THE SPARKS FOUNDATION

Batch-September2022

Name: Varsha Reddy

Task 1 - Prediction Using Supervised ML

Statement of the Problem

Determine a student's percentage based on the number of study hours they have completed.

Dataset: http://bit.ly/w-data

Importing Libraries for Data Manipulation

In [1]:
```
import numpy as np
import pandas as pd
```

Importing Libraries for Data Visualization

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Importing the Dataset

In [3]:
```
df= pd.read_csv("Data.csv")
```

Processing the Data

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

Out[4]:

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

In [5]:
```
df.tail()
```

Out[5]:

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

In [6]:
```
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 [7]:
```
df.describe()
```

Out[7]:

	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 [8]:
```
df.shape
```

Out[8]:
```
(25, 2)
```

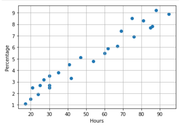
In [9]:
```
df.columns
```

Out[9]:
```
Index(['Hours', 'Scores'], dtype='object')
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
Hours     0
Scores    0
dtype: int64
```

In [11]:
```
plt.scatter(df['Scores'],df['Hours'])
plt.xlabel('Hours')
plt.ylabel('Percentage')
plt.style.use('dark_background')
plt.grid()
plt.show()
```



Creating the matrix of features

In [12]:
```
x= df.iloc[:,:-1].values
```

Creating the dependent variable vector

In [13]:
```
y= df.iloc[:,1].values
```

Splitting Data into Training and Testing datasets

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state=7)
```

Linear Regression

In [16]:
```
from sklearn.linear_model import LinearRegression
```

In [17]:
```
LR =  LinearRegression()
```

In [18]:
```
LR.fit(x_train,y_train)
```

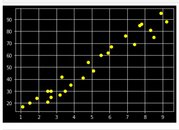
Out[18]:
```
LinearRegression()
```

In [19]:
```
print(x_test)
y_pred = LR.predict(x_test)
```

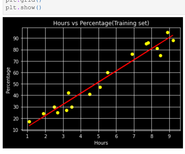
```
[[2.7]
 [2.5]
 [6.1]
 [7.4]
 [1.5]
 [5.9]
 [4.8]
 [3.8]]
```

Visualizing the Data

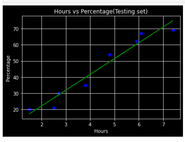
In [20]:
```
plt.scatter(x,y,color='yellow')
plt.style.use('dark_background')
plt.grid()
plt.show()
```



In [21]:
```
plt.scatter(x_train,y_train,color='yellow')
plt.plot(x_train,LR.predict(x_train),color ='red')
plt.title("Hours vs Percentage(Training set)")
plt.xlabel("Hours")
plt.ylabel("Percentage")
plt.style.use('dark_background')
plt.grid()
plt.show()
```



In [22]:
```
plt.scatter(x_test,y_test,color='blue')
plt.plot(x_test,LR.predict(x_test),color ='green')
plt.title("Hours vs Percentage(Testing set)")
plt.xlabel("Hours")
plt.ylabel("Percentage")
plt.style.use('dark_background')
plt.grid()
plt.show()
```



In [23]:
```
print(LR.coef_)
```

```
[9.77334064]
```

In [24]:
```
print(LR.intercept_)
```

```
2.560902255639397
```

Predicted and Actual Scores

In [25]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[25]:

	Actual	Predicted
0	30	28.948922
1	21	26.994254
2	67	62.178280
3	69	74.883623
4	20	17.220913
5	62	60.223612
6	54	49.472937
7	35	39.699597

In [26]:
```
LR.score(x_train,y_train)
```

Out[26]:
```
0.9516091323831719
```

In [27]:
```
LR.score(x_test,y_test)
```

Out[27]:
```
0.9498016481178696
```

Analyzing 9.25 hours/day of study in order to predict the score

In [28]:
```
hours = np.array([[9.25]])
pred = LR.predict(hours)
print('No. of hours = {}'.format(hours[0][0]))
print('Predicted = %.2f'%pred)
```

```
No. of hours = 9.25
Predicted = 92.96
```

The Model's Evaluation

In [29]:
```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 3.9416011054125395
```

In [30]:
```
from sklearn import metrics
print('Mean Squared Error:',
      metrics.mean_squared_error(y_test, y_pred))
```

```
Mean Squared Error: 18.545153623454564
```

In [31]:
```
from sklearn import metrics
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
RMSE: 4.306409843673717
```