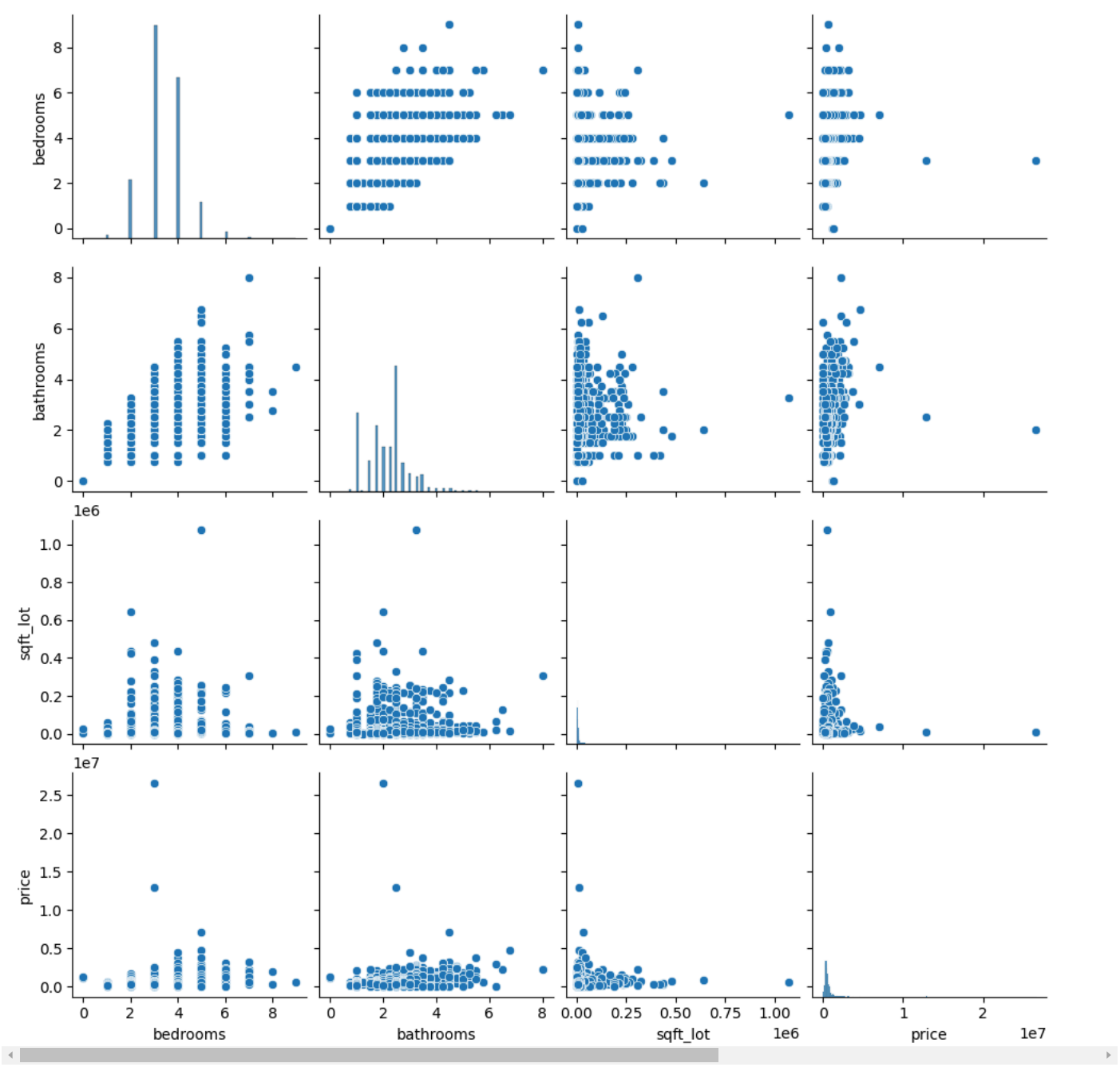
	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_
0	0	402	3	4	93	1399	1	0	0	2	96	0	55	
1	0	1719	5	8	406	1701	2	0	4	4	373	29	21	
2	0	487	3	6	180	2196	0	0	0	3	182	0	66	
3	0	706	3	7	191	1439	0	0	0	3	52	109	63	
4	0	998	4	8	181	2026	0	0	0	3	70	85	76	

Next steps: Generate code with data | View recommended plots | New interactive sheet

4. Visualizing the Data

In this step, we visualize the data to get an initial understanding of how the features are distributed

```
df = data[['bedrooms', 'bathrooms', 'sqft_lot', 'price']]
sns.pairplot(df)
plt.show()
```



5. Feature Selection

Choose relevant features (independent variables) to predict house prices (target variable).

```
x = data.drop('price',axis=1)
y = data['price']
```

6. Train-Test Split

Split the data into training and testing sets to evaluate the model's performance.

```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 1, test_size =.2)
```

```
print(f'x_train shape: {x_train.shape}')
print(f'x_test shape: {x_test.shape}')
print(f'y_train shape: {y_train.shape}')
print(f'y_test shape: {y_test.shape}')
```

```
x_train shape: (3680, 17)
x_test shape: (920, 17)
y_train shape: (3680,)
y_test shape: (920,)
```

