linearregression-example

April 18, 2023

0.1 Predict Banglore House Price Using Liearn Regression - Supervised Learning Algo.

[1]:
```
import pandas as pd
```

[2]:
```
df = pd.read_csv("/content/drive/MyDrive/MyDataSet/bangalore house price prediction OHE-data.csv")
df.head()
```

[2]:

	bath	balcony	price	total_sqft_int	bhk	price_per_sqft	\
0	3.0	2.0	150.0	1672.0	3	8971.291866	
1	3.0	3.0	149.0	1750.0	3	8514.285714	
2	3.0	2.0	150.0	1750.0	3	8571.428571	
3	2.0	2.0	40.0	1250.0	2	3200.000000	
4	2.0	2.0	83.0	1200.0	2	6916.666667	

	area_typeSuper built-up Area	area_typeBuilt-up Area	\
0	1	0	
1	0	1	
2	1	0	
3	1	0	
4	0	0	

	area_typePlot Area	availability_Ready To Move	…	\
0	0	1	…	
1	0	1	…	
2	0	1	…	
3	0	1	…	
4	1	1	…	

	location_Kalena Agrahara	location_Horamavu Agara	location_Vidyaranyapura	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	

	location_BTM 2nd Stage	location_Hebbal Kempapura	location_Hosur Road	\

```
0                         0                         0                    0
1                         0                         0                    0
2                         0                         0                    0
3                         0                         0                    0
4                         0                         0                    0

   location_Horamavu Banaswadi  location_Domlur  location_Mahadevpura  \
0                            0                0                     0
1                            0                0                     0
2                            0                0                     0
3                            0                0                     0
4                            0                0                     0

   location_Tumkur Road
0                     0
1                     0
2                     0
3                     0
4                     0

[5 rows x 108 columns]
```

[3]:
```
# "price" is our dependent Variable
X = df.drop("price",axis=1) # X = Matrix
y = df["price"]             # y = Vector
print(X.shape)
print(y.shape)
```

```
(7120, 107)
(7120,)
```

[4]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,␣
 ↪random_state=51)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(5696, 107)
(5696,)
(1424, 107)
(1424,)
```

1 Feature Scaling

[5]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

#Linear Regression - ML- Model Training

[6]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

lr.fit(X_train,y_train)
```

[6]:
```
LinearRegression()
```

[7]:
```
lr.coef_
```

[7]:
```
array([-5.70206143e+00, -1.25679916e+00,  8.27341833e+01, -1.44906911e+01,
        5.75662723e+01,  1.88468905e-01, -1.72593897e+00, -4.51058311e+00,
       -2.22589244e+00, -4.28978455e+00, -2.44590976e+00,  5.40246226e-01,
       -1.03633400e+00,  1.43064873e+00, -6.25029424e-02, -1.51548783e+00,
       -2.14422789e-01,  2.16244155e+00, -1.48710228e+00,  1.95250816e+00,
       -3.10761125e+00, -1.28138668e+00, -1.01367155e+00,  1.37968545e-02,
        1.10383858e+00,  1.26497611e+00, -3.52405517e+00, -1.21398741e+00,
       -5.04622019e-01,  1.46299181e+00, -5.50064233e-01, -8.46468162e-02,
        6.84882188e-01, -1.39849820e+00, -1.94761710e-02, -1.57716300e+00,
        4.20886278e-01,  8.03443207e-01,  2.99182164e+00,  3.86430413e-03,
        1.05037261e-01,  2.89115612e-01, -3.16916626e-01,  1.05625868e+00,
       -1.39649279e+00, -3.10533604e+00,  1.01764011e-01, -7.49672917e-02,
       -8.03271555e-01, -1.27061856e+00, -8.54046164e-01,  2.64566484e-01,
        9.10688839e-01, -8.23059458e-01, -9.07215234e-01,  1.22059216e+00,
        2.11418894e+00, -5.38187400e-01, -1.32164338e+00, -8.28349340e-01,
        1.28167980e+00, -1.92911295e-01,  6.65824485e-02,  3.65563139e-02,
       -1.85069853e+00,  1.49068024e+00, -9.57964753e-01, -9.36110163e-01,
       -7.45634897e-01,  7.22643165e-02, -6.79260144e-01, -1.70853833e-01,
       -1.72288643e+00, -1.15833746e+00,  5.78931788e-01,  1.37836966e+00,
       -1.14424496e+00,  3.96188294e-01, -6.08013157e-01, -2.20959218e+00,
        3.45270810e-01,  1.01747431e-03,  1.06563895e-01,  3.04728530e+00,
        2.09496392e+00, -8.13481923e-01, -4.18437282e-01,  2.30993396e+00,
        3.31858800e-02,  8.07865914e-02,  5.37064987e-02,  1.55347699e+00,
        8.13889657e-01, -1.14636462e+00,  3.41805788e-01, -8.28022037e-01,
        1.68897360e+00,  2.97657524e-01,  9.59437517e-01,  4.57297702e-01,
       -2.22729515e-01, -1.48290835e+00, -6.26342867e-01,  5.86538254e-01,
       -1.78547310e+00,  2.19020231e-01, -3.45032599e-01])
```

