Aayush Shah
19BCE245
1 May 2021

Practical 9

Linear regression

- **Definition :** Write a program to implement simple and multiple linear regression. Use the model parameters to make prediction for new data. Evaluate the model on standard evaluation measures. Your program should compute and display important model statistics (all the ones which are reported by statsmodels) and critically comment on these statistics.

- **Code :**
  - **Simple Linear Regression**

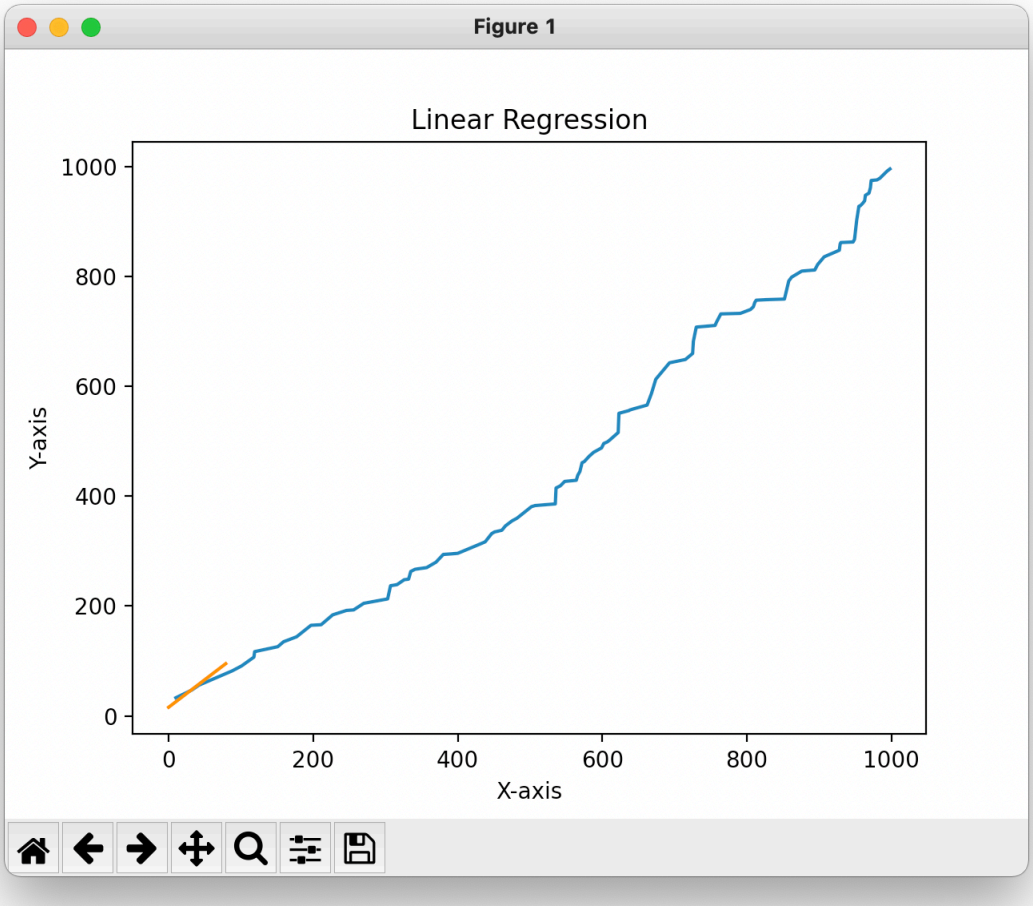
```
#!/usr/bin/env python3

import random
import matplotlib.pyplot as plt

#user input
size_of_data = int(input("Enter size of data you wanted : "))
starting_range = int(input("Enter starting limit of data : "))
ending_range = int(input("Enter ending limit of data : "))

#random data generating
independent_data =
random.sample(range(starting_range,ending_range),size_of_data)
dependent_data =
random.sample(range(starting_range,ending_range),size_of_data)

#sorting data
independent_data.sort()
dependent_data.sort()

x = independent_data
y = dependent_data
```

```
#calculating intercept and slope
n=len(x)
E_y=sum(y)
E_x=sum(x)
E_xy=sum([i*j for i,j in zip(x,y)])
E_x2=sum([i*i for i in x])
E_y2=sum([i*i for i in y])
intercept=(E_y*E_x2-E_x*E_xy)/(E_x2-E_x**2)
slope=(n*E_xy-E_x*E_y)/(n*E_x2-E_x**2)

#generating graph
x_val=list(range(80));
y_val=[intercept+slope*i for i in x_val];
print(f"\nIntercept = {round(intercept,2)}\nSlope =
{round(slope,2)}")
plt.plot(x,y)
plt.plot(x_val,y_val)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Linear Regression")
plt.show()
```

