gpquaf5pj

February 5, 2025

1 Machine Learning Part 3

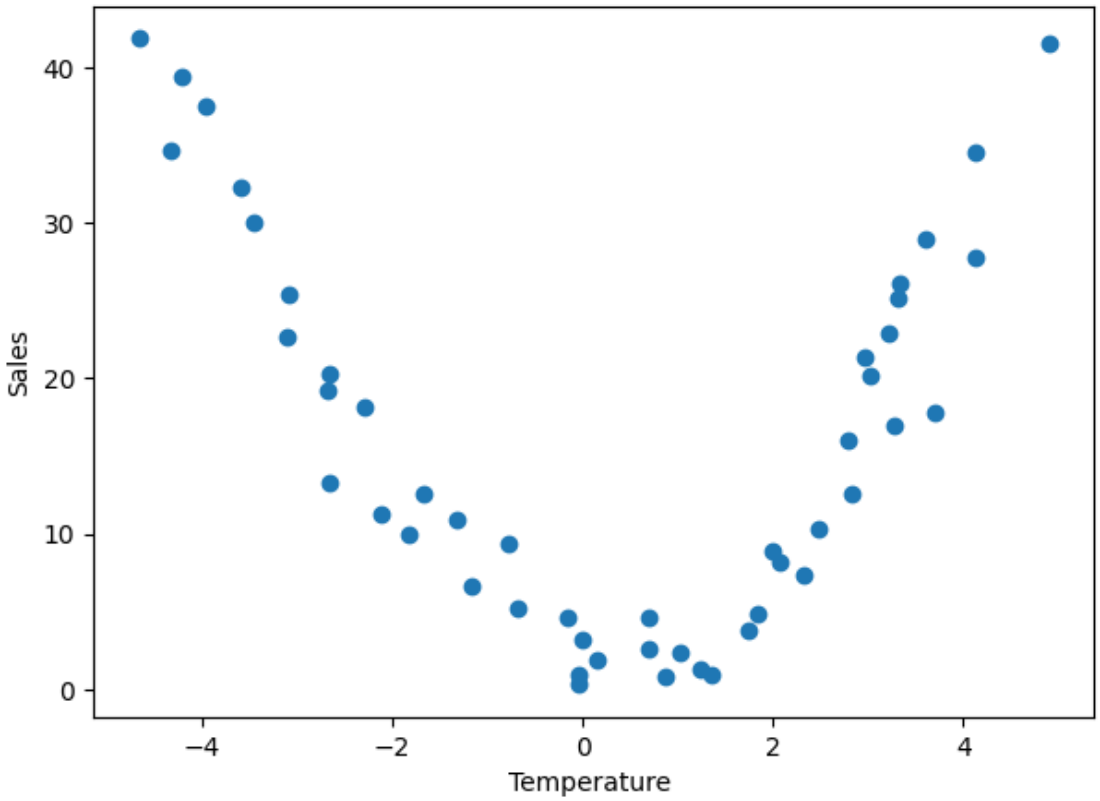
[79]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression , Lasso , Ridge
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

data = pd.read_csv("Ice_cream selling data.csv")
data.head()
```

[79]:

	Temperature (°C)	Ice Cream Sales (units)
0	-4.662263	41.842986
1	-4.316559	34.661120
2	-4.213985	39.383001
3	-3.949661	37.539845
4	-3.578554	32.284531

2 Polynomial Regression

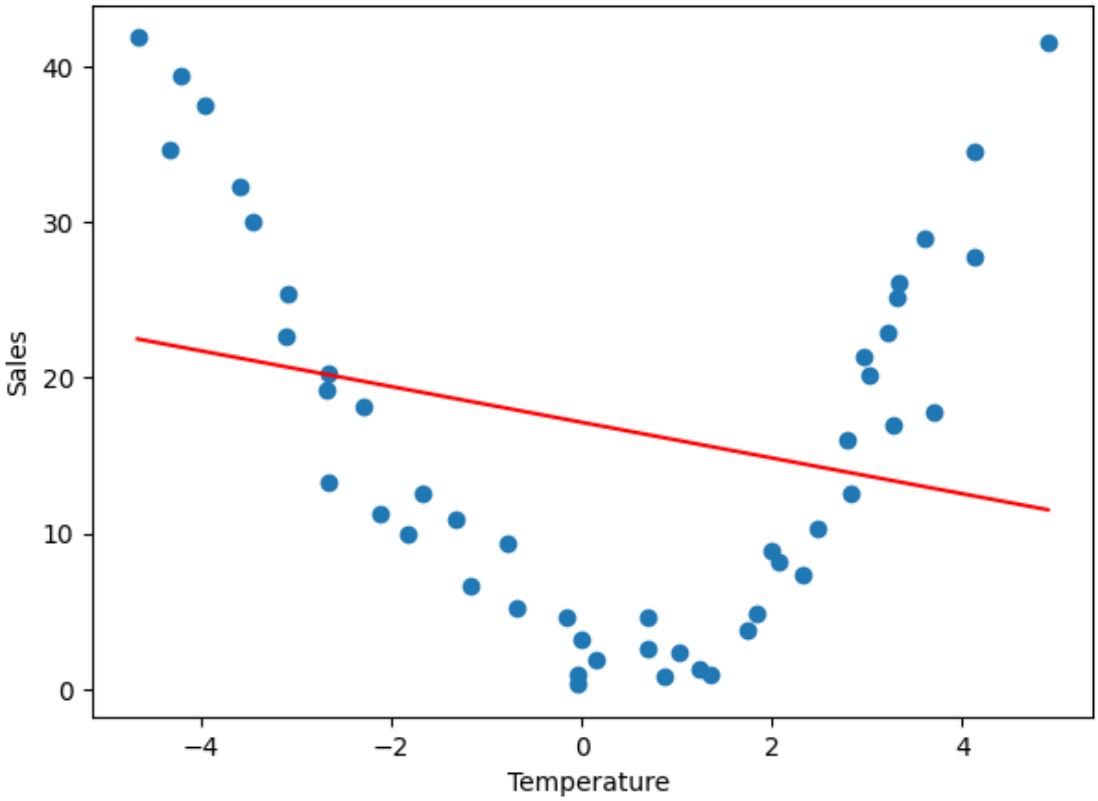
[5]:
```
plt.figure(figsize = (7 ,5))
plt.scatter(data["Temperature (°C)"],data["Ice Cream Sales (units)"])
plt.xlabel("Temperature ")
plt.ylabel("Sales")
plt.show()
```



[7]:
```
x = data[["Temperature (°C)"]]
y = data["Ice Cream Sales (units)"]

pf = PolynomialFeatures(degree = 2)

pf.fit(x)

pf.transform(x)

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.
↪2,random_state = 42)

lr = LinearRegression()

lr.fit(x_train,y_train)

lr.score(x_test,y_test) * 100
```

[7]:
```
-57.512701901882245
```

[9]:
```
plt.figure(figsize = (7 ,5))
pred = lr.predict(x)
plt.scatter(data["Temperature (°C)"],data["Ice Cream Sales (units)"])
plt.plot(data["Temperature (°C)"],pred,c = "red")
plt.xlabel("Temperature ")
plt.ylabel("Sales")
plt.show()
```



3 Cost Function

[10]:
```
# Two Type of Cost Function
# Regression Cost Function
# Classification Cost Function

# Regression Cost Function

# 1. MSE (Mean Square Error)
# 2. RMSE (Root Mean Square Error)
# 3. MAE (Mean Absolute Error)
# 4. R^2 Accuracy
```

```
# Classification Cost Function

# 1. Binary Classification Cost Function(determine 0 or 1)
# 2. Multi-Class Classification Cost Function (use Binary cross entropy , Log␣
 ↪loss )
```

4 Regularization Technique

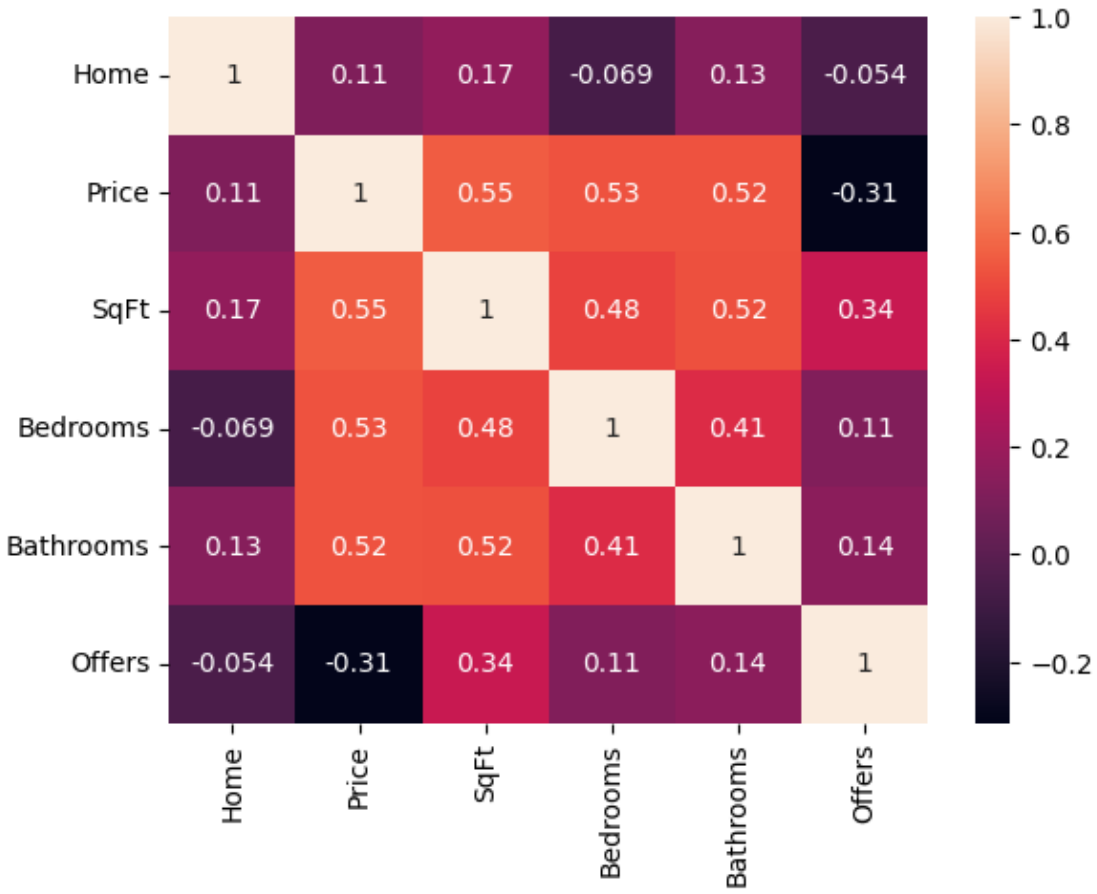
[11]:
```
# 1.Lasso Regularization L1 (Feature Selection )

    # cost Function = loss + meu * ==||w||

    # loss = sum of squared residual
    # meu = penalty
    # w = slope of the curve

# 2.Ridge Regularization L2

     # cost Function = loss + meu * ==||w||^2

    # loss = sum of squared residual
    # meu = penalty
    # w = slope of the curve
```

[38]:
```
dataset = pd.read_csv("/Users/ratnadeepgurav/Desktop/AIDS/Study for carrer␣
 ↪material/WSCUBE_Data_Analyst/ML/house-prices.csv")


# dataset.drop(columns = ["Brick","Neighborhood"],inplace = True)
dataset.head(3)
```

[38]:

	Home	Price	SqFt	Bedrooms	Bathrooms	Offers	Brick	Neighborhood
0	1	114300	1790	2	2	2	No	East
1	2	114200	2030	4	2	3	No	East
2	3	114800	1740	3	2	1	No	East

[20]:
```
numeric_dataset = dataset.select_dtypes(include=['number'])
sns.heatmap(data=numeric_dataset.corr(), annot=True)
plt.show()
```



[39]:
```
x1 = numeric_dataset.iloc[:,:-1]
y1 = numeric_dataset["Price"]
```

[62]:
```
sc1 = StandardScaler()

sc1.fit(x1)
sc1.transform(x1)
```

[62]:
```
array([[-1.71857161, -0.6025848 , -1.00091648, -1.41532739, -0.86893879],
       [-1.69150749, -0.60632122,  0.13790405,  1.35050324, -0.86893879],
       [-1.66444337, -0.58390272, -1.23817076, -0.03241208, -0.86893879],
       [-1.63737925, -1.33492252, -0.09935023, -0.03241208, -0.86893879],
       [-1.61031513, -0.39708187,  0.6124126 , -0.03241208,  1.08236235],
       [-1.58325101, -0.59137555, -1.04836734, -0.03241208, -0.86893879],
       [-1.55618689,  0.79109872, -0.81111306, -0.03241208,  1.08236235],
       [-1.52912277,  0.75747096,  0.75476517,  1.35050324, -0.86893879],
       [-1.50205865, -0.41950037,  0.51751089,  1.35050324, -0.86893879],
       [-1.47499453, -0.98743575, -1.28562162, -0.03241208,  1.08236235],
       [-1.44793041,  0.07744308,  0.13790405, -0.03241208, -0.86893879],
```