[8]:
```
lr.intercept_
```

[8]:
```
95.0802729985955
```

2 Predict the Value of Home and Test

[9]:
```
X_test[0,:]
```

[9]:
```
array([ 0.71301986,  0.0112734 ,  0.30202307,  0.65677518, -0.48064341,
       -1.7385623 ,  2.11587407, -0.25430867,  0.51007548, -0.18373025,
       -0.16389438, -0.1473229 , -0.13023539, -0.12812824, -0.12598816,
       -0.12454231, -0.12953656, -0.12381344, -0.12010681, -0.11551113,
       -0.10992018, -0.10909925, -0.10660036, -0.11234866, -0.09315135,
       -0.08618799, -0.08923672, -0.09023078, -0.08721571, -0.09023078,
       -0.08721571, -0.08195215, -0.08195215, -0.07633675, -0.0751646 ,
       -0.08085949, -0.0739743 , -0.07975227, -0.07153563, -0.0751646 ,
       -0.0677166 , -0.08085949, -0.07153563, -0.07862985, -0.0751646 ,
       -0.07862985, -0.06504853, -0.0751646 , -0.06901264, -0.0751646 ,
       -0.06901264, -0.07028523, -0.07276497, -0.07028523, -0.06367332,
       -0.06226825, -0.06226825, -0.06639573, -0.06504853, -0.05935999,
       -0.06083125, -0.06639573, -0.06639573, -0.06226825, -0.06367332,
       -0.05935999, -0.06639573, -0.06367332, -0.06226825, -0.06226825,
       -0.05935999, -0.05935999, -0.05935999, -0.05630391, -0.05935999,
       -0.05785186, -0.05935999, -0.05935999, -0.06083125, -0.06083125,
       -0.05471275, -0.06083125, -0.06226825, -0.05935999, -0.05935999,
       -0.06226825, -0.06226825, -0.05785186, -0.06504853, -0.06226825,
       -0.06083125, -0.05935999, -0.05307449, -0.05630391, -0.06226825,
       -0.05471275, -0.05935999, -0.05471275, -0.05471275, -0.05138463,
       -0.05307449, -0.05307449, -0.05471275, -0.05471275, -0.05630391,
       -0.05630391, -0.05138463])
```

[10]:
```
lr.predict([X_test[0,:]])
```

[10]:
```
array([76.90661876])
```

[11]:
```
lr.predict(X_test)
```

[11]:
```
array([ 76.90661876,  15.25005377, 113.6828165 , …,  21.30296864,
        71.43462962, 230.0414626 ])
```

[12]:
```
y_test
```

[12]:
```
2435     80.00
3113     40.00
426     120.00
1124     79.00
```

```
1161     45.00
         …
2078     28.34
6855     84.00
4381     32.00
3862     63.00
43      180.00
Name: price, Length: 1424, dtype: float64
```

#Check Accuracy for Train Predicted Data and Test Data

[13]:
```
lr.score(X_test,y_test)
```

[13]:
```
0.7903837092682251
```

#Implementing Ridge & Lasso Regression

[14]:
```
from sklearn.linear_model import Ridge, Lasso
```

[15]:
```
rd = Ridge()
rd.fit(X_train,y_train)
```

[15]:
```
Ridge()
```

[16]:
```
rd.score(X_test, y_test)
```

[16]:
```
0.7905686374336628
```

[17]:
```
ls =  Lasso()
ls.fit(X_train,y_train)
ls.score(X_test, y_test)
```

[17]:
```
0.8036373003525774
```

[18]:
```
rd2 = Ridge(alpha=2)
rd2.fit(X_train,y_train)
rd2.score(X_test, y_test)
```

[18]:
```
0.7907530260397942
```

[19]:
```
ls2=  Lasso(alpha=2)
ls2.fit(X_train,y_train)
ls2.score(X_test, y_test)
```

[19]:
```
0.8160181533703601
```

[20]:
```
ls3=  Lasso(alpha=3)
ls3.fit(X_train,y_train)
```

```
ls3.score(X_test, y_test)
```

[20]: 0.8263450613017426