central on linear regression

March 16, 2024

1 TUGUME JAMES : 2023-U-MMU-BCS-0I680, LINEAR REGRESSION

[ ]:
```
# importing the neccessary
import pandas as pd
import numpy as np
from sklearn.model_selection import  train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
```

[79]:
```
# reading the dataset
data = pd.read_csv("C:\\Users\\tugumejame\\Desktop\\LINEAR PROGRAMMING CENTRAL␣
↪TEST\\Linear Regression - Sheet1.csv")
```

[80]:
```
data
```

[80]:
```
        X           y
0       1    3.888889
1       2    4.555556
2       3    5.222222
3       4    5.888889
4       5    6.555556
..    …          …
295   296  200.555556
296   297  201.222222
297   298  201.888889
298   299    1.888889
299   300    1.888889

[300 rows x 2 columns]
```

[81]:
```
X = np.array(data["X"]).reshape(-1,1)
y = np.array(data["y"])
```

[82]:
```
X
```

```
[82]: array([[  1],
             [  2],
             [  3],
             [  4],
             [  5],
             [  6],
             [  7],
             [  8],
             [  9],
             [ 10],
             [ 11],
             [ 12],
             [ 13],
             [ 14],
             [ 15],
             [ 16],
             [ 17],
             [ 18],
             [ 19],
             [ 20],
             [ 21],
             [ 22],
             [ 23],
             [ 24],
             [ 25],
             [ 26],
             [ 27],
             [ 28],
             [ 29],
             [ 30],
             [ 31],
             [ 32],
             [ 33],
             [ 34],
             [ 35],
             [ 36],
             [ 37],
             [ 38],
             [ 39],
             [ 40],
             [ 41],
             [ 42],
             [ 43],
             [ 44],
             [ 45],
             [ 46],
             [ 47],
```

```
[ 48],
[ 49],
[ 50],
[ 51],
[ 52],
[ 53],
[ 54],
[ 55],
[ 56],
[ 57],
[ 58],
[ 59],
[ 60],
[ 61],
[ 62],
[ 63],
[ 64],
[ 65],
[ 66],
[ 67],
[ 68],
[ 69],
[ 70],
[ 71],
[ 72],
[ 73],
[ 74],
[ 75],
[ 76],
[ 77],
[ 78],
[ 79],
[ 80],
[ 81],
[ 82],
[ 83],
[ 84],
[ 85],
[ 86],
[ 87],
[ 88],
[ 89],
[ 90],
[ 91],
[ 92],
[ 93],
[ 94],
```

```
[ 95],
[ 96],
[ 97],
[ 98],
[ 99],
[100],
[101],
[102],
[103],
[104],
[105],
[106],
[107],
[108],
[109],
[110],
[111],
[112],
[113],
[114],
[115],
[116],
[117],
[118],
[119],
[120],
[121],
[122],
[123],
[124],
[125],
[126],
[127],
[128],
[129],
[130],
[131],
[132],
[133],
[134],
[135],
[136],
[137],
[138],
[139],
[140],
[141],
```

[142],
[143],
[144],
[145],
[146],
[147],
[148],
[149],
[150],
[151],
[152],
[153],
[154],
[155],
[156],
[157],
[158],
[159],
[160],
[161],
[162],
[163],
[164],
[165],
[166],
[167],
[168],
[169],
[170],
[171],
[172],
[173],
[174],
[175],
[176],
[177],
[178],
[179],
[180],
[181],
[182],
[183],
[184],
[185],
[186],
[187],
[188],

[189],
[190],
[191],
[192],
[193],
[194],
[195],
[196],
[197],
[198],
[199],
[200],
[201],
[202],
[203],
[204],
[205],
[206],
[207],
[208],
[209],
[210],
[211],
[212],
[213],
[214],
[215],
[216],
[217],
[218],
[219],
[220],
[221],
[222],
[223],
[224],
[225],
[226],
[227],
[228],
[229],
[230],
[231],
[232],
[233],
[234],
[235],

[236],
[237],
[238],
[239],
[240],
[241],
[242],
[243],
[244],
[245],
[246],
[247],
[248],
[249],
[250],
[251],
[252],
[253],
[254],
[255],
[256],
[257],
[258],
[259],
[260],
[261],
[262],
[263],
[264],
[265],
[266],
[267],
[268],
[269],
[270],
[271],
[272],
[273],
[274],
[275],
[276],
[277],
[278],
[279],
[280],
[281],
[282],

```
       [283],
       [284],
       [285],
       [286],
       [287],
       [288],
       [289],
       [290],
       [291],
       [292],
       [293],
       [294],
       [295],
       [296],
       [297],
       [298],
       [299],
       [300]], dtype=int64)
```

[83]:
```
y
```

[83]:
```
array([  3.88888889,   4.55555556,   5.22222222,   5.88888889,
         6.55555556,   7.22222222,   7.88888889,   8.55555556,
         9.22222222,   9.88888889,  10.55555556,  11.22222222,
        11.88888889,  12.55555556,  13.22222222,  13.88888889,
        14.55555556,  15.22222222,  15.88888889,  16.55555556,
        17.22222222,  17.88888889,  18.55555556,  19.22222222,
        19.88888889,  20.55555556,  21.22222222,  21.88888889,
        22.55555556,  23.22222222,  23.88888889,  24.55555556,
        25.22222222,  25.88888889,  26.55555556,  27.22222222,
        27.88888889,  28.55555556,  29.22222222,  29.88888889,
        30.55555556,  31.22222222,  31.88888889,  32.55555556,
        33.22222222,  33.88888889,  34.55555556,  35.22222222,
        35.88888889,  36.55555556,  37.22222222,  37.88888889,
        38.55555556,  39.22222222,  39.88888889,  40.55555556,
        41.22222222,  41.88888889,  42.55555556,  43.22222222,
        43.88888889,  44.55555556,  45.22222222,  45.88888889,
        46.55555556,  47.22222222,  47.88888889,  48.55555556,
        49.22222222,  49.88888889,  50.55555556,  51.22222222,
        51.88888889,  52.55555556,  53.22222222,  53.88888889,
        54.55555556,  55.22222222,  55.88888889,  56.55555556,
        57.22222222,  57.88888889,  58.55555556,  59.22222222,
        59.88888889,  60.55555556,  61.22222222,  61.88888889,
        62.55555556,  63.22222222,  63.88888889,  64.55555556,
        65.22222222,  65.88888889,  66.55555556,  67.22222222,
        67.88888889,  68.55555556,  69.22222222,  69.88888889,
        70.55555556,  71.22222222,  71.88888889,  72.55555556,
```

```
 73.22222222,  73.88888889,  74.55555556,  75.22222222,
 75.88888889,  76.55555556,  77.22222222,  77.88888889,
 78.55555556,  79.22222222,  79.88888889,  80.55555556,
 81.22222222,  81.88888889,  82.55555556,  83.22222222,
 83.88888889,  84.55555556,  85.22222222,  85.88888889,
 86.55555556,  87.22222222,  87.88888889,  88.55555556,
 89.22222222,  89.88888889,  90.55555556,  91.22222222,
 91.88888889,  92.55555556,  93.22222222,  93.88888889,
 94.55555556,  95.22222222,  95.88888889,  96.55555556,
 97.22222222,  97.88888889,  98.55555556,  99.22222222,
 99.88888889, 100.5555556 , 101.2222222 , 101.8888889 ,
102.5555556 , 103.2222222 , 103.8888889 , 104.5555556 ,
105.2222222 , 105.8888889 , 106.5555556 , 107.2222222 ,
107.8888889 , 108.5555556 , 109.2222222 , 109.8888889 ,
110.5555556 , 111.2222222 , 111.8888889 , 112.5555556 ,
113.2222222 , 113.8888889 , 114.5555556 , 115.2222222 ,
115.8888889 , 116.5555556 , 117.2222222 , 117.8888889 ,
118.5555556 , 119.2222222 , 119.8888889 , 120.5555556 ,
121.2222222 , 121.8888889 , 122.5555556 , 123.2222222 ,
123.8888889 , 124.5555556 , 125.2222222 , 125.8888889 ,
126.5555556 , 127.2222222 , 127.8888889 , 128.5555556 ,
129.2222222 , 129.8888889 , 130.5555556 , 131.2222222 ,
131.8888889 , 132.5555556 , 133.2222222 , 133.8888889 ,
134.5555556 , 135.2222222 , 135.8888889 , 136.5555556 ,
137.2222222 , 137.8888889 , 138.5555556 , 139.2222222 ,
139.8888889 , 140.5555556 , 141.2222222 , 141.8888889 ,
142.5555556 , 143.2222222 , 143.8888889 , 144.5555556 ,
145.2222222 , 145.8888889 , 146.5555556 , 147.2222222 ,
147.8888889 , 148.5555556 , 149.2222222 , 149.8888889 ,
150.5555556 , 151.2222222 , 151.8888889 , 152.5555556 ,
153.2222222 , 153.8888889 , 154.5555556 , 155.2222222 ,
155.8888889 , 156.5555556 , 157.2222222 , 157.8888889 ,
158.5555556 , 159.2222222 , 159.8888889 , 160.5555556 ,
161.2222222 , 161.8888889 , 162.5555556 , 163.2222222 ,
163.8888889 , 164.5555556 , 165.2222222 , 165.8888889 ,
166.5555556 , 167.2222222 , 167.8888889 , 168.5555556 ,
169.2222222 , 169.8888889 , 170.5555556 , 171.2222222 ,
171.8888889 , 172.5555556 , 173.2222222 , 173.8888889 ,
174.5555556 , 175.2222222 , 175.8888889 , 176.5555556 ,
177.2222222 , 177.8888889 , 178.5555556 , 179.2222222 ,
179.8888889 , 180.5555556 , 181.2222222 , 181.8888889 ,
182.5555556 , 183.2222222 , 183.8888889 , 184.5555556 ,
185.2222222 , 185.8888889 , 186.5555556 , 187.2222222 ,
187.8888889 , 188.5555556 , 189.2222222 , 189.8888889 ,
190.5555556 , 191.2222222 , 191.8888889 , 192.5555556 ,
193.2222222 , 193.8888889 , 194.5555556 , 195.2222222 ,
195.8888889 , 196.5555556 , 197.2222222 , 197.8888889 ,
```

```
       198.5555556 , 199.2222222 , 199.8888889 , 200.5555556 ,
       201.2222222 , 201.8888889 ,   1.88888889,   1.88888889])
```

