GRIP: The Spark Foundation

Data Science and Bussiness Analytics Intern

Author: Shalaka Patil

Task 1 : Prediction Using Supervised ML

In this task we have to predict the percentage score of a student based on the number of hours studied. The task has two variables where the features is the no. of hours studied and the target values is the percentage score. This can be solved using Simple Linear Regression.

In [1]:
```
# Import Python Library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#Importing data set
data =pd.read_csv("C:/Users/DELL/OneDrive/Desktop/data.txt")
print("Importing Data Successfully")
data.head(10)
```

```
Importing Data Successfully
```

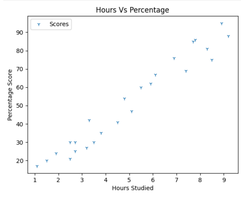
Out[2]:

	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 [3]:
```
# Check wheather Data imported successfully or not
print("For this we print first 10 data of Data.set")
data.head(10)
print("You have correctly imported Data set")
```

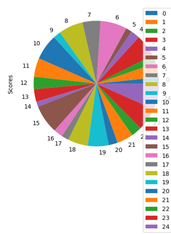
```
For this we print first 10 data of Data.set
You have correctly imported Data set
```

In [4]:
```
# Plot the graph, for detail analysis of data
data.plot(x='Hours',y='Scores',style='1')
plt.title("Hours Vs Percentage")
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



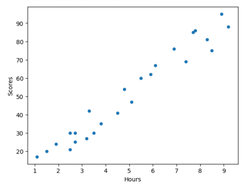
In [5]:
```
data.plot.pie(x='Hours',y='Scores')
```

Out[5]:
```
<AxesSubplot: ylabel='Scores'>
```



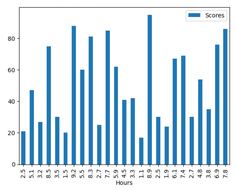
In [6]:
```
data.plot.scatter(x='Hours',y='Scores')
```

Out[6]:
```
<AxesSubplot: xlabel='Hours', ylabel='Scores'>
```



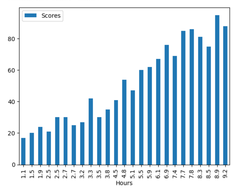
In [7]:
```
data.plot.bar(x='Hours',y='Scores')
```

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



In [8]:
```
data.sort_values(["Hours"],axis=0,ascending=[True],inplace=True)
data.head(10)
data.plot.bar(x='Hours',y='Scores')
```

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



After plotting Different graph,we have observed that as study hours increases,score is also increase.Which is good sign of correct data.In our daily life,we have observed the same phenomenon.

In [9]:
```
# Now we have prepared the data for our model
X=data.iloc[:, :-1].values
y=data.iloc[:, 1].values
#print(X)
```

In [10]:
```
# Now we have divide the data from training and testing the model
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 [11]:
```
# Training the Algorithm
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()


#from sklearn.ensemble import RandomForestRegressor
#regressor = RandomForestRegressor(n_estimators = 1000,random_state= 42)


regressor.fit(X_train, y_train)
print("Training Complete.")
```

```
Training Complete.
```

In [12]:
```
#Now,Our model is ready it's time to test it.
print(X_test)
print("Predection of Score")
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[[2.7]
 [1.9]
 [7.7]
 [6.1]
 [4.5]]
Predection of Score
[28.6177145  20.88803334 76.92822173 61.46885942 46.0094971 ]
```

In [13]:
```
#Now checking the Accuracy of Our Model
df = pd.DataFrame ({'Actual' : y_test, 'predicted' : y_pred})

df
```

Out[13]:

	Actual	predicted
0	30	28.617714
1	24	20.880033
2	85	76.928222
3	67	61.468859
4	41	46.009497

In [14]:
```
#Now It's time to prediction with custom Input
hours = [[9.25]]
pred = regressor.predict(hours)
print(pred)
```

```
[91.90447898]
```

In [15]:
```
#Evaluating the Model
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.621333622532765
```

In [16]:
```
# Training the Algorithm
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()


from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 1000,random_state= 42)


regressor.fit(X_train, y_train)
print("Training Complete.")
```

```
Training Complete.
```

In [17]:
```
#Now,Our model is ready it's time to test it.
print(X_test)
print("Predection of Score")
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[[2.7]
 [1.9]
 [7.7]
 [6.1]
 [4.5]]
Predection of Score
[25.2673  21.74895 81.476   62.308   50.121  ]
```

In [18]:
```
#Now checking the Accuracy of Our Model
df = pd.DataFrame ({'Actual' : y_test, 'predicted' : y_pred})


df
```

Out[18]:

	Actual	predicted
0	30	25.26730
1	24	21.74895
2	85	81.47600
3	67	62.30800
4	41	50.12100

In [19]:
```
#Now It's time to prediction with custom Input
hours = [[9.25]]
pred = regressor.predict(hours)
print(pred)
```

```
[88.332]
```

In [20]:
```
#Evaluating the Model
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.864150000000003
```

Conclusion

1. Linear Regression Mean Absolute Error: 4.621333622532765

2. Random Forest Mean Absolute Error: 4.864150000000003