student-score-prediction

July 2, 2021

0.1 Wreet Sarker

0.1.1 wreet.sarker@gmail.com

0.1.2 The Sparks Foundation

Task - 01

Score Prediction Using Supervised ML

[2]:
```
# At first let us import the required modules

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
```

[3]:
```
#Now let's import the data for this task

scores_df = pd.read_csv("student_scores.csv")
```

[4]:
```
# let's see what our data contains

scores_df.head()
```

[4]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

[5]:
```
scores_df.tail()
```

[5]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76

```
24    7.8      86
```

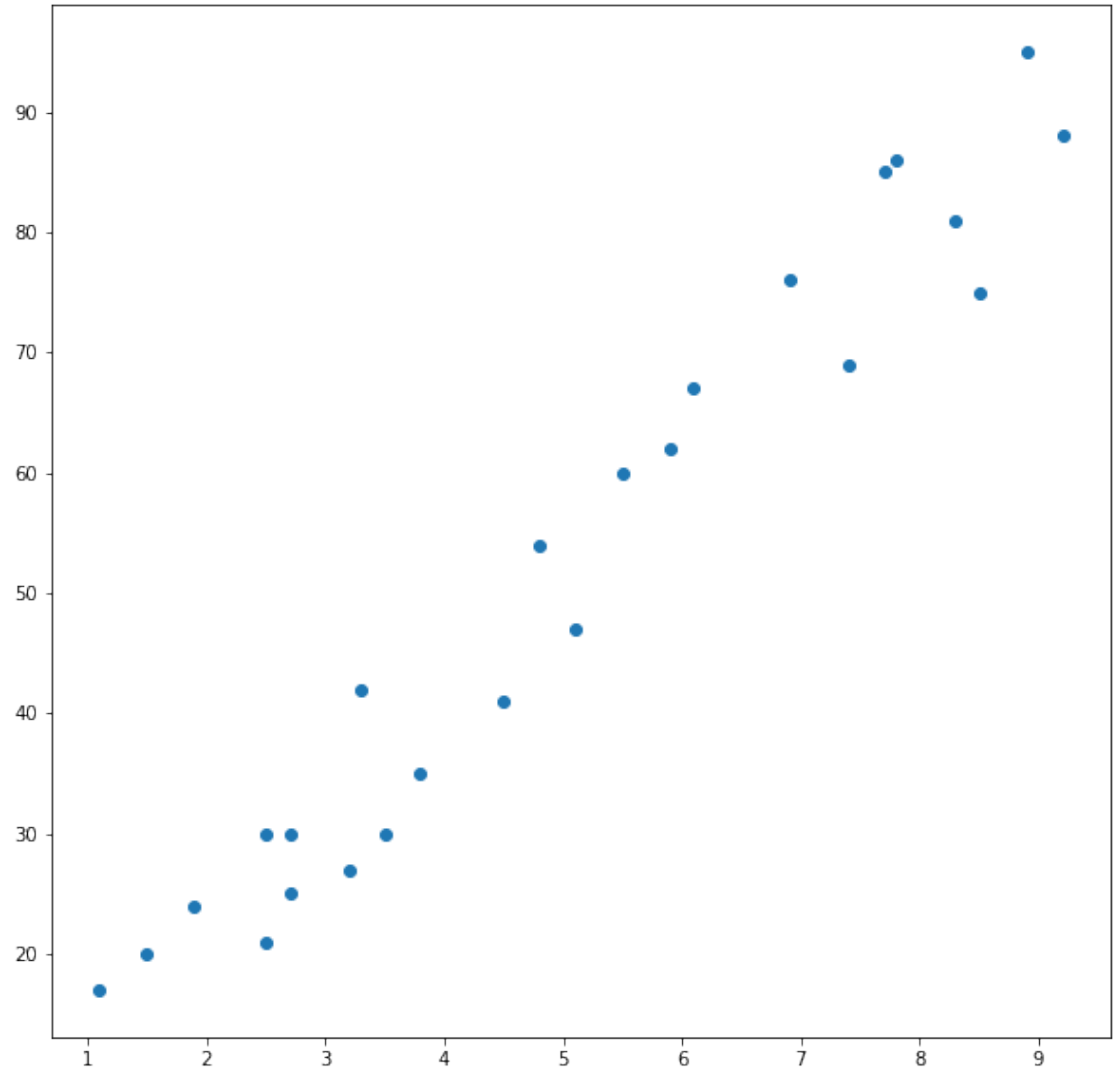
[7]:
```
scores_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

0.1.3 We can see our data contains two variables. We will be using our Hours column as our feature column to predict the score(label) of a student

This is a simple regression model. We will be using LinearRegression model for this problem

[23]:
```
# Let's see a line plot of our data
fig, ax = plt.subplots(figsize=(10,10))
ax.scatter(scores_df["Hours"], scores_df["Scores"]);
```



[31]:
```
# Now we will split the data into X and y and then into train and test splits

from sklearn.model_selection import train_test_split

X = scores_df.drop("Scores", axis=1)
y = scores_df["Scores"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[32]:
```
X_train
```

[32]:
```
    Hours
21    4.8
24    7.8
```

```
22    3.8
20    2.7
15    8.9
2     3.2
5     1.5
14    1.1
0     2.5
16    2.5
8     8.3
7     5.5
13    3.3
17    1.9
4     3.5
12    4.5
18    6.1
10    7.7
9     2.7
19    7.4
```

[33]:
```
y_train
```

[33]:
```
21    54
24    86
22    35
20    30
15    95
2     27
5     20
14    17
0     21
16    30
8     81
7     60
13    42
17    24
4     30
12    41
18    67
10    85
9     25
19    69
Name: Scores, dtype: int64
```

[34]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[34]:
```
((20, 1), (5, 1), (20,), (5,))
```

[35]:
```
## Now let's fit and score our model
from sklearn.linear_model import LinearRegression

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

model.score(X_train, y_train)
```

[35]: 0.9599941165428756

[36]:
```
# Let's check the score in the test data
model.score(X_test, y_test)
```

[36]: 0.7171348937516495

[37]:
```
# Our linear regression model is giving around 70% accurate predictions. Let's␣
 ↪check with another model
from sklearn.ensemble import RandomForestRegressor

reg = RandomForestRegressor()
reg.fit(X_train, y_train)

reg.score(X_train, y_train)
```

[37]: 0.9838174148224329

[38]:
```
# Random Forest Regressor model is giving a better result on the training set.␣
 ↪Let's see it's score on the test data

reg.score(X_test, y_test)
```

[38]: 0.8493943921408105

0.1.4 That's a great improvement! Our Random Forest Regressor model has given almost a 85% accuracy in the test data.

We will be using this model to predict the score of the student

[41]:
```
# In our problem we are given a time of 9.25 hours to predict the score of a␣
 ↪student.
#Let's see what is the predicted score for this given time

y_pred = reg.predict([[9.25]])

y_pred[0]
```

[41]: 90.02

0.1.5 So our Random Forest Regressor model Predicts a score of 90.2 for a student studying 9.25 hours

[ ]: