Lab 5 - Linear Regression Diabetes Dataset

Author: Krishna Swaroop

181CO125, NITK Surathkal

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

We are using the [Diabetes] dataset. This dataset has ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of n = 442 diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.

There are a total of **10 columns** in this dataset, providing us a rich set of features to model our problems.

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

Linear Regression

1) Import libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import datasets
```

2) Load dataset

```
diabetes = datasets.load_diabetes() # load data
```

Explore data

```
print(f"Shape of the dataset is: {diabetes.data.shape}")
print(f"Shape of the target varialbe is: {diabetes.target.shape}")
print(f"Features of the diabetes dataset is: {diabetes.feature_names}")
```

```
Shape of the dataset is: (442, 10)
Shape of the target varialbe is: (442,)
Features of the diabetes dataset is: ['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

3) Split data

Use `train_test_split()` to split the data to training and testing dataset. Here, 20% of the dataset is reserved to test our algorithm

```
X_train, X_test, y_train, y_test = train_test_split(diabetes.data, diabetes.target, test_size=0.2, random_state=10)
```

4) Fit model

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

5) Predict

The model coefficients are

```
print(model.coef_)
```

```
[  -3.89155188 -225.62880027  517.89525355  328.32132183 -727.23345563
  410.96799392   80.26601137  218.18738355  704.2805541    40.02247238]
```

The model intercept is

```
print(model.intercept_)
```

```
152.2555053490769
```

```
model.predict(X_test)
print(f"Model performance is: {model.score(X_test, y_test)}")
```

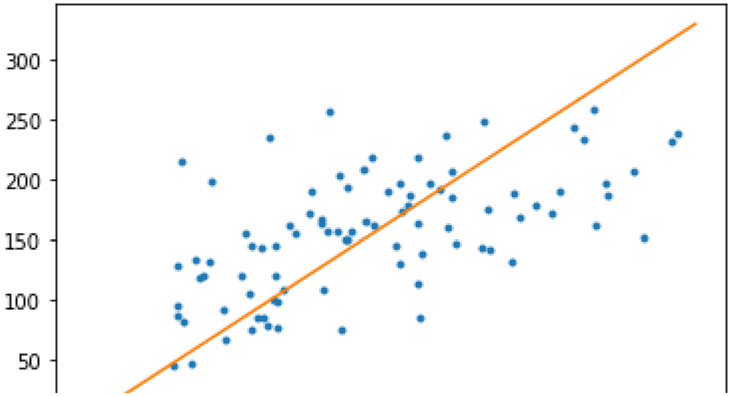
```
Model performance is: 0.5341962544929233
```

6) Analysis

Let us see why the model performance is low by plotting the fitting line

```
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()
```



Here, the x-axis represents the original test values wheras teh y-axis represents the 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 a low accuracy of 53%