```
[-1.42086629, -0.27751653, -0.62130964, -1.41532739, -0.86893879],
[-1.39380217, -1.03974559, -0.43150622, -0.03241208, -0.86893879],
[-1.36673805, -0.15421477,  0.70731432, -0.03241208,  1.08236235],
[-1.33967393,  1.73267578,  2.79515196,  1.35050324,  1.08236235],
[-1.31260981,  0.57438653, -1.04836734,  1.35050324, -0.86893879],
[-1.28554569,  0.62295995,  0.89711774, -0.03241208,  1.08236235],
[-1.25848157, -1.74966481, -0.05189937, -0.03241208,  1.08236235],
[-1.23141745, -0.7109409 , -1.42797418, -1.41532739, -0.86893879],
[-1.20435333,  1.37397976, -0.38405536, -0.03241208,  1.08236235],
[-1.17728921, -0.53159288, -1.00091648, -0.03241208, -0.86893879],
[-1.15022509, -0.62126689, -0.00444852, -0.03241208, -0.86893879],
[-1.12316097, -1.44701503, -1.47542504, -0.03241208, -0.86893879],
[-1.09609685, -0.90897099, -0.85856392, -0.03241208, -0.86893879],
[-1.06903273,  0.97044673,  0.99201945,  1.35050324,  1.08236235],
[-1.04196861,  0.70516113,  1.37162629,  1.35050324,  1.08236235],
[-1.01490449,  0.24558184, -0.00444852,  1.35050324, -0.86893879],
[-0.98784037, -1.16304735, -1.42797418, -0.03241208, -0.86893879],
[-0.96077625, -2.29144526, -1.90248274, -1.41532739, -0.86893879],
[-0.93371213,  2.15115448,  0.1853549 ,  1.35050324,  1.08236235],
[-0.90664801,  1.92696947,  1.18182287,  1.35050324,  1.08236235],
[-0.87958389, -0.67731314, -0.33660451, -1.41532739, -0.86893879],
[-0.85251977,  0.1708535 ,  1.18182287, -0.03241208,  1.08236235],
[-0.82545565,  0.34272868,  1.32417544,  2.73341855,  1.08236235],
[-0.79839153, -0.47181021, -0.00444852, -1.41532739, -0.86893879],
[-0.77132741, -0.49796513,  0.37515833, -0.03241208,  1.08236235],
[-0.74426329, -0.48301946, -0.57385878, -1.41532739, -0.86893879],
[-0.71719918,  0.61922354,  1.98848742,  1.35050324,  1.08236235],
[-0.69013506,  0.03260608, -1.33307247, -0.03241208, -0.86893879],
[-0.66307094, -0.83050624, -1.23817076, -0.03241208, -0.86893879],
[-0.63600682, -0.89028891, -2.09228616, -1.41532739, -0.86893879],
[-0.6089427 ,  0.11854367, -0.76366221,  1.35050324,  1.08236235],
[-0.58187858, -0.92765308, -0.05189937, -1.41532739, -0.86893879],
[-0.55481446,  0.88077272, -0.38405536, -0.03241208, -0.86893879],
[-0.52775034,  1.34782484, -0.28915365, -0.03241208,  1.08236235],
[-0.50068622, -1.01732708, -0.90601477, -0.03241208, -0.86893879],
[-0.4736221 , -0.02344018, -0.05189937, -1.41532739,  1.08236235],
[-0.44655798, -1.49932487,  0.23280576, -0.03241208, -0.86893879],
[-0.41949386, -0.54280213, -0.09935023, -1.41532739, -0.86893879],
[-0.39242974, -0.85666116, -1.42797418, -0.03241208, -0.86893879],
[-0.36536562,  0.77241663,  0.47006004, -0.03241208, -0.86893879],
[-0.3383015 , -1.46943353, -0.6687605 , -1.41532739, -0.86893879],
[-0.31123738, -0.48675588,  0.70731432, -1.41532739,  1.08236235],
[-0.28417326,  0.01392399,  0.47006004, -0.03241208, -0.86893879],
[-0.25710914, -1.83560239, -1.66522846, -0.03241208, -0.86893879],
[-0.23004502, -0.17663327, -1.33307247, -1.41532739, -0.86893879],
[-0.2029809 ,  0.3913021 ,  0.89711774, -0.03241208, -0.86893879],
[-0.17591678,  0.81725363,  1.13437202,  1.35050324,  1.08236235],
```

