Task 1: Prediction using supervised Machine Learning

Predict the percentage of a student based on the no. of study hours.

Importing Required Libraries

```
#importing require libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

Read Data From Dataset

```
data = pd.read_csv("http://bit.ly/w-data")
```

Explore The Data

```
# Check the shape of the dataset
data.head()
```

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

```
data.describe()
data.dtypes
```

```
Hours      float64
Scores       int64
dtype: object
```

```
data.info()
```

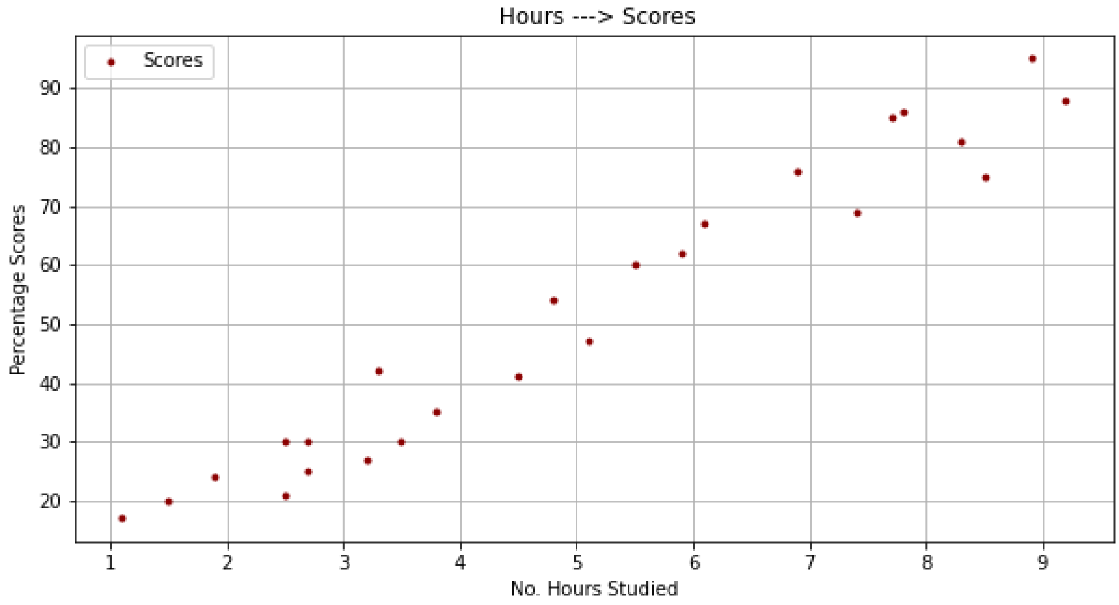
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0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

```
# Check missing value
data.isnull().sum()
```

```
Hours     0
Scores    0
dtype: int64
```

Step 2: Visualizing The Dataset

```
data.plot(x='Hours',y='Scores',style=".",color="darkred",figsize=(10,5))
plt.title(' Hours ---> Scores')
plt.xlabel('No. Hours Studied')
plt.ylabel('Percentage Scores')
plt.grid()
plt.show()
```



```
# Check the correlation between Hours and Scores
data.corr()
```

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

This shows the positive correlation between Hours and Scores

Step 3: Data Preparation

```
#divide the data by iloc function
X = data.iloc[:,:1].values
Y = data.iloc[:,1:].values
```

```
X
```

```
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]])
```

```
Y
```

```
array([[21],
       [47],
       [27],
       [75],
       [30],
       [20],
       [88],
       [60],
       [81],
       [25],
       [85],
       [62],
           [41],
           [42],
           [17],
           [95],
           [30],
           [24],
           [67],
           [69],
           [30],
           [54],
           [35],
           [76],
           [86]])
```

```
#split the data into training and split the data
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

Step 4: Apply : Simple Linear Algorithm

```
#create linear regression instance
reg_model = LinearRegression()
reg_model.fit(X_train,Y_train)
```

```
    LinearRegression()
```

```
#find the equation of the fit line

# find the slop
Slop = reg_model.coef_

# find the intercept
Intercept = reg_model.intercept_

print("Slop of the fit line is : ",float(Slop))

print("Intercept of the fit line is : ",float(Intercept))
```

```
    Slop of the fit line is :  9.91065648064224
    Intercept of the fit line is :  2.018160041434662
```

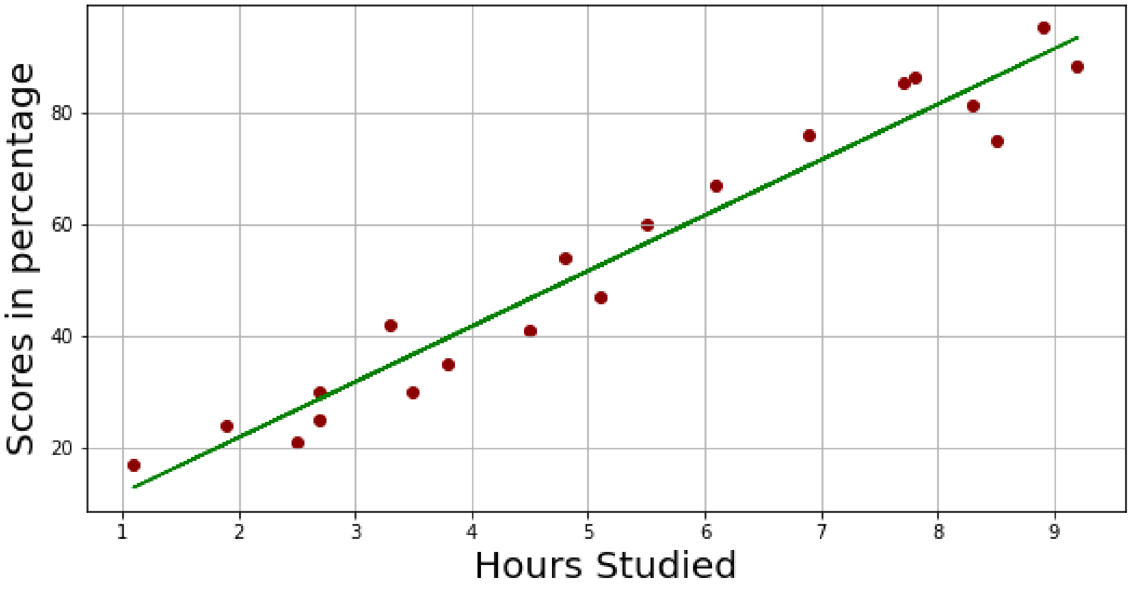
```
# Equation is
print(" Y = {:.4f} * X + {:.4f} ".format(float(Slop),float(Intercept)))
```

```
     Y = 9.9107 * X + 2.0182
```

Step 5: Visualizing The Model

```
#ploting for the traing data

line = Slop * X + Intercept
plt.rcParams["figure.figsize"]=[10,5]
plt.scatter(X_train,Y_train,color="darkred")
plt.plot(X,line,color="green")
plt.xlabel('Hours Studied',fontsize=20)
plt.ylabel('Scores in percentage',fontsize=20)
plt.grid()
plt.show()
```



Step 6: Making Prediction

```
#Testing Data
print(X_test)
#Predicted Scores
Y_pred = reg_model.predict(X_test)
```

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

```
#Compair Actual and Predicted
print(f"Acutal :\n {Y_test}",end="\n")
print(f"Predict :\n {Y_pred}")
```

```
Acutal :
 [[20]
 [27]
 [69]
 [30]
 [62]]
Predict :
 [[16.88414476]
 [33.73226078]
 [75.357018  ]
 [26.79480124]
 [60.49103328]]
```

Predict The Score for 9.25 Hours/Day

```
X = 9.25
predicted_score = reg_model.predict([[X]])
```

```
print(float(predicted_score))
```

```
93.69173248737539
```

Predicted Score For 9.25 Hours/Day is : 93.6917324%

Step 7: Evaluating The Data

```
#find mean absolute and squared error
from sklearn import metrics
print(f"Mean Absolute Error is : {metrics.mean_absolute_error(Y_test,Y_pred)}")
print(f"Mean Squared Error is : {metrics.mean_squared_error(Y_test,Y_pred)}")
```

```
Mean Absolute Error is : 4.183859899002982
Mean Squared Error is : 21.598769307217456
```

Colab paid products - Cancel contracts here

✓ 0s completed at 6:10 PM ● ×