CS353 Machine Learning Lab

Linear Regression (12/02/21)

Shumbul Arifa (181CO152)

Introduction

Linear Regression is a machine learning algorithm based on supervised learning. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

Dataset

Diabetes dataset

This dataset is available in scikit learn std dataset library.

Attributes:

1. age age in years
2. sex
3. bmi body mass index
4. bp average blood pressure
5. s1 tc, T-Cells (a type of white blood cells)
6. s2 ldl, low-density lipoproteins
7. s3 hdl, high-density lipoproteins
8. s4 tch, thyroid stimulating hormone
9. s5 ltg, lamotrigine
10. s6 glu, blood sugar level

Result

Model performance metrics are as below:

1. Mean-sq-error : 2634.3990247377064
2. Mean-abs-error : 41.03324080030628

3. R2-score : 0.5379603832310327
4. Mean-sq-log-error : 0.13730557437964364
5. Explained-variance-score : 0.5407959471638365

Implementation

Importing Python Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets,linear_model
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.metrics import mean_squared_log_error,explained_variance_score
from sklearn import metrics
```

Loading dataset

```
data = datasets.load_diabetes()
data
```

```
        -0.04687948,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452837, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00421986,  0.00306441]]),
 'data_filename': '/usr/local/lib/python3.6/dist-packages/sklearn/datasets/d
 'feature_names': ['age',
  'sex',
  'bmi',
  'bp',
  's1',
  's2',
  's3',
  's4',
  's5',
  's6'],
 'target': array([151.,  75., 141., 206., 135.,  97., 138.,  63., 110., 310.,
        101.,  69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
         68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
         87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
        259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
        128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
        150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
        200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
         42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
         83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
        104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
        173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
```

```
        173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
        107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
         60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
        197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
         59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
        237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
        143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
        142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
         77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
         78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
        154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
         71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
        150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
        145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
         94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
         60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
         31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
        114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
        191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
        244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
        263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
         77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
         58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
        140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
        219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
         43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
        140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
         84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
         94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
        220.,  57.]),
 'target_filename': '/usr/local/lib/python3.6/dist-packages/sklearn/datasets
```

Splitting dataset

```
X = data.data
y = data.target
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_stat
```

Buliding Model

```
model = linear_model.LinearRegression()
model.fit(x_train,y_train)
prediction = model.predict(x_test)
```

Analyzing

```
print('\n\nMetrics for model evualtion:\n\n')

mean_sq_error = mean_squared_error(y_test,prediction)
mean_abs_error = mean_absolute_error(y_test,prediction)
```

```
R2_score = metrics.r2_score(y_test,prediction)
mean_sq_log_error = mean_squared_log_error(y_test,prediction)
explained_variance_score = metrics.explained_variance_score( y_test,predictior

print(f' Mean-sq-error           : {mean_sq_error}\n\n Mean-abs-error


```

```
Metrics for model evualtion:


 Mean-sq-error            : 2634.3990247377064

 Mean-abs-error           : 41.03324080030628

 R2-score                 : 0.5379603832310327

 Mean-sq-log-error        : 0.13730557437964364

 Explained-variance-score : 0.5407959471638365
```

```
x_test
```

```
array([[ 0.0090156 , -0.04464164,  0.01427248, ..., -0.03949338,
        -0.03324879, -0.05906719],
       [-0.02730979, -0.04464164,  0.04768465, ...,  0.13025177,
         0.04506617,  0.13146972],
       [-0.00551455,  0.05068012, -0.00836158, ..., -0.00259226,
         0.08058546,  0.00720652],
       ...,
       [-0.00914709,  0.05068012, -0.03961813, ...,  0.07120998,
         0.01776348, -0.06735141],
       [ 0.06350368,  0.05068012, -0.00405033, ..., -0.00259226,
         0.08449528, -0.01764613],
       [-0.01277963, -0.04464164, -0.06548562, ..., -0.0070204 ,
        -0.03075121, -0.05078298]])
```

```
x_test.shape
```

```
(133, 10)
```

```
prediction.shape
```

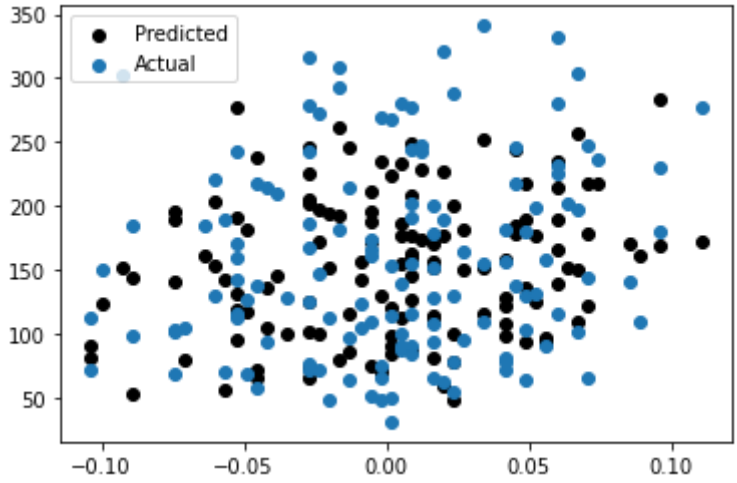
```
(133,)
```

Plotting Graph