```
[-0.14885266,  0.28668243, -0.76366221, -0.03241208,  1.08236235],
[-0.12178854,  0.93308256,  0.42260918,  1.35050324, -0.86893879],
[-0.09472442,  1.88586888,  0.94456859, -0.03241208,  1.08236235],
[-0.0676603 , -1.10326467, -1.85503189, -1.41532739, -0.86893879],
[-0.04059618,  1.15353116,  1.0394703 ,  1.35050324,  1.08236235],
[-0.01353206, -0.37092695, -0.43150622, -1.41532739,  1.08236235],
[ 0.01353206, -0.00475809, -0.6687605 , -0.03241208, -0.86893879],
[ 0.04059618, -0.72215015, -2.61424557, -1.41532739, -0.86893879],
[ 0.0676603 , -0.15795119,  0.99201945, -0.03241208,  1.08236235],
[ 0.09472442,  0.80230797,  0.1853549 ,  1.35050324,  1.08236235],
[ 0.12178854, -1.37602311,  0.65986346, -0.03241208, -0.86893879],
[ 0.14885266,  1.31419709,  0.37515833,  1.35050324,  1.08236235],
[ 0.17591678,  1.35529767, -0.2417028 , -0.03241208,  1.08236235],
[ 0.2029809 ,  1.01528373,  0.75476517,  1.35050324, -0.86893879],
[ 0.23004502, -0.86413399, -1.66522846, -0.03241208, -0.86893879],
[ 0.25710914, -0.17663327,  0.1853549 , -0.03241208,  1.08236235],
[ 0.28417326,  0.51460386,  0.65986346, -0.03241208,  1.08236235],
[ 0.31123738, -0.87907966, -0.47895707, -1.41532739, -0.86893879],
[ 0.3383015 , -0.02344018, -0.33660451, -0.03241208, -0.86893879],
[ 0.36536562,  1.72146653,  1.32417544,  1.35050324,  1.08236235],
[ 0.39242974, -0.34103562,  0.6124126 , -0.03241208, -0.86893879],
[ 0.41949386,  0.49218536, -1.04836734,  1.35050324, -0.86893879],
[ 0.44655798,  0.48471253,  0.89711774, -0.03241208,  1.08236235],
[ 0.4736221 ,  2.01290706,  0.65986346,  1.35050324,  1.08236235],
[ 0.50068622,  1.28430575,  0.23280576, -1.41532739, -0.86893879],
[ 0.52775034,  0.64537845,  1.94103656, -0.03241208,  1.08236235],
[ 0.55481446, -1.49185204, -2.28208959, -1.41532739, -0.86893879],
[ 0.58187858,  2.16236373,  1.18182287,  1.35050324,  1.08236235],
[ 0.6089427 , -1.03600917, -0.47895707,  1.35050324, -0.86893879],
[ 0.63600682,  1.57200986, -0.57385878, -0.03241208,  1.08236235],
[ 0.66307094, -0.10190493, -0.33660451, -0.03241208,  1.08236235],
[ 0.69013506, -1.2190936 ,  0.04300234, -1.41532739, -0.86893879],
[ 0.71719918,  0.47350328, -0.38405536,  1.35050324, -0.86893879],
[ 0.74426329, -0.52038363,  0.70731432, -0.03241208, -0.86893879],
[ 0.77132741,  0.45482119,  0.51751089, -0.03241208, -0.86893879],
[ 0.79839153,  0.99660165,  0.37515833, -0.03241208,  1.08236235],
[ 0.82545565,  1.12737624,  0.70731432,  1.35050324,  1.08236235],
[ 0.85251977,  0.82472647, -0.14680108, -1.41532739, -0.86893879],
[ 0.87958389,  0.10733441,  2.08338913, -0.03241208,  1.08236235],
[ 0.90664801, -0.13553269, -0.00444852, -1.41532739, -0.86893879],
[ 0.93371213,  0.56317728,  0.28025662, -0.03241208, -0.86893879],
[ 0.96077625,  1.5159636 ,  0.37515833, -0.03241208,  1.08236235],
[ 0.98784037, -1.01732708,  0.04300234, -0.03241208, -0.86893879],
[ 1.01490449, -0.27378011,  1.22927373, -0.03241208,  1.08236235],
[ 1.04196861,  0.23810901,  1.94103656, -0.03241208,  1.08236235],
[ 1.06903273,  3.01800321,  2.08338913,  1.35050324,  1.08236235],
[ 1.09609685, -1.79823823, -0.43150622, -0.03241208, -0.86893879],
```

```
[ 1.12316097,  0.61548712,  2.51044683,  1.35050324,  1.08236235],
[ 1.15022509, -0.81929699,  0.6124126 , -0.03241208, -0.86893879],
[ 1.17728921,  0.13348933, -0.52640793, -0.03241208, -0.86893879],
[ 1.20435333, -0.50170155, -0.05189937, -0.03241208,  1.08236235],
[ 1.23141745, -0.81182415,  0.51751089, -0.03241208, -0.86893879],
[ 1.25848157, -0.70346806, -1.38052333, -1.41532739, -0.86893879],
[ 1.28554569, -0.5801663 , -1.23817076, -1.41532739, -0.86893879],
[ 1.31260981, -0.25509803, -0.28915365, -1.41532739, -0.86893879],
[ 1.33967393, -0.55027497, -0.00444852, -0.03241208, -0.86893879],
[ 1.36673805, -0.22147028,  0.04300234,  1.35050324,  1.08236235],
[ 1.39380217, -1.043482  , -0.47895707, -0.03241208,  1.08236235],
[ 1.42086629,  2.58084243,  1.37162629,  2.73341855,  3.03366348],
[ 1.44793041, -0.47181021, -0.38405536, -0.03241208, -0.86893879],
[ 1.47499453,  0.73878888, -0.2417028 , -0.03241208, -0.86893879],
[ 1.50205865, -0.77445998, -0.38405536, -1.41532739, -0.86893879],
[ 1.52912277, -0.74830506, -0.33660451, -1.41532739,  1.08236235],
[ 1.55618689, -0.92765308, -0.33660451, -0.03241208,  1.08236235],
[ 1.58325101,  0.53702236,  0.28025662, -1.41532739, -0.86893879],
[ 1.61031513, -0.40081829, -0.47895707, -0.03241208,  1.08236235],
[ 1.63737925,  0.65285129,  0.75476517,  1.35050324,  1.08236235],
[ 1.66444337, -0.63247614,  0.32770747, -1.41532739, -0.86893879],
[ 1.69150749,  0.72757963,  0.09045319, -0.03241208,  1.08236235],
[ 1.71857161, -0.21773386,  1.18182287, -0.03241208,  1.08236235]])
```

[77]:
```
x_train,x_test,y_train,y_test = train_test_split(x1,y1,test_size = 0.
↪2,random_state = 42)

lr1 = LinearRegression()

lr1.fit(x_train, y_train)

print("Accuracy Are :- ",lr1.score(x_test,y_test)*100,"%")
```

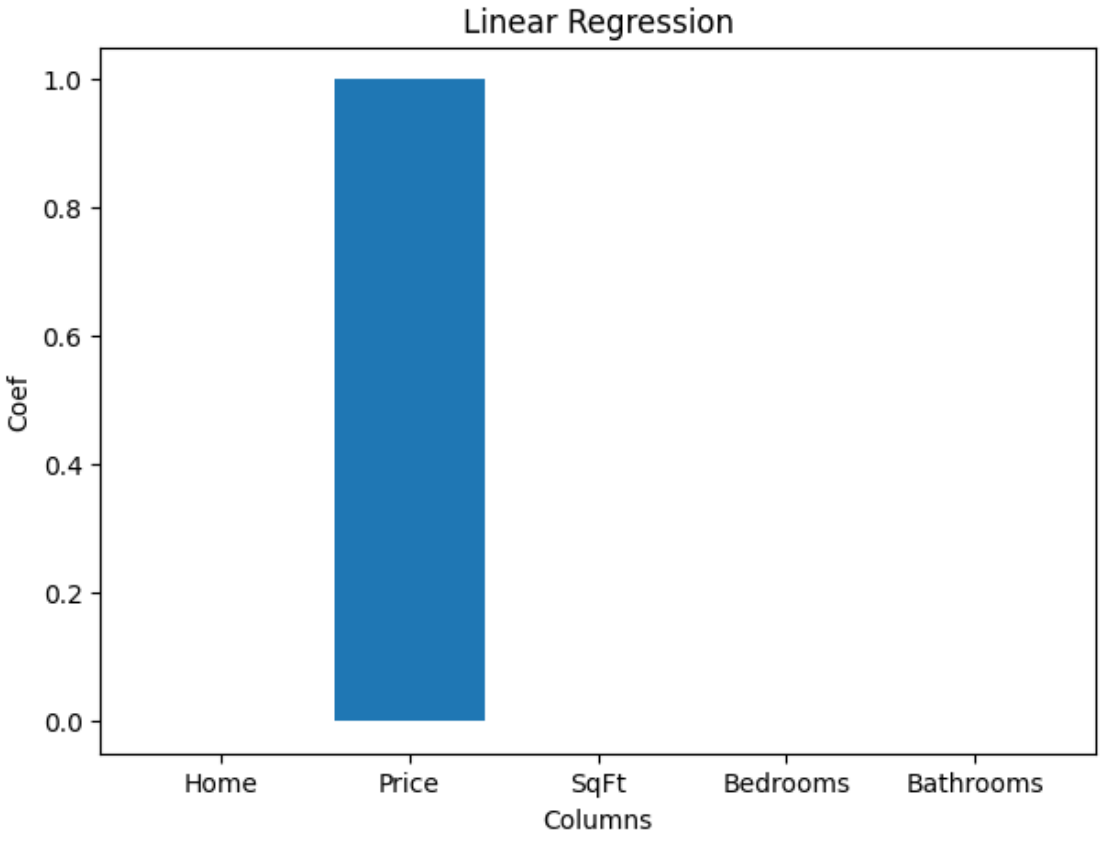
```
Accuracy Are :-  100.0 %
```

5 MSE , MAE , Sqrt (Linear Regression)

[81]:
```
print(mean_squared_error(y_test,lr1.predict(x_test)))
print(mean_absolute_error(y_test,lr1.predict(x_test)))
print(np.sqrt(mean_squared_error(y_test,lr1.predict(x_test))))
```

```
4.072273784876444e-23
2.7984452362243947e-12
6.381436973657613e-12
```

[58]:
```
plt.figure(figsize = (7,5))
plt.bar(x1.columns , lr1.coef_)
plt.title("Linear Regression")
plt.xlabel("Columns")
plt.ylabel("Coef")
plt.show()
```



6 Lasso Regularization

[82]:
```
ls = Lasso(alpha = 0.01)
ls.fit(x_train,y_train)
ls.score(x_test,y_test)*100
```

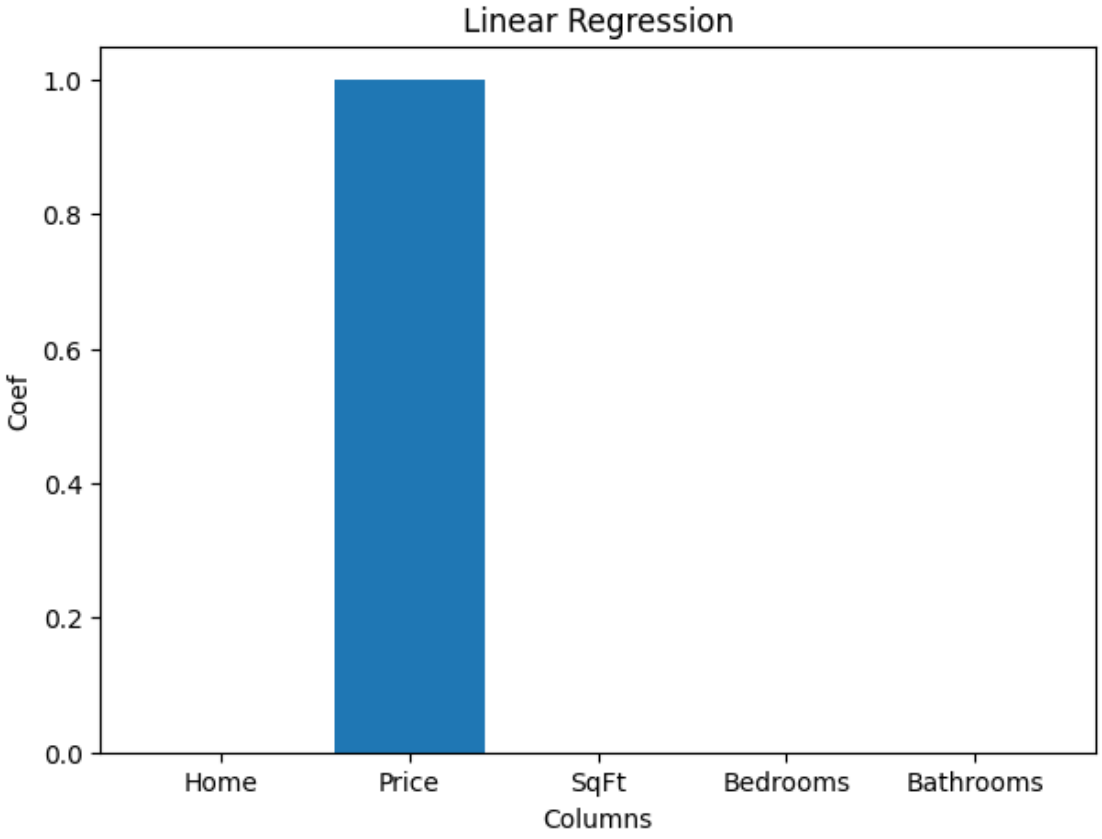
[82]: 99.99999999999237

7 MSE , MAE , Sqrt (Lasso Regularization)

[84]:
```
print(mean_squared_error(y_test,ls.predict(x_test)))
print(mean_absolute_error(y_test,ls.predict(x_test)))
print(np.sqrt(mean_squared_error(y_test,ls.predict(x_test))))
```

