Assignment 2 : Linear Regression

House Price Prediction

Importing Libraries

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

In [ ]:
```
df = pd.read_csv('./Dataset/Housing.csv')
```

In [ ]:
```
df.head()
```

Out[ ]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating
0	13300000	7420	4	2	3	yes	no	no	no
1	12250000	8960	4	4	4	yes	no	no	no
2	12250000	9960	3	2	2	yes	no	yes	no
3	12215000	7500	4	2	2	yes	no	yes	no
4	11410000	7420	4	1	2	yes	yes	yes	no

EDA

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
price             0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
prefarea          0
furnishingstatus  0
dtype: int64
```

In [ ]:
```
df['furnishingstatus'].unique()
```

Out[ ]:
```
array(['furnished', 'semi-furnished', 'unfurnished'], dtype=object)
```

Label Encoding

In [ ]:
```
label_encoder_obj = LabelEncoder()
```

In [ ]:
```
categorical_columns = ['mainroad','guestroom','basement','hotwaterheating','airconditioning',
```

In [ ]:
```
for column in categorical_columns:
    df[column] = label_encoder_obj.fit_transform(df[column])
```

In [ ]:
```
df.head()
```

Out[ ]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating
0	13300000	7420	4	2	3	1	0	0	0
1	12250000	8960	4	4	4	1	0	0	0
2	12250000	9960	3	2	2	1	0	1	0
3	12215000	7500	4	2	2	1	0	1	0
4	11410000	7420	4	1	2	1	1	1	0

Scaling

In [ ]:
```
std_scaler_obj = StandardScaler()
```

In [ ]:
```
df['area'] = std_scaler_obj.fit_transform(df['area'].values.reshape(-1,1))
df
```

Out[ ]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterh
0	13300000	1.046726	4	2	3	1	0	0	
1	12250000	1.757010	4	4	4	1	0	0	
2	12250000	2.218232	3	2	2	1	0	1	
3	12215000	1.083624	4	2	2	1	0	1	
4	11410000	1.046726	4	1	2	1	1	1	
...	...	...	...	...	...	...	...	...	
540	1820000	-0.991879	2	1	1	1	0	1	
541	1767150	-1.268613	3	1	1	0	0	0	
542	1750000	-0.705921	2	1	1	1	0	0	
543	1750000	-1.033389	3	1	1	0	0	0	
544	1750000	-0.599839	3	1	2	1	0	0	

545 rows × 13 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    float64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    int32
 6   guestroom         545 non-null    int32
 7   basement          545 non-null    int32
 8   hotwaterheating   545 non-null    int32
 9   airconditioning   545 non-null    int32
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    int32
 12  furnishingstatus  545 non-null    int32
dtypes: float64(1), int32(7), int64(5)
memory usage: 40.6 KB
```

In [ ]:
```
X = df.drop('price',axis=1)
y = df['price']
```

In [ ]:
```
X
```

Out[ ]:

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	air
0	1.046726	4	2	3	1	0	0	0	
1	1.757010	4	4	4	1	0	0	0	
2	2.218232	3	2	2	1	0	1	0	
3	1.083624	4	2	2	1	0	1	0	
4	1.046726	4	1	2	1	1	1	0	
...	...	...	...	...	...	...	...	...	
540	-0.991879	2	1	1	1	0	1	0	
541	-1.268613	3	1	1	0	0	0	0	
542	-0.705921	2	1	1	1	0	0	0	
543	-1.033389	3	1	1	0	0	0	0	
544	-0.599839	3	1	2	1	0	0	0	

545 rows × 12 columns

In [ ]:
```
y
```

Out[ ]:
```
0      13300000
1      12250000
2      12250000
3      12215000
4      11410000
         ...
540     1820000
541     1767150
542     1750000
543     1750000
544     1750000
Name: price, Length: 545, dtype: int64
```

Train-Test Splitting

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
```

In [ ]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(381, 12)
(381,)
(164, 12)
(164,)
```