- **Sample I/O :**

```
Enter size of data you wanted : 100
Enter starting limit of data : 1
Enter ending limit of data : 1000

Intercept = 15.71
Slope = 1.0
```

- **Generated Graph :**



- **Multiple Linear Regression**

```
#on boston house-prices dataset
#Samples Total 506
#Dimensionality 13
#Features real, positive
#targets real
import numpy as np
from sklearn import datasets, linear_model, metrics
from sklearn.preprocessing import StandardScaler
# Load the boston dataset
X, y = datasets.load_boston(return_X_y=True)
X_train=X[0:400,:]
y_train=y[0:400]
X_test=X[400:506,:]
y_test=y[400:506]

```

```
print(X_train)
print(X_test)

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test=scaler.transform(X_test)

# Create linear regression object
regr =
  linear_model.SGDRegressor(max_iter=1000,learning_rate='const
  ant', eta0=0.01)

# Train the model using the training sets
regr.fit(X_train, y_train)
predictions=regr.predict(X_test)

print("Predictions :",predictions)

# Calculating the mean squared error of the prediction
mse = np.sum((predictions-y_test)**2)/len(y_test)
print(f'Mean Square Error : {mse}')

# Calculating the mean absolute error of the prediction
mae = np.sum(abs(predictions-y_test))/len(y_test)
print(f'Mean Absolute Error : {mae}')

print("Predicted Coefficients : ")

# Calculating the slope
for i in range(len(regr.coef_)):
    print(f'x{i+1} : {(regr.coef_)[i]}')

print(f'Predicted Intercept : {(regr.intercept_)[0]}')
```

- **Sample I/O :**

```
[[6.32000e-03 1.80000e+01 2.31000e+00 ... 1.53000e+01 3.96900e+02
  4.98000e+00]
 [2.73100e-02 0.00000e+00 7.07000e+00 ... 1.78000e+01 3.96900e+02
  9.14000e+00]
 [2.72900e-02 0.00000e+00 7.07000e+00 ... 1.78000e+01 3.92830e+02
  4.03000e+00]
 ...
```

...

```
Predictions : [15.39973066 21.91917666 22.40359965 16.3305633
    10.17980293 11.46469693
 10.17967746 23.65286405 16.54062672 23.62127872 18.14757829
    20.25744476
```

...

```
    27.08610038
 26.56773696 30.11068853 19.65948144 19.53006282 24.68588755
    14.42850297
 23.1258666  25.88602494 27.4429049  31.67099326 33.44514723
    24.57899122
 22.95310467 26.36491009 23.69053448 24.97576848 12.09668802
    8.12024726
  3.02824115 14.454009   16.72423774 20.17274638 20.11266642
    15.93373159
 13.21669587 18.9172974  21.16565039 17.98863125 20.46322271
    23.81461123
 22.40055793 28.57174142 26.91965471 22.44940593]
Mean Square Error : 36.2017819105814
Mean Absolute Error : 5.155303536458874
Predicted Coefficients :
x1 : -0.6683920245155992
x2 : 1.099755826411337
x3 : 0.5819253390429671
x4 : 0.589416098097974
x5 : -1.6779094099166818
x6 : 3.323290268510643
x7 : 0.35664291244947427
x8 : -2.77481020793202
x9 : 3.0825463439951513
x10 : -1.7275294645889065
x11 : -1.538601619181304
x12 : 0.4569558461354575
x13 : -3.855921650421727
Predicted Intercept : 24.173442899588068
```

Run Succeeded | Time 1 662 ms | Peak Memory 67.1M | Symbol | Spaces: 4 | Line 47, Column 55