```
4.49281312773769e-05
0.005261028317573409
0.00670284501367717
```

[71]:
```
plt.figure(figsize = (7,5))
plt.bar(x1.columns , ls.coef_)
plt.title("Linear Regression")
plt.xlabel("Lasso")
plt.ylabel("Coef")
plt.show()
```



8 Ridge Regularization

[83]:
```
ri = Ridge(alpha = 10)
ri.fit(x_train,y_train)
ri.score(x_test,y_test)*100
```

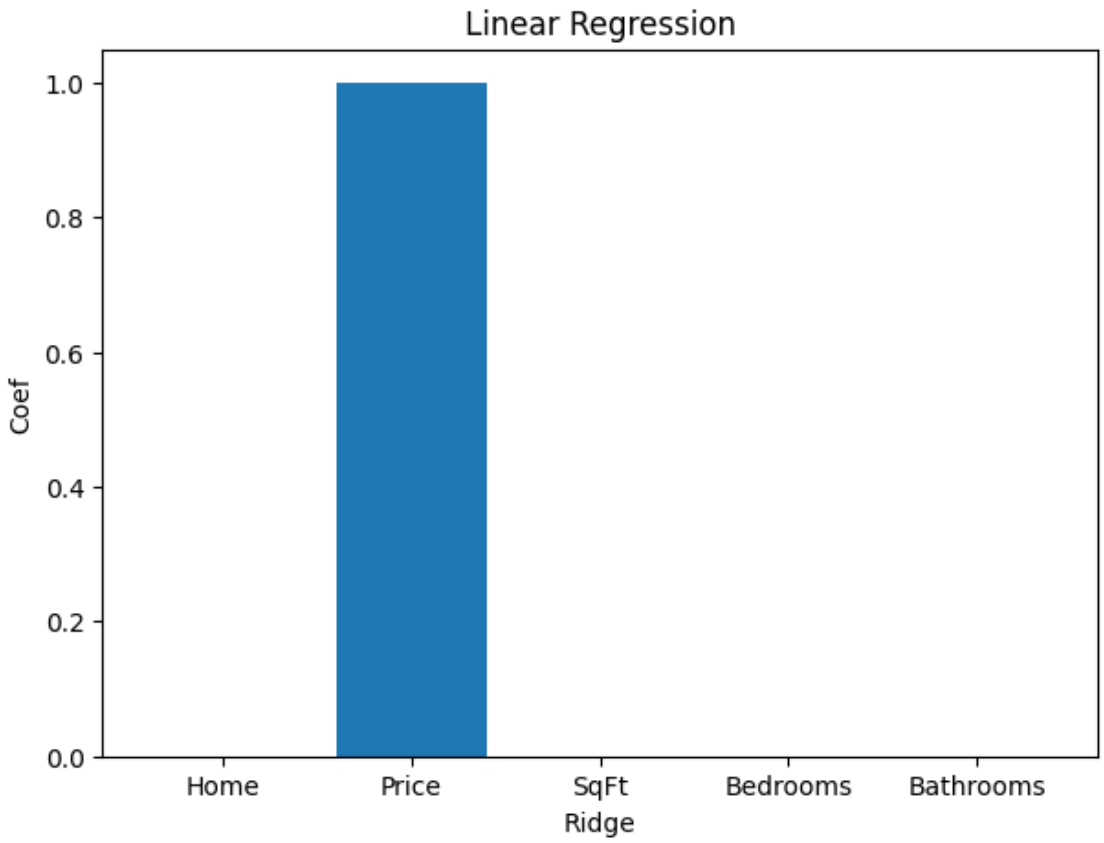
[83]: 100.0

9 MSE , MAE , Sqrt (Ridge Regularization)

[86]:
```
print(mean_squared_error(y_test,ri.predict(x_test)))
print(mean_absolute_error(y_test,ri.predict(x_test)))
print(np.sqrt(mean_squared_error(y_test,ri.predict(x_test))))
```

```
1.699484056352091e-11
3.4043461290331415e-06
4.122479904562411e-06
```

[75]:
```
plt.figure(figsize = (7,5))
plt.bar(x1.columns , ls.coef_)
plt.title("Linear Regression")
plt.xlabel("Ridge")
plt.ylabel("Coef")
plt.show()
```



[92]:
```
data_show = pd.DataFrame({"Columns ":x1.columns,"Linear Regression":lr1.
 ↪coef_,"Lasso":ls.coef_,"Ridge":ri.coef_})
data_show
```

[92]:

	Columns	Linear Regression	Lasso	Ridge
0	Home	2.326706e-14	0.000031	1.310271e-08
1	Price	1.000000e+00	1.000000	1.000000e+00
2	SqFt	1.185332e-14	0.000023	1.048505e-08
3	Bedrooms	4.666661e-13	0.000000	2.093017e-06
4	Bathrooms	-5.045109e-12	0.000000	1.943074e-06