si-assignment01

February 2, 2024

Assignment 01 Name : Yatharth Thakare PRN no : 12111403 Roll no : AI C Subject : SI

[66]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

[68]:
```
df_housing.head(5)
```

[68]:

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

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	no	yes	2	yes	furnished
1	no	yes	3	no	furnished
2	no	no	2	yes	semi-furnished
3	no	yes	3	yes	furnished
4	no	yes	2	no	furnished

[69]:
```
missing_values = df_housing.isnull().sum()
missing_values
```

[69]:
```
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
```

[70]:
```
df_housing.corr()
```

```
<ipython-input-70-4e998b32f5e8>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df_housing.corr()
```

[70]:

	price	area	bedrooms	bathrooms	stories	parking
price	1.000000	0.535997	0.366494	0.517545	0.420712	0.384394
area	0.535997	1.000000	0.151858	0.193820	0.083996	0.352980
bedrooms	0.366494	0.151858	1.000000	0.373930	0.408564	0.139270
bathrooms	0.517545	0.193820	0.373930	1.000000	0.326165	0.177496
stories	0.420712	0.083996	0.408564	0.326165	1.000000	0.045547
parking	0.384394	0.352980	0.139270	0.177496	0.045547	1.000000

[71]:
```
correlation_matrix = df_housing.corr()
correlation_with_price = correlation_matrix['price'].abs().
 ↪sort_values(ascending=False)
print(correlation_with_price)
```

```
price        1.000000
area         0.535997
bathrooms    0.517545
stories      0.420712
parking      0.384394
bedrooms     0.366494
Name: price, dtype: float64
```

```
<ipython-input-71-4cd243fff811>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = df_housing.corr()
```

[72]:
```
X = np.array(df_housing['area'])
y = np.array(df_housing['price'])
# print(len(y))
```

[74]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

##Implementing Linear Regression from scratch

m = cov(x, y) / var(x) b = mean(y) — m * mean(x)

[75]:
```
X_mean = np.mean(X_train)
y_mean = np.mean(y_train)
print(X_mean,y_mean)
```

```
5128.40157480315 4762406.272965879
```

[76]:
```
var = np.var(X_train)
print(var)
```

```
4728270.187819042
```

[77]:
```
sum = 0
for i in range(len(X_train)):
  sum+= ((X_train[i]-X_mean)*(y_train[i]-y_mean))

covv = sum/(len(X_train)-1)
```

[78]:
```
m = covv/var
print(m)
b = y_mean - (X_mean*m)
print(b)
```

```
450.6297634720927
2451395.884322428
```

Pred values for X_test

[79]:
```
pred = []
for x in X_test:
  y = m*x+b
  pred.append(y)

print(pred)
```

```
[5110111.488807775, 5380489.346891031, 4271940.128749683, 4704544.701682892,
4235889.747671915, 5479627.894854891, 6290761.469104658, 4700038.404048171,
3911436.317972008, 3668096.2456970783, 6317799.254912984, 4224173.37382164,
4127738.604438613, 3848348.1510859155, 4240396.045306636, 4087181.9257261246,
3352655.4112666138, 5155174.465154985, 5087580.000634171, 5155174.465154985,
4713557.296952333, 5526944.0200194605, 4095743.8912320943, 4163788.9855163805,
6169091.432967193, 6939668.328504471, 3843841.8534511942, 3803285.174738706,
8399708.762154052, 3803285.174738706, 4253914.938210798, 3857360.7463553576,
5155174.465154985, 5432311.769690322, 4603153.004901671, 4524292.796294054,
4545021.76541377, 3884398.532163683, 4019587.4612053107, 3707301.0351191508,
6047421.396829728, 4087181.9257261246, 5344438.965813263, 4344040.890905217,
5493146.787759054, 5103352.042355694, 5155174.465154985, 4542317.986832938,
6078965.480272774, 3803285.174738706, 5819853.366276321, 3803285.174738706,
5952789.146500588, 4479229.819946845, 4091688.223360846, 3762728.496026218,
5042517.024286961, 3532907.3166554505, 5357957.858717427, 4163788.9855163805,
```

```
3884398.532163683, 3413490.429335346, 5353451.561082705, 4686519.5111440085,
4866771.416532844, 4145763.7949774964, 5425552.323238241, 3803285.174738706,
6236685.897488007, 5019985.536113357, 3735690.710217892, 5317401.180004938,
6236685.897488007, 7183008.400779401, 3660886.169481525, 4371078.676713543,
4087181.9257261246, 4621178.195440555, 5155174.465154985, 4839733.6307245195,
5317401.180004938, 4289965.319288567, 4929859.583418937, 5457096.406681286,
3217466.482224986, 5137149.2746161, 3713159.2220442877, 4006068.568301148,
5795068.729285356, 5155174.465154985, 3309845.5837367647, 6078965.480272774,
4907328.095245333, 3988043.377762264, 6317799.254912984, 4118726.009169171,
5077666.145837784, 5065048.512460565, 4929859.583418937, 5425552.323238241,
3911436.317972008, 4524292.796294054, 3789766.2818345437, 3938474.103780334,
5155174.465154985, 5574260.145184031, 5155174.465154985, 5155174.465154985,
5200237.441502193, 5155174.465154985, 5695930.181321496, 4170999.0617319336,
3627539.56698459, 4141257.4973427756, 4709050.9993176125, 4186320.473689985,
4738792.563706771, 3884398.532163683, 5164187.060424427, 3857360.7463553576,
5921245.063057542, 5605804.228627076, 4060144.1399177993, 3803285.174738706,
5357957.858717427, 3532907.3166554505, 4839733.6307245195, 5796871.248339244,
7899509.724700029, 5722967.967129821, 3803285.174738706, 4794670.65437731,
6236685.897488007, 5391755.090977833, 4096194.5209955666, 4089435.0745434854,
4253914.938210798, 4524292.796294054, 4992947.7503050305, 4037612.6517441943,
3278752.1300571905, 3870879.63925952, 5010972.940843915, 3417996.7269700672,
6101496.9684463795, 4222370.854767753, 5344438.965813263, 4312496.807462171,
7606600.3784431685, 6807633.807807148, 7119920.233893309, 5137149.2746161,
3749209.603122055, 5317401.180004938, 4019587.4612053107, 6101496.9684463795,
5398514.537429914, 3707301.0351191508, 5371476.751621589, 4073663.0328219617,
3886651.6809810433, 4438673.141234357, 6507063.755571263, 4096194.5209955666]
```

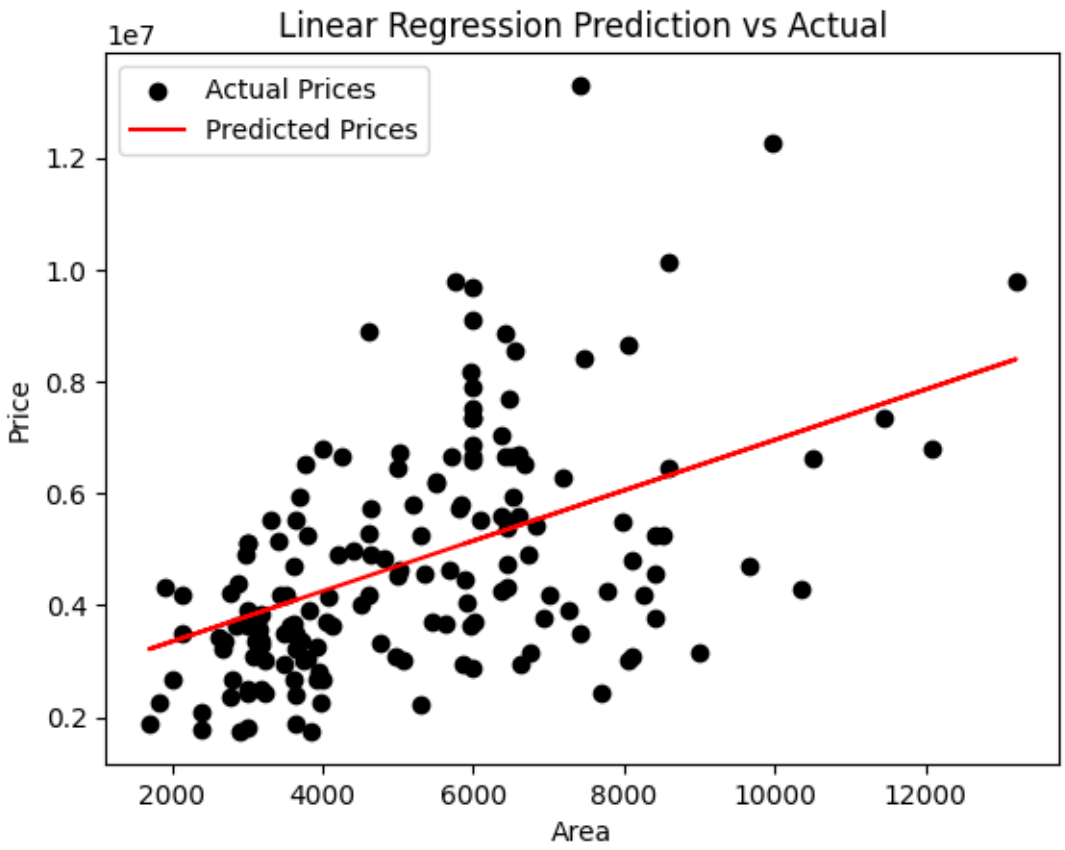
[80]:
```
print(len(X_test),len(pred))
```

```
164 164
```