[87]:
```
# Spliting data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 42)
```

[88]:
```
#Standardizing the  independent variable,data preprocessing
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[89]:
```
# BUILDING LINEAR REGRESSION MODEL
model = LinearRegression()
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
```

[90]:
```
y_pred
```

[90]:
```
array([135.74657151, 175.63456787, 103.45628874,  12.91686842,
       154.74085549, 150.3088559 , 131.31457191,  76.23114836,
        10.38429722, 118.01857313, 157.27342669,  43.30772279,
       145.2437135 ,  35.7100092 , 122.45057272, 147.1431419 ,
       190.19685226, 140.8117139 , 100.92371754, 111.68714513,
        56.60372158,  78.76371956, 164.87114028, 165.50428308,
        73.06543437,  33.8105808 , 185.13170986, 193.99570906,
       106.62200274, 157.90656949,  17.98201082, 111.05400233,
        28.11229561,  22.41401041, 143.3442851 ,  82.56257636,
        11.65058282,  64.20143517,  36.343152  ,  53.43800758,
        66.10086357,  55.33743598, 188.29742386,  45.20715119,
        55.97057878,  47.10657959, 155.37399829, 152.20828429,
        77.49743396, 153.47456989, 121.18428712,  98.39114634,
       158.53971229,  54.70429318, 195.26199465, 183.23228147,
        68.63343477,  65.46772077, 128.78200072,  23.04715321])
```

[91]:
```
# Evaluate the model
mas = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2  = r2_score(y_test, y_pred)
```

[92]:
```
mas
```

[92]:
```
2.8293196617609264
```

[93]:
```
mse
```

[93]:
```
11.554304857221384
```

[94]:
```
r2
```

[94]:
```
0.9966137419987549
```

[95]:
```
# getting the intercept, evualting the model performance
model.intercept_
```

[95]:
```
101.73888889500417
```

[96]:
```
model.coef_
```

[96]:
```
array([54.66833305])
```

[97]:
```
score = model.score(X_test_scaled, y_test)
score
```

[97]:
```
0.9966137419987549
```

2 model optmization on linear regression

[98]:
```
# importing necessary libraries
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import  numpy as np
import pandas as pd
```

[99]:
```
# reading the dataset
data = pd.read_csv("C:\\Users\\tugumejames\\Desktop\\LINEAR PROGRAMMING CENTRAL␣
 ↪TEST\\Linear Regression - Sheet1.csv")
```

[74]:
```
X = np.array(data["X"]).reshape(-1,1)     # independent variables
y = np.array(data["y"])                   # dependent variables
```

[75]:
```
# spliting the data into test and train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 42)
```

[76]:
```
#Standardizing the  independent variable
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Initialize linear regression model
lr = LinearRegression()
```

[77]:
```
# Defining hyperparameters to tune
param_grid = {
```

```
    'fit_intercept': [True, False],
    'positive': [True, False]  # Adding 'positive' parameter
}

# Perform grid search with cross-validation

grid_search = GridSearchCV(lr, param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')
grid_search.fit(X_train_scaled, y_train)
```

[77]:
```
GridSearchCV(cv=5, estimator=LinearRegression(),
             param_grid={'fit_intercept': [True, False],
                         'positive': [True, False]},
             scoring='neg_mean_squared_error')
```

[78]:
```
# Get best hyperparameters
best_params = grid_search.best_params_
print(f"best_params",best_params)
# Create a new instance of LinearRegression with best parameters
best_lr = LinearRegression(**best_params)

# Fit the model with best parameters to the training data
best_lr.fit(X_train_scaled, y_train)

# Step 5: Evaluation
# Evaluate the model on test data
y_pred = best_lr.predict(X_test_scaled)
r2 = r2_score(y_test, y_pred)
print(f'R-squared Score: {r2}')
```

```
best_params {'fit_intercept': True, 'positive': False}
R-squared Score: 0.9966137419987549
```

[ ]: