The Sparks Foundation - Data Science & Business Analytics Internship

TASK 1 - Prediction using Supervised Machine Learning

In this task it is required to predict the percentage of a student on the basis of number of hours studied using the Linear Regression supervised machine learning algorithm.

Author: Nabajyoti Nath

STEP 1 - Importing the dataset

In this step, we will import the dataset through the link provided by The Sparks Foundation with the help of pandas library and then we will observe the data

In [1]:
```
# Importing all the required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# To ignore the warnings
import warnings as wg
wg.filterwarnings("ignore")
```

In [2]:
```
# Reading data from remote link

url = "https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20student_scores.csv"
df = pd.read_csv(url)
```

In [3]:
```
# now let's observe the dataset
df.head()
```

Out[3]:

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

In [4]:
```
df.tail()
```

Out[4]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [5]:
```
# To find the number of columns and rows
df.shape
```

Out[5]:
```
(25, 2)
```

In [6]:
```
df.describe()
```

Out[6]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [7]:
```
# To find more information about our dataset
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
```

In [8]:
```
# now we will check if our dataset contains null or missings values
df.isnull().sum()
```

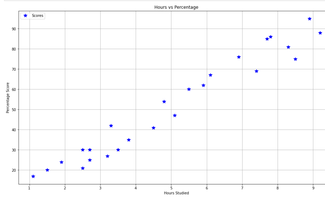
Out[8]:
```
Hours     0
Scores    0
dtype: int64
```

As we can see we do not have any null values in our data set so we can now move on to our next step

STEP 2 - Visualizing the dataset

In this we will plot the dataset to check whether we can observe any relation between the two variables or not

In [9]:
```
# Plotting the dataset
plt.rcParams["figure.figsize"] = [16,9]
df.plot(x='Hours', y='Scores', style='*', color='blue', markersize=10)
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



From the graph above, we can observe that there is a linear relationship between "hours studied" and "percentage score". So, we can use the linear regression supervised machine model on it to predict further values.

In [10]:
```
# we can also use .corr to determine the corelation between the variables
df.corr()
```

Out[10]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

STEP 3 - Data preparation

In this step we will divide the data into "features" (inputs) and "labels" (outputs). After that we will split the whole dataset into 2 parts - testing data and training the data

In [11]:
```
df.head()
```

Out[11]:

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

In [12]:
```
# using iloc function we will divide the data
X = df.iloc[:, :1].values
y = df.iloc[:, 1:].values
```

In [13]:
```
X
```

Out[13]:
```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
       [8.3],
       [2.7],
       [7.7],
       [5.9],
       [4.5],
       [3.3],
       [1.1],
       [8.9],
       [2.5],
       [1.9],
       [6.1],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

In [14]:
```
# Splitting data into training and testing data

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                            test_size=0.2, random_state=0)
```

STEP 4 - Training the Algorithm

We have splited our data into training and testing sets, and now we will train our Model.

In [15]:
```
from sklearn.linear_model import LinearRegression

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

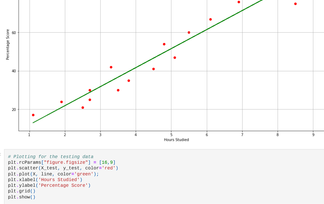
Out[15]:
```
LinearRegression()
```

STEP 5 - Visualizing the model

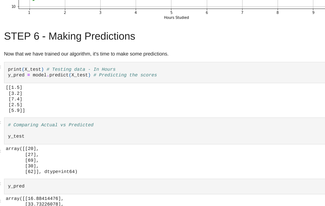
After training the model, now its time to visualize it.

In [16]:
```
line = model.coef_*X + model.intercept_

# Plotting for the training data
plt.rcParams["figure.figsize"] = [16,9]
plt.scatter(X_train, y_train, color='red')
plt.plot(X, line, color='green');
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



In [17]:
```
# Plotting for the testing data
plt.rcParams["figure.figsize"] = [16,9]
plt.scatter(X_test, y_test, color='red')
plt.plot(X, line, color='green');
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



STEP 6 - Making Predictions

Now that we have trained our algorithm, it's time to make some predictions.

In [18]:
```
print(X_test) # Testing data - In Hours
y_pred = model.predict(X_test) # Predicting the scores
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

In [19]:
```
# Comparing Actual vs Predicted

y_test
```

Out[19]:
```
array([[20],
       [27],
       [69],
       [30],
       [62]], dtype=int64)
```

In [20]:
```
y_pred
```

Out[20]:
```
array([[16.884144763],
       [33.732260793],
       [75.357018033],
       [26.794801242],
       [60.491033283]])
```

In [21]:
```
# Comparing Actual vs Predicted
comp = pd.DataFrame({ 'Actual':[y_test],'Predicted':[y_pred] })
comp
```

Out[21]:

	Actual	Predicted
0	[[20], [27], [69], [30], [62]]	[[16.884144763279903], [33.732260779489846], [7...

In [22]:
```
# Testing with your own data

hours = 9.25
own_pred = model.predict([[hours]])
print("The predicted score if a person studies for",hours,"hours is",own_pred[0])
```

```
The predicted score if a person studies for 9.25 hours is [93.69173249]
```

STEP 7 - Evaluating the model

In the last step, we are going to evaluate our trained model by calculating mean absolute error

In [23]:
```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```