Linear Regression

In [ ]:
```
lr_model = LinearRegression()
```

In [ ]:
```
lr_model.fit(X_train,y_train)
```

Out[ ]:
```
▾ LinearRegression
LinearRegression()
```

In [ ]:
```
y_pred = lr_model.predict(X_test)
```

In [ ]:
```
r2_score(y_test,y_pred)
```

Out[ ]:
```
0.7235015223200352
```

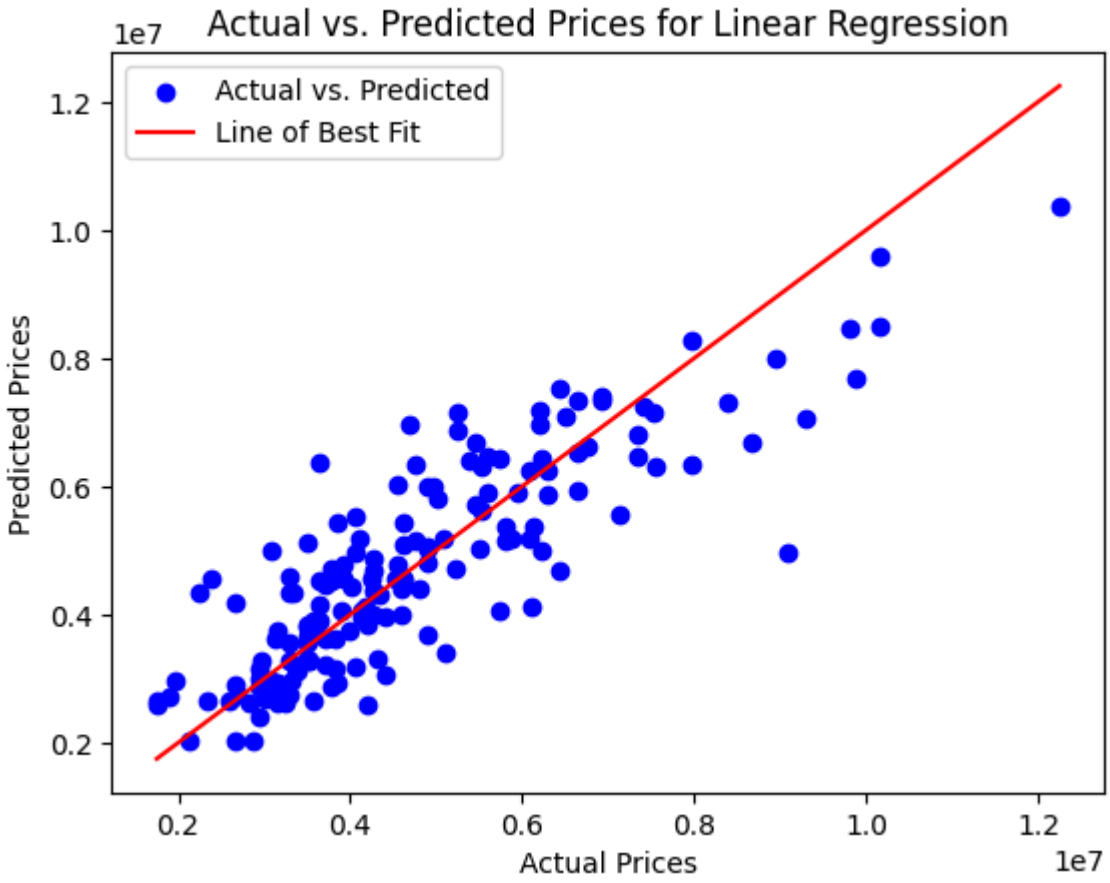
In [ ]:
```
lr_model.intercept_
```

Out[ ]:
```
1388886.23529877
```

In [ ]:
```
lr_model.score(X_test,y_test)
```

Out[ ]:
```
0.7235015223200352
```

In [ ]:
```
plt.scatter(y_test, y_pred, color='blue', label='Actual vs. Predicted')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', label='Line of
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs. Predicted Prices for Linear Regression')
plt.legend()
plt.show()
```



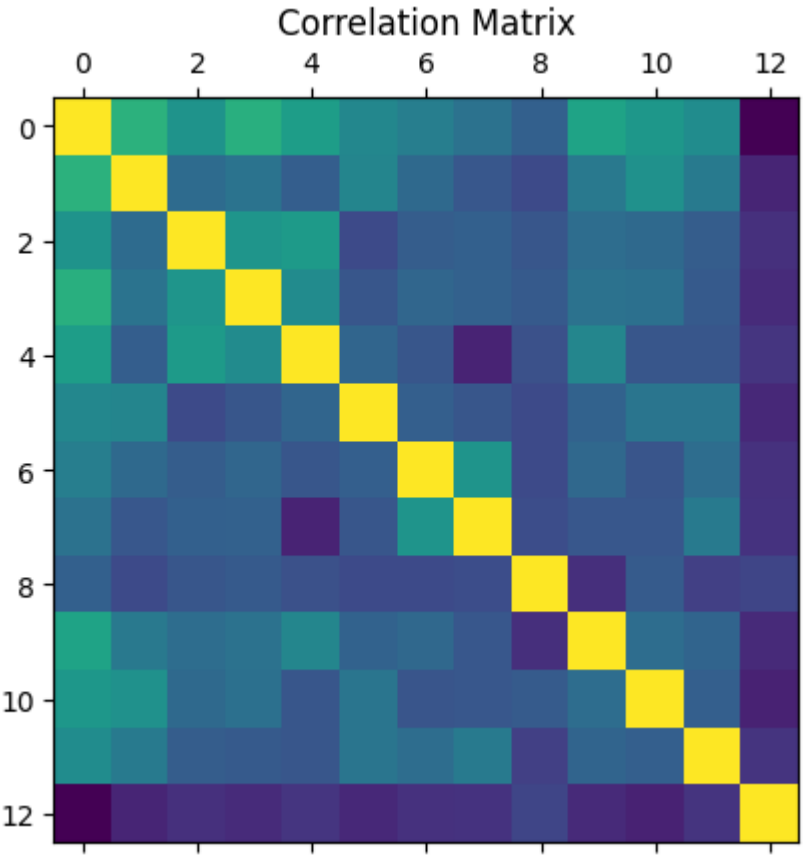
Simple Linear Regression

In [ ]:
```
df.corr()
```

Out[ ]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basem
price	1.000000	0.535997	0.366494	0.517545	0.420712	0.296898	0.255517	0.187
area	0.535997	1.000000	0.151858	0.193820	0.083996	0.288874	0.140297	0.047
bedrooms	0.366494	0.151858	1.000000	0.373930	0.408564	-0.012033	0.080549	0.097
bathrooms	0.517545	0.193820	0.373930	1.000000	0.326165	0.042398	0.126469	0.102
stories	0.420712	0.083996	0.408564	0.326165	1.000000	0.121706	0.043538	-0.172
mainroad	0.296898	0.288874	-0.012033	0.042398	0.121706	1.000000	0.092337	0.044
guestroom	0.255517	0.140297	0.080549	0.126469	0.043538	0.092337	1.000000	0.372
basement	0.187057	0.047417	0.097312	0.102106	-0.172394	0.044002	0.372066	1.000
hotwaterheating	0.093073	-0.009229	0.046049	0.067159	0.018847	-0.011781	-0.010308	0.004
airconditioning	0.452954	0.222393	0.160603	0.186915	0.293602	0.105423	0.138179	0.047
parking	0.384394	0.352980	0.139270	0.177496	0.045547	0.204433	0.037466	0.051
prefarea	0.329777	0.234779	0.079023	0.063472	0.044425	0.199876	0.160897	0.228
furnishingstatus	-0.304721	-0.171445	-0.123244	-0.143559	-0.104672	-0.156726	-0.118328	-0.112

In [ ]:
```
plt.matshow(df.corr())
plt.title('Correlation Matrix')
plt.show()
```



In [ ]:
```
X = df['area']
y = df['price']
```

In [ ]:
```
X
```

Out[ ]:
```
0      1.046726
1      1.757010
2      2.218232
3      1.083624
4      1.046726
         ...
540   -0.991879
541   -1.268613
542   -0.705921
543   -1.033389
544   -0.599839
Name: area, Length: 545, dtype: float64
```

In [ ]:
```
y
```

Out[ ]:
```
0      13300000
1      12250000
2      12250000
3      12215000
4      11410000
         ...
540     1820000
541     1767150
542     1750000
543     1750000
544     1750000
Name: price, Length: 545, dtype: int64
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
```

In [ ]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(381,)
(381,)
(164,)
(164,)
```

In [ ]:
```
lr_model.fit(X.values.reshape(-1, 1), y)
```

Out[ ]:
```
▾ LinearRegression

LinearRegression()
```

In [ ]:
```
y_pred = lr_model.predict(X_test.values.reshape(-1,1))
```

In [ ]:
```
r2_score(y_test.values.reshape(-1, 1), y_pred)
```

Out[ ]:
```
0.3545242726947234
```
