The Sparks Foundation - Graduate Rotational Internship Program

Name : Aakifah Fatima

Task-1 : Prediction using Supervised ML

Description: Predict the percentage of a student based on the no. of study hours. This is a simple linear regression task as it involves just 2 variables. What will be predicted score if a student studies for 9.25 hrs/ day?

Importing the initial libraries required.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
Matplotlib is building the font cache; this may take a moment.
```

Reading the data from the csv file and taking a look at it.

In [2]:
```
link = 'http://bit.ly/w-data'
df = pd.read_csv(link)
```

In [3]:
```
df
```

Out[3]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67
19	7.4	69
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [4]:
```
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 [5]:
```
df.describe()
```

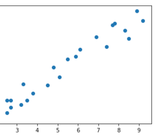
Out[5]:

	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

Plotting hours and scores.

In [6]:
```
plt.scatter(x=df['Hours'], y=df['Scores'])
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x1b3976f9c10>
```



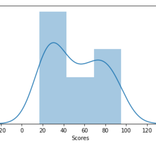
A linear relationship seems to be present.

Checking the distribution of the target variable, scores.

In [7]:
```
sns.distplot(df.loc[:,'Scores'], norm_hist=True)
```

```
c:\users\user\appdata\local\programs\python\python39\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[7]:
```
<AxesSubplot:xlabel='Scores', ylabel='Density'>
```



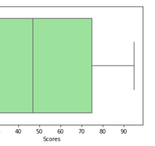
A somewhat normal distribution is observed.

Checking the presence of outliers.

In [8]:
```
sns.boxplot(df.loc[:, 'Scores'], color='lightgreen')
```

```
c:\users\user\appdata\local\programs\python\python39\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]:
```
<AxesSubplot:xlabel='Scores'>
```



No outliers are observed.

Building the supervised machine learning model:

Splitting the data into train and test sets.

In [13]:
```
from sklearn.model_selection import train_test_split

X = df.drop('Scores', axis=1)
y = df['Scores']
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
```

Linear Regression.

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

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

print('linear regression model coeff (w):', linreg.coef_)
print('linear regression model intercept (b):', linreg.intercept_)
```

```
linear regression model coeff (w): [9.94167834]
linear regression model intercept (b): 1.9322042531516446
```

Evaluating the model.

In [16]:
```
from sklearn.metrics import mean_squared_error

y_pred = linreg.predict(X_train)
rmse = mean_squared_error(y_train, y_pred)
print(rmse)

y_predicted = linreg.predict(X_test)
rmse = mean_squared_error(y_test, y_predicted)
print(rmse)
```

```
32.555077067504286
20.33292367497997
```

In [17]:
```
from sklearn.metrics import r2_score

print('Training r2 score:', r2_score(y_train, y_pred))
print('Testing r2 score:', r2_score(y_test, y_predicted))
```

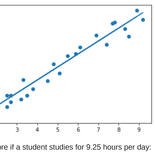
```
Training r2 score: 0.9484509249326872
Testing r2 score: 0.9367661043365055
```

Visualising the fitted model.

In [18]:
```
line = linreg.coef_*df['Hours'] + linreg.intercept_
```

In [19]:
```
plt.scatter(x=df['Hours'], y=df['Scores'])
plt.plot(df['Hours'], line)
```

Out[19]:
```
[<matplotlib.lines.Line2D at 0x1b39d11ebb0>]
```



Predicted score if a student studies for 9.25 hours per day:

In [20]:
```
linreg.predict([[9.25]])
```

Out[20]:
```
array([93.89272889])
```

The predicted score for a student who studies for 9.25 hours a day is 93.89%