Name: PAULOMI NANDI

The Sparks Foundation- "Data Science & Business Analytics Internship"

GRIP: JUNE-2022

Task-1: Prediction Using Supervised Machine Learning

In [33]:
```
# Importing the required liabaries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
```

In [4]:
```
# Reading data from remote link
url_data = "http://bit.ly/w-data"
sl_data = pd.read_csv(url_data)
print("Data imported successfully.")
sl_data.head(10)
```

Data imported successfully.

Out[4]:

	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

In [7]:
```
sl_data.sample(10)
```

Out[7]:

	Hours	Scores
16	2.5	30
4	3.5	30
12	4.5	41
6	9.2	88
23	6.9	76
18	6.1	67
7	5.5	60
5	1.5	20
9	2.7	25
1	5.1	47

In [8]:
```
sl_data.columns
```

Out[8]:
```
Index(['Hours', 'Scores'], dtype='object')
```

In [9]:
```
sl_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
```

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

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

In [11]:
```
sl_data.describe()
```

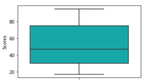
Out[11]:

	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

Data Visualization

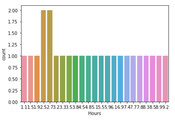
In [14]:
```
plt.figure(figsize=(5,3))
sns.boxplot(data=sl_data,y='Scores',color='c')
```

Out[14]:
```
<AxesSubplot:ylabel='Scores'>
```



In [15]:
```
sns.countplot(data=sl_data, x='Hours')
```

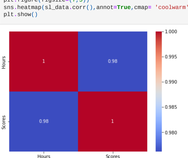
Out[15]:
```
<AxesSubplot:xlabel='Hours', ylabel='count'>
```



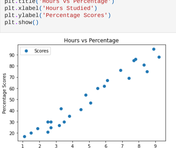
In [16]:
```
sns.countplot(data=sl_data, x='Scores')
```

Out[16]:
```
<AxesSubplot:xlabel='Scores', ylabel='count'>
```

In [17]:
```
plt.figure(figsize=(7,5))
sns.heatmap(sl_data.corr(),annot=True,cmap= 'coolwarm')
plt.show()
```



In [18]:
```
# Plotting the distribution of the scores
sl_data.plot(x='Hours', y='Scores',style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Scores')
plt.show()
```



Data Processing

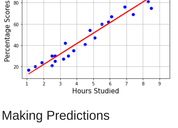
In [19]:
```
x=sl_data.iloc[:,:-1].values
y=sl_data.iloc[:, 1].values
```

In [20]:
```
x_train, x_test,y_train, y_test= train_test_split(x,y,test_size=0.3,random_state= 0)
regressor= LinearRegression()
regressor.fit(x_train.reshape(-1,1), y_train)
print("Training Completed")
```

```
Training Completed
```

In [22]:
```
# Plotting the regression line
line= regressor.coef_*x+regressor.intercept_

# Plotting for the test data
plt.scatter(x,y, color='blue')
plt.plot(x,line,color='red')
plt.title('Hours vs Percentage', size=20)
plt.xlabel('Hours Studied', size=15)
plt.ylabel('Percentage Scores', size=15)
plt.grid()
plt.show()
```



Making Predictions

In [23]:
```
# testing data
print(x_test)

# Model Prediction
y_pred= regressor.predict(x_test)
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]
 [3.8]
 [1.9]
 [7.8]]
```

In [24]:
```
y_test
```

Out[24]:
```
array([20, 27, 69, 30, 62, 35, 24, 86], dtype=int64)
```

In [25]:
```
y_pred
```

Out[25]:
```
array([17.053665, 33.694229, 74.806209, 26.842232, 60.123359, 39.567369,
       20.969092, 78.721636])
```

In [27]:
```
# Comparing Actual vs Predicted
df= pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[27]:

	Actual	Predicted
0	20	17.053665
1	27	33.694229
2	69	74.806209
3	30	26.842232
4	62	60.123359
5	35	39.567369
6	24	20.969092
7	86	78.721636

In [28]:
```
# Now testing the given test data  9.25 hrs
hours =9.25
own_pred= regressor.predict([[hours]])
print("No of Hours= {}".format(hours))
print("Predicted Scores = {}".format(own_pred[0]))
```

```
No of Hours= 9.25
Predicted Scores = 92.91505723477056
```

Evaluating the Model

In [29]:
```
from sklearn import metrics
print ('Mean Absolute Error:', metrics.mean_absolute_error(y_test,y_pred))
print ('Mean Squared Error:', metrics.mean_squared_error(y_test,y_pred))
print ('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print('R2:', metrics.r2_score(y_test, y_pred))
```

```
Mean Absolute Error: 4.419727808027651
Mean Squared Error: 22.96509721270428
Root Mean Squared Error: 4.792191274636144
R2: 0.9568211804435257
```

I was successfully able to carry-out prediction using supervised ML task and was able to evaluate the model's preformances on various parameters.

Thank you.