GRIP : The Spark Foundation

Data Science and Business Analytics Intern

Author : Purba Chakraborty

Task 1: Prediction Using Supervised ML

In this regression task we will predict the percentage of marks that a student is expected to score based upon the number of hours they studied. This is a simple linear regression task as it involves just two variables. We have to also predict the score if the student studies 9.25hrs/day .

Importing required libraries

In [27]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing the data set

In [4]:
```
task1=pd.read_csv("task1.csv")
```

Exploring Data

In [15]:
```
print(task1.shape)
task1.head()
```

```
(25, 2)
```

Out[15]:

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

In [16]:
```
task1.describe()
```

Out[16]:

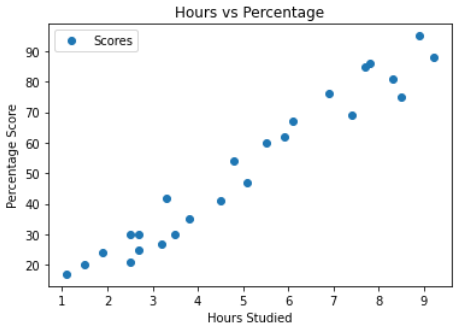
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [20]:
```
task1.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
```

Plotting the distribution of the scores

In [21]:
```
task1.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



In [22]:
```
task1.corr(method='pearson')
```

Out[22]:

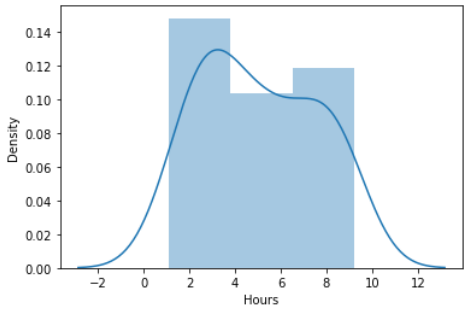
	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [24]:
```
hours=task1['Hours']
scores=task1['Scores']
```

In [25]:
```
sns.distplot(hours)
```

```
D:\D\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pleas
e 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)
```

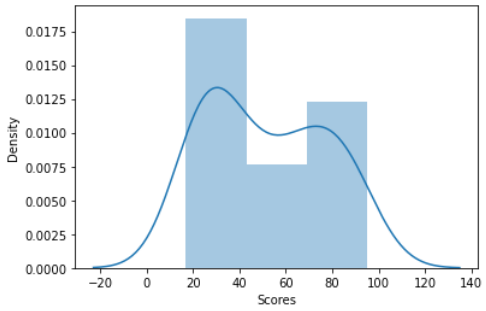
Out[25]: <AxesSubplot:xlabel='Hours', ylabel='Density'>



In [26]:
```
sns.distplot(scores)
```

```
D:\D\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pleas
e 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[26]: <AxesSubplot:xlabel='Scores', ylabel='Density'>



Linear Regression

we create two separate arrays,

one that contains our independent variables (also called the input features), and another one that contains our dependent variable (what we want to predict).

In [17]:
```
X = task1.iloc[:, :-1].values
y = task1.iloc[:, 1].values
```

In [9]:
```
print(X)
```

```
[[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]]
```

In [18]:
```
print(y)
```

```
[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]
```

Splitting the dataset into the Training set and Test set

We'll do this by using Scikit-Learn's built-in train_test_split() method:

In [29]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                test_size=0.2, random_state=0)
```

In [30]:
```
print(X_train)
```

```
[[3.8]
 [1.9]
 [7.8]
 [6.9]
 [1.1]
 [5.1]
 [7.7]
 [3.3]
 [8.3]
 [9.2]
 [6.1]
 [3.5]
 [2.7]
 [5.5]
 [2.7]
 [8.5]
 [2.5]
 [4.8]
 [8.9]
 [4.5]]
```

In [31]:
```
print(X_test)
```

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

Training the Simple Linear Regression model on the Training set

In [32]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

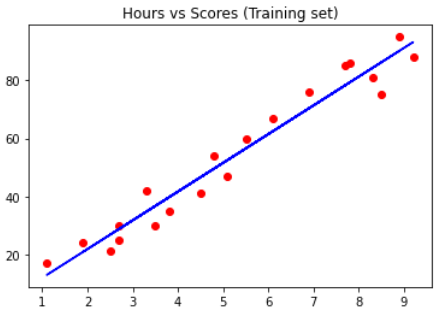
Out[32]:
```
LinearRegression()
```

Predicting the Test set results

In [33]:
```
y_pred = regressor.predict(X_test)
```

Visualising the Training set results

In [34]:
```
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Hours vs Scores (Training set)')
plt.show()
```



Visualising the Test set results

In [35]:
```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.show()
```

Comparing Actual vs Predicted

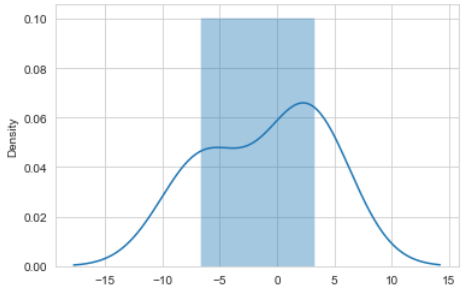
In [36]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[36]:

	Actual	Predicted
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

In [37]:
```
sns.set_style('whitegrid')
sns.distplot(np.array(y_test-y_pred))
plt.show()
```

```
D:\D\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pleas
e 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)
```



Predict the score if a student studies for 9.25hours/day

In [41]:
```
h=9.25
s=regressor.predict([[h]])
print("If a student studies for{} hours per day he/she will score {}% in exam.".format(h,s))
```

```
If a student studies for9.25 hours per day he/she will score [93.69173249]% in exam.
```

Evaluating the model

For simplicity here, we have chosen the mean square error. There are many such metrics.

In [42]:
```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```

In [ ]: