The Sparks Foundation

Function: Data Science and Business Analytics

Task 1- Prediction Using Supervised ML

Submitted by: Kumari Soni (July 2021 Batch)

In [1]:
```
# Import all the required libraries
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import os
import seaborn as sns
```

In [2]:
```
# Load the dataset
data = pd.read_csv("http://bit.ly/w-data")

# Print top 5 values of the dataset to get idea about the dataset
data.head()
```

Out[2]:

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

In [3]:
```
# Check if there is any null value in the Dataset
data.isnull().sum()

# We observe that there is no null value present
```

Out[3]:
```
Hours     0
Scores    0
dtype: int64
```

In [4]:
```
# Print the information of the dataset to get ingights of the dataset
data.info()
```

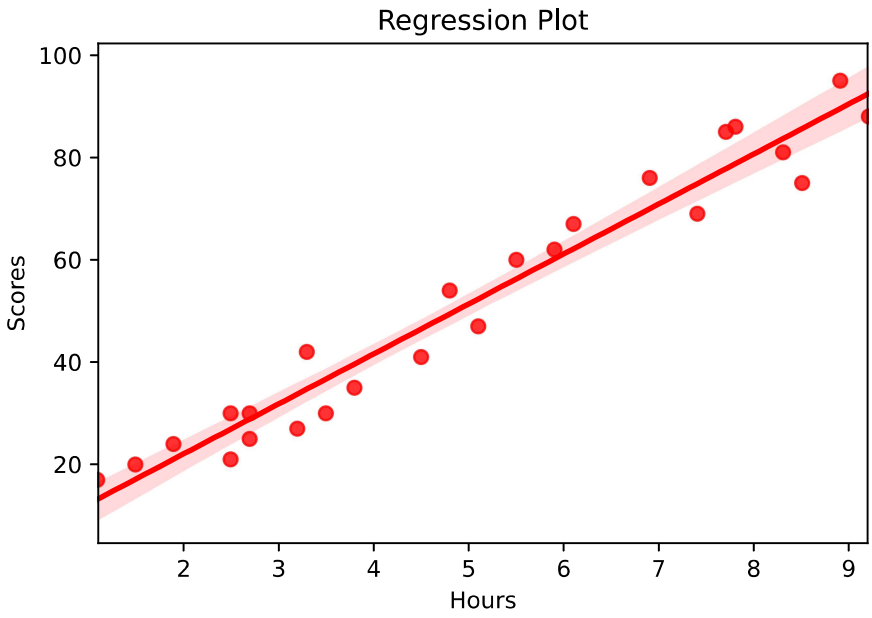
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes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

In [5]:
```
# Visualizing the data
sns.regplot(x= data['Hours'], y= data['Scores'], color= "red")

plt.title('Regression Plot')

plt.ylabel('Scores')
plt.xlabel('Hours')

plt.show()
```

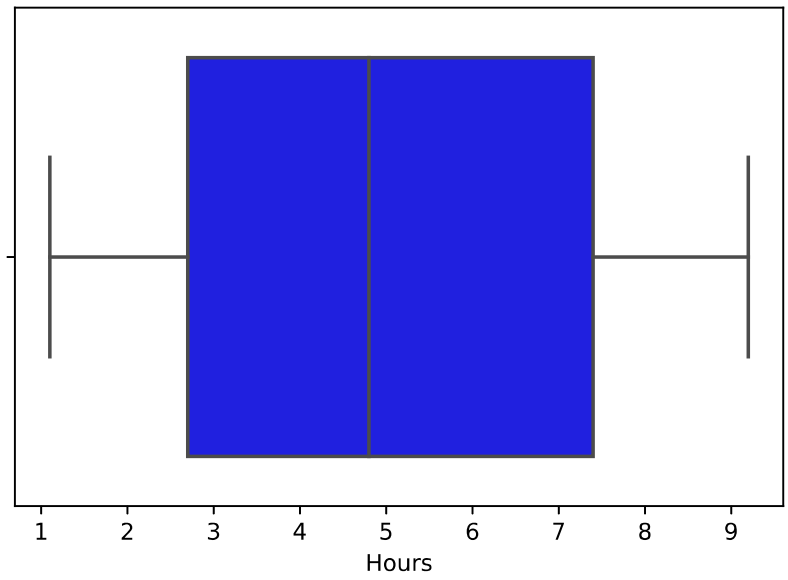


In [6]:
```
# Find the correlation between the attributes
print(data.corr())
```

```
            Hours    Scores
Hours    1.000000  0.976191
Scores   0.976191  1.000000
```

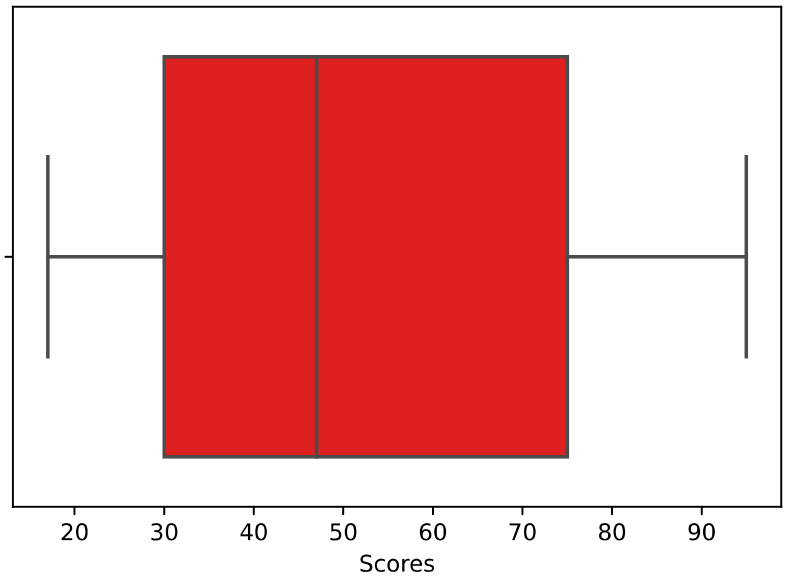
In [7]:
```
#Box plot- Hours
sns.boxplot(x= data['Hours'], color= "blue")
```

Out[7]: <AxesSubplot:xlabel='Hours'>



In [8]:
```
#Box plot- Scores
sns.boxplot(x= data['Scores'], color= "red")
```

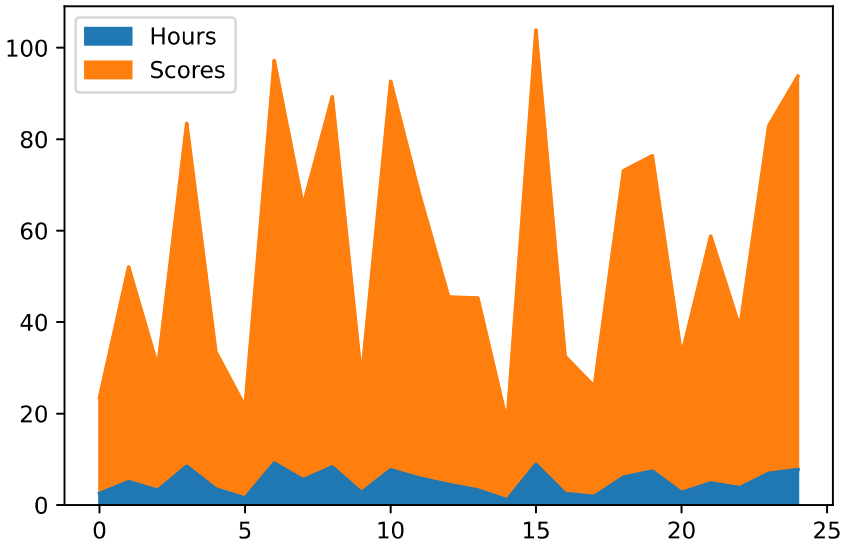
Out[8]: <AxesSubplot:xlabel='Scores'>

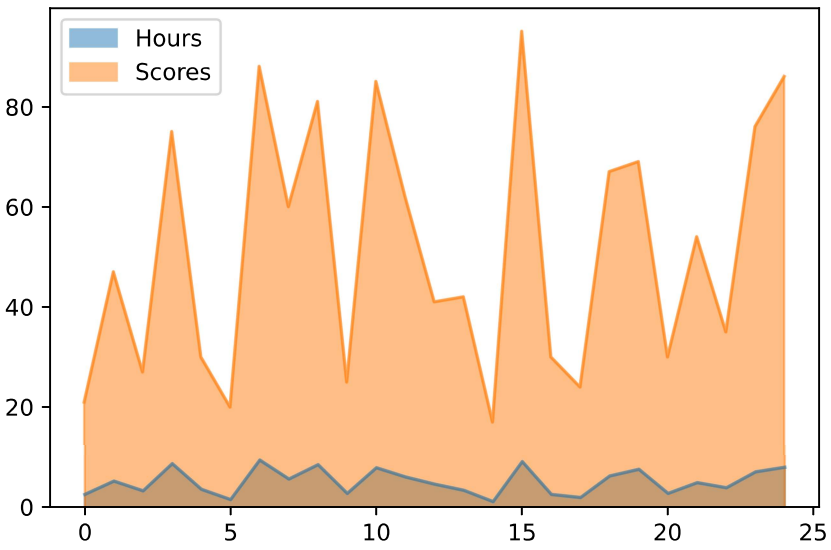


In [9]:
```
#Area plot over the dataset
data.plot.area()

#Area plots are stacked by default. To produce an unstacked plot, pass stacked=False:
data.plot.area(stacked= False)
```

Out[9]: <AxesSubplot:>





In [10]:

```
# Splitting the dataset into the Training set and Test set in the ratio 80:20
x = data["Hours"].values.reshape(-1,1)
y = data["Scores"].values.reshape(-1,1)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=0)
```

In [11]:

```
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[11]:

```
LinearRegression()
```

In [12]:

```
# Predicting the Test set results
y_pred = regressor.predict(x_test)

# Print the predicted values
y_pred
```

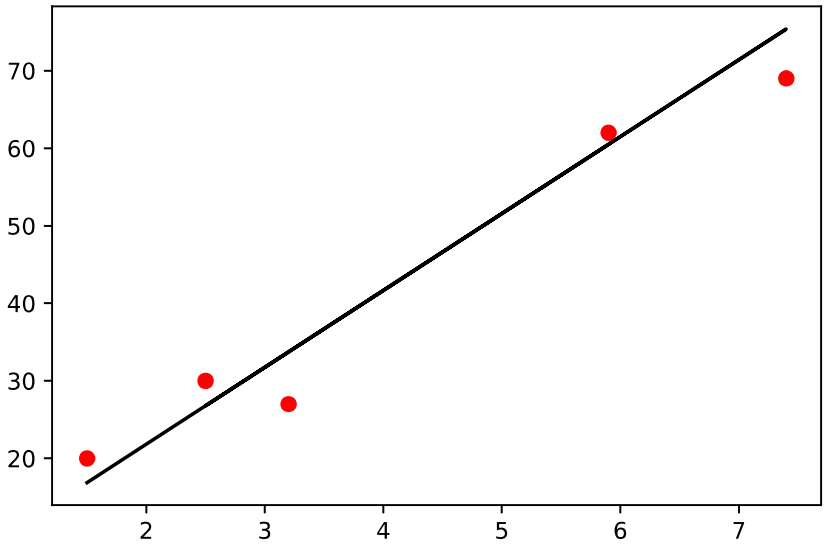
Out[12]:

```
array([[16.88414476],
       [33.73226078],
       [75.357018  ],
       [26.79480124],
       [60.49103328]])
```

In [13]:

```
# Visualizing Test Data
plt.scatter(x_test,y_test , color = "red")
plt.plot(x_test,y_pred , color = "black")

plt.show()
```



In [14]:

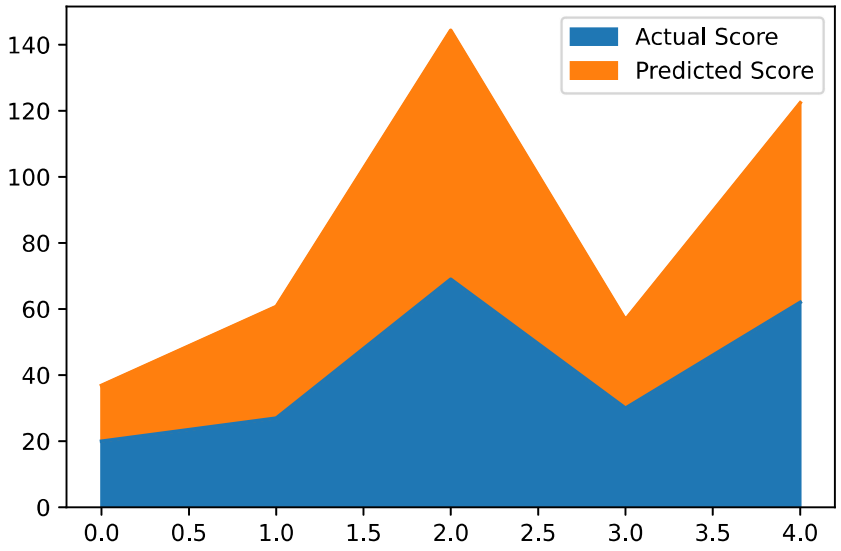
```
# Print actual and predicted scores
data1= pd.DataFrame({"Hours": x_test.reshape(1,-1)[0] , "Actual Score" : y_test.reshape(1,-1)[0] , "
data1
```

Out[14]:

	Hours	Actual Score	Predicted Score
0	1.5	20	16.884145
1	3.2	27	33.732261
2	7.4	69	75.357018
3	2.5	30	26.794801
4	5.9	62	60.491033

In [15]:
```
# Visualising actual vs predicted scores
data1.iloc[:, 1:3].plot.area()
```

Out[15]: <AxesSubplot:>



In [16]:
```
# Prediction: What will be predicted score if a student studies for 9.25 hrs/day?
hours=[[9.25]]
result= regressor.predict(hours)

print("The predicted score of the student who studies for 9.25 hrs/ day is:" , result[0,0])
```

```
The predicted score of the student who studies for 9.25 hrs/ day is: 93.69173248737535
```