7. Train the Linear Regression Model

Fit the model on the training data.

```
model = LinearRegression()
model.fit(x_train,y_train)
```

```
LinearRegression
LinearRegression()
```

8. Make Predictions

Predict house prices on the test data.

```
y_pred = model.predict(x_test)
```

```
y_pred
```

```
array([ 753.27887397,  991.16170748,  599.01246364, 1084.10393328,
       1236.59402921, 1019.05182296,  359.80897402,  546.27602373,
        621.550604  ,  760.15649119,  633.43070487,  418.03375666,
        466.62670751, 1100.42501572,  463.55648748,  765.89396274,
        955.65662854,  744.13848783,  791.85040629,  894.77559092,
        597.48640136,  929.63710065,  964.28856883,  210.77461542,
        879.79590908,  884.63507569,  406.08414348,  445.78196433,
        976.05789925,  700.86877556,  636.88799366, 1337.21170882,
        403.31594293,  687.59219902,  783.43204974,  712.81375689,
        149.30947512,  666.81087866, 1096.62060826,  272.24540901,
       1241.35377374,  788.30923487, 1261.27012744,  824.79301225,
        730.87322062,  856.09027145,  779.68723574, 1037.52547827,
        748.55351445, 1368.04301638, 1465.39377721,  775.4323382 ,
        684.92017217,  412.07954786,  784.8636073 , 1157.595603  ,
        644.73896165,  434.17124047,  538.4587308 ,  999.46145114,
        483.78544566,  649.13297511,  580.24602017, 1406.1683971 ,
        608.85105437,  556.78996983,  660.71986133,  647.19379335,
        655.50046899, 1054.04189154,  678.92552451,  444.73640141,
       1068.28323917,  428.92587795,  892.13430402,  666.61042938,
        391.1112235 , 1079.46424539,  720.26039276,  787.77069089,
        485.88264232,  564.29950102,  373.57054803, 1029.87045006,
        902.13277828, 1055.39471575, 1189.15759942,  316.23314575,
        593.29917717, 1027.58684172,  609.28648808,  358.54633412,
        286.51723093, 1360.85153961, 1076.08233436,  328.04365219,
        478.79353046,  731.81825299,  679.28321414,  700.16854245,
        646.27144237,  838.24248742,  461.20785026,  485.98614672,
        724.91737869,  650.49492971, 1673.55836147,  971.86191775,
        637.97940148, 1163.57959968, 1487.66706186,  630.67997596,
       1140.90092256,  697.34014498,  907.98014884,  939.20909538,
        346.62430377, 1063.11923712,  894.0657741 , 1481.07234366,
        902.54077479,  707.65348064,  481.14532866,  798.06325456,
       1355.07010265,  477.05679668,  543.90327794,  912.47101013,
        844.39125176,  610.62434375, 1013.68740989,  660.22037867,
        574.62704448,  827.7623344 , 1207.58857914, 1259.87651004,
       1300.98612986,  572.45522166,  447.90357097,  698.61739798,
       1532.18222031,  442.93161408, 1611.88285508,  575.05461403,
        917.89443461,  642.46835562,  740.8802548 , 1077.99775554,
       1045.72952814, 1078.403818  , 1744.29525684,  499.75935262,
        863.16271945,  626.87354401, 1290.24679376,  948.12416557,
        972.69869643, 1073.2311262 , 1142.24793856,  887.72145053,
        636.32485554, 1665.56421545,  552.27489973,  334.42002064,
        668.65645848,  875.96904211, 1160.89318138, 1104.19300377,
        688.50233208, 1332.04020918, 1442.24779326,  752.19749661,
       1035.57078237, 1085.88730057,  579.70991198,  960.94115036,
       1406.53908876, 1134.25813073,  750.59356833,  681.97129481,
```

```
 480.20191474, 1034.90381043,  494.20411856,  504.33574293,
 868.15609616, 1195.02013054, 1302.03967424, 1200.59941716,
 993.0493114 , 1255.7577855 ,  533.91002182,  940.55419318,
 612.89564977,  482.99779023,  695.9204277 ,  666.96093436,
1567.14356605,  407.68117523,  887.57623918,  848.14852841,
 932.85370552, 1323.38184494, 1222.10981717,  967.53365099,
 822.64248296, 1279.52448055,  701.7401474 ,  309.88044088,
 614.76369543,  609.58501912,  457.0212797 ,  958.55519329,
1170.57561451,  139.90352411,  567.72719514,  465.59953369,
1159.86476739, 1208.30250049,  550.50597852, 1052.71376001,
 994.415606  ,  492.81403377, 1165.63679304,  501.48958928,
 802.67275639,  400.31168296,  808.21643847,  731.30103352,
 500.56366931,  923.69189884,  565.88160038,  819.24901895,
```

9. Evaluate the Model

Evaluate the performance of the model using metrics like Mean Squared Error (MSE) and R² score.

```
model.score(x_test,y_test)
```

```
0.4783513285423847
```

```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 111408.94582510261
```

```
r2_score = model.score(x_test, y_test)
print(f'R² Score: {r2_score}')
```

```
R² Score: 0.4783513285423847
```

10. Visualize the Results

You can plot the predicted vs actual house prices to visualize the model's performance.

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5, color='b')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.title('Predicted vs Actual House Prices')
plt.xlabel('Actual House Prices')
plt.ylabel('Predicted House Prices')
plt.show()
```