```
  print("Plot on AGE")
  plt.scatter(x_test[:,0],prediction,color='black')
  plt.scatter(x_test[:,0],y_test)
  plt.legend(['Predicted','Actual'],loc =0 )
```

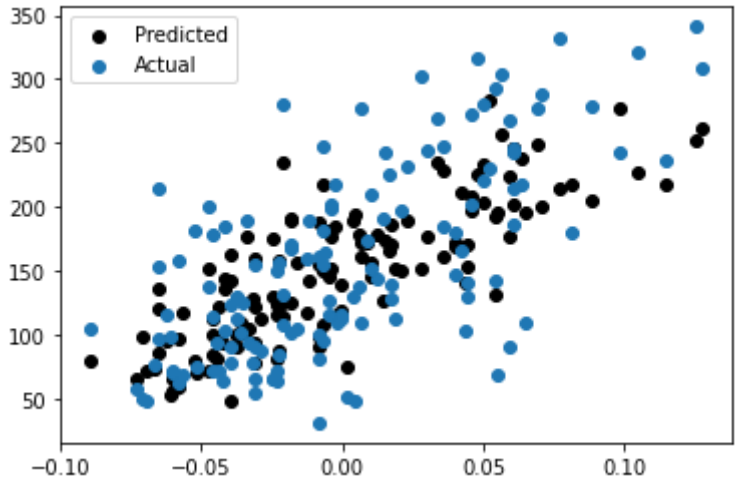
```
5     plt.show()
6     #age
7
```

Plot on AGE



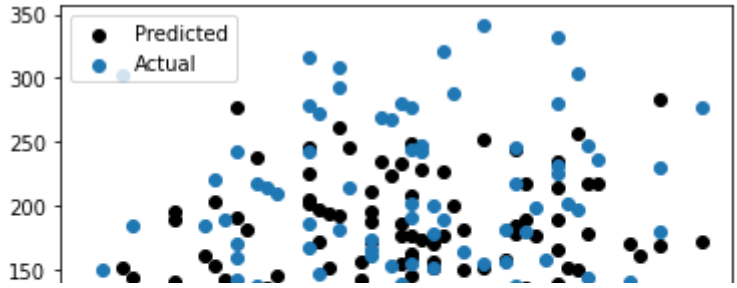

```
1  print("Plot on BMI")
2  plt.scatter(x_test[:,2],prediction,color='black')
3  plt.scatter(x_test[:,2],y_test)
4  plt.legend(['Predicted','Actual'],loc =0 )
5  plt.show()
6  #bmi
```

Plot on BMI



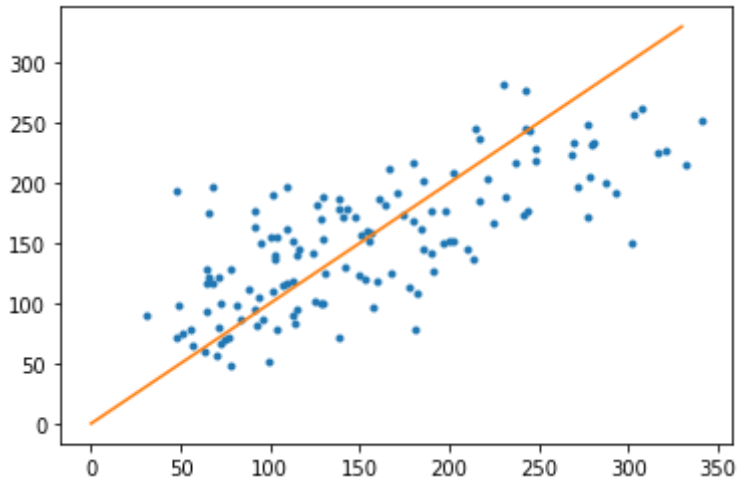

```
1  print("Plot on BP")
2  plt.scatter(x_test[:,0],prediction,color='black')
3  plt.scatter(x_test[:,0],y_test)
4  plt.legend(['Predicted','Actual'],loc =0 )
5  plt.show()
6  #bp
```

```
Plot on BP
```




```
y_pred = model.predict(x_test)
plt.plot(y_test, y_pred, '.')

# plot a line, a perfit predict would all fall on this line
x = np.linspace(0, 330, 100)
y = x
plt.plot(x, y)
plt.show()
```



Observation

By looking at the graphs, we can say that the performance is quite low when linear regression is used.

x-axis: original test values

y-axis: prediction values.

A perfect prediction would fall on the line x-y. As you can see, there are quite a lot of points not on the line y = x giving us low accuracy as a result.