[81]:
```
pred_y = np.array(pred)
```

0.1 Output

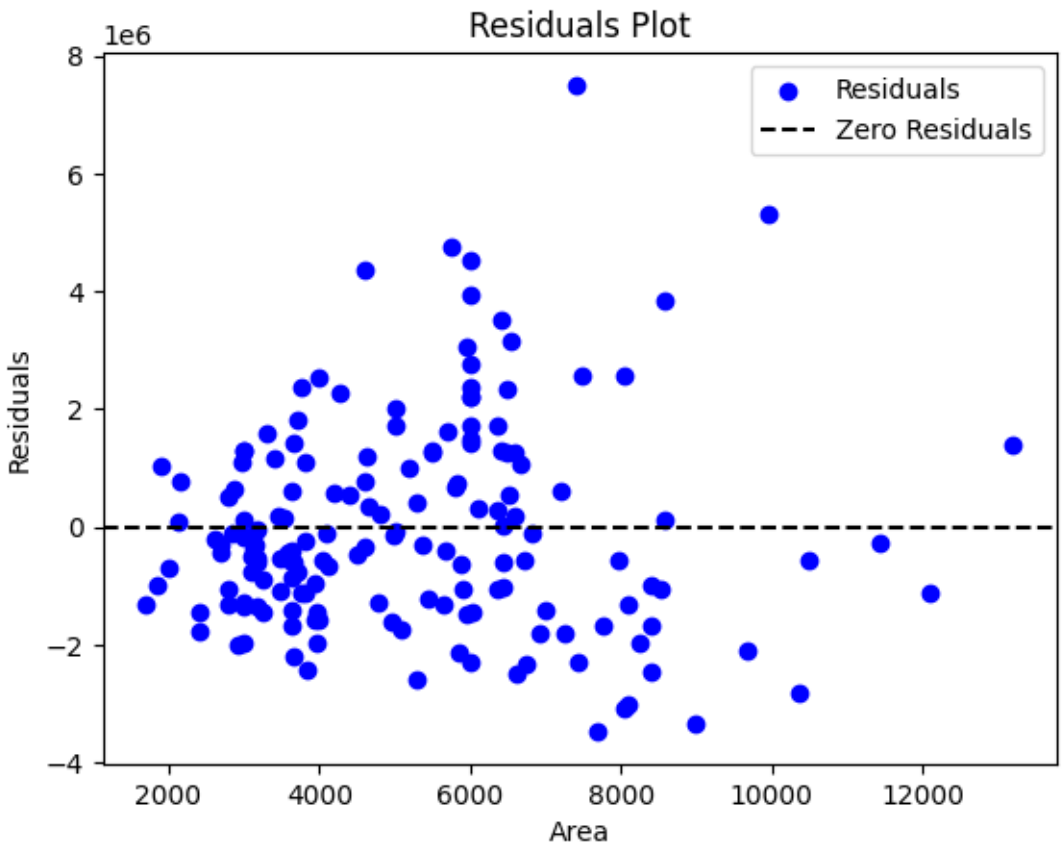
[82]:
```
plt.scatter(X_test, y_test, c='black', label='Actual Prices')
plt.plot(X_test, pred_y, c='red', label='Predicted Prices')
plt.title('Linear Regression Prediction vs Actual')
plt.xlabel('Area')
plt.ylabel('Price')
plt.legend()
plt.show()
```



[85]:
```
MSE = np.mean((y_test - pred)**2)
print("Mean Squared Error:", MSE)
```

```
Mean Squared Error: 3193534442987.1797
```

[89]:
```
residuals = y_test - pred
plt.scatter(X_test, residuals, c='blue', label='Residuals')
plt.axhline(y=0, color='black', linestyle='--', label='Zero Residuals')
plt.title('Residuals Plot')
plt.xlabel('Area')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```


Residuals Plot
1e6
8
6
4
2
0
−2
−4
Residuals
2000
4000
6000
8000
10000
12000
Area
Residuals
Zero Residuals

[83]:

[83]:

[83]:

[83]: