Prediction using Supervised ML

Prompt

- Predict the percentage of an student based on the no. of study hours.
- This is a simple linear regression task as it involves just 2 variables.
- You can use R, Python, SAS Enterprise Miner or any other tool
- Data can be found at http://bit.ly/w-data
- What will be predicted score if a student studies for 9.25 hrs/ day?
- Sample Solution : https://bit.ly/2HxiGGJ

Social Media

Github - https://github.com/VarunBhattacharya/TheSparkFoundation_Supervised_ML (https://github.com/VarunBhattacharya/TheSparkFoundation_Supervised_ML)
LinkedIn - https://www.linkedin.com/in/varunbhattacharya/ (https://www.linkedin.com/in/varunbhattacharya/)
Instagram - https://www.instagram.com/varunbhattacharya.in/ (https://www.instagram.com/varunbhattacharya.in/)

Import necessary libraries

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

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression

%matplotlib inline
```

Load the Dataset

In [2]:
```
df = pd.read_csv('Dataset.csv')
```

In [3]:
```
df.head()
```

Out[3]:

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

Information about dataset

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

Out[4]:

	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

Training the dataset

In [5]:
```
#Differentiating between given train data and outcome data
X = df['Hours'].values.reshape(-1,1)
y = df['Scores'].values.reshape(-1,1)
```

In [6]:
```
#Split the data into 70% train and 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

Model - Linear Regression

In [7]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

In [8]:
```
y_pred
```

Out[8]:
```
array([[69.11660197],
       [91.47446999],
       [28.2891908 ],
       [26.34502837],
       [88.55822634],
       [59.39578979],
       [28.2891908 ],
       [16.62421618]])
```

Accuracy scores of predicted value

In [9]:
```
print('Mean Squarred Error: ', mean_squared_error(y_test,y_pred))
print('Root Mean Squarred Error: ', np.sqrt(mean_squared_error(y_test,y_pred)))
print('R2 Score: ' , r2_score(y_test,y_pred) * 100)
```

```
Mean Squarred Error:  18.278980896571674
Root Mean Squarred Error:  4.275392484506151
R2 Score:  97.7604434156892
```

Creating a new dataframe with the actual and predicted values

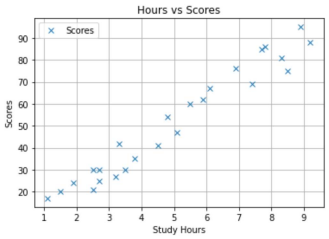
In [10]:
```
df_new = pd.DataFrame({'Actual':y_test.flatten(), 'Predicted':y_pred.flatten()})
df_new.head()
```

Out[10]:

	Actual	Predicted
0	76	69.116602
1	88	91.474470
2	30	28.289191
3	30	26.345028
4	95	88.558226

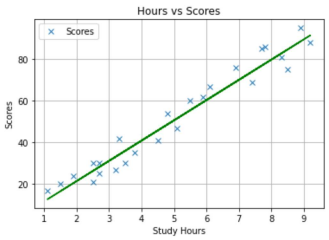
Visualizing the data

In [11]:
```
#Actual Data Points
df.plot(x = 'Hours', y = 'Scores', style = 'x')
plt.title('Hours vs Scores')
plt.xlabel('Study Hours')
plt.ylabel('Scores')
plt.grid()
plt.show()
```



In [12]:
```
#Plotting the linear regression line
line = lr.coef_*X + lr.intercept_

df.plot(x = 'Hours', y = 'Scores', style = 'x')
plt.plot(X, line, color = 'green');
plt.title('Hours vs Scores')
plt.xlabel('Study Hours')
plt.ylabel('Scores')
plt.grid()
plt.show()
```



Predicted score

In [13]:
```
lrCoef = list(lr.coef_)
lrIntr = list(lr.intercept_)
regLine = 'scores = ' + str(lrCoef[0][0]) + ' * hours + ' + str(lrIntr[0])
print(f'The regression line for the above scenario is: {regLine}.')
```

```
The regression line for the above scenario is: scores = 9.72081218274112 * hours + 2.0429979098238107.
```

In [14]:
```
givenHours = 9.25
```

In [15]:
```
predScore = lrCoef[0][0] * givenHours + lrIntr[0]
print(f'The predicted score for study of {givenHours} hours is: {predScore}.')
```

```
The predicted score for study of 9.25 hours is: 91.96051060017918.
```

Done By:
Varun Bhattacharya