Task - 1 : Prediction using Supervised ML

Linear Regression

In this section we will see how the Python and Scikit-Learn Library for Machine Learning can be used to implement Regression functions. Let's start with simple Linear Regression involving Two(2) variables.

Simple Linear Regression

In this Regression Task we are going to predict the Percentage(%) of marks that a Student is expected to score based on the number of studied hours. This is a simple Linear Regression task.

Author: Vedant Amit Pandya

In [1]:
```
#Let's import the libraries we are going to use.
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#Import the data from the CSV File
data = pd.read_csv("Task_1_data.csv")
print("Data Imported Successfully!")
```

```
Data Imported Successfully!
```

In [3]:
```
data.head(10)
```

Out[3]:

	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 [4]:
```
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
```

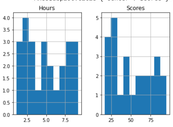
In [5]:
```
data.describe()
```

Out[5]:

	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 [6]:
```
data.hist()
```

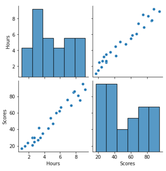
Out[6]:
```
array([[<AxesSubplot:title={'center':'Hours'}>,
        <AxesSubplot:title={'center':'Scores'}>]], dtype=object)
```



The histogram is used to summarize discrete or continuous data. In other words, it provides a visual interpretation.

In [7]:
```
sns.pairplot(data)
```

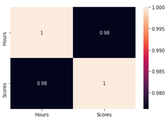
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x2903fc9c2e0>
```



Pairplots are used to plot pairwise relationships in the DataSets.

In [8]:
```
sns.heatmap(data.corr(), annot=True)
```

Out[8]:
```
<AxesSubplot:>
```



Here the HeatMap shows the positive correlation between Hours and Scores.

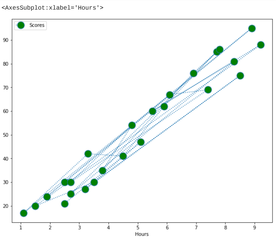
Heatmap is a two-dimensional Graphical Representation of Data where the individual values that are contained in a matrix are represented as colors.

Heatmaps are great for making trends in this kind of data more readily apparent, particularly when the data is ordered and clustered.

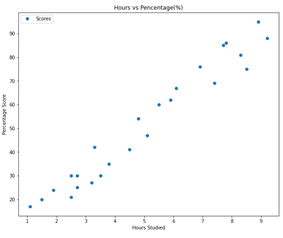
The columns with the correlation 1 are the best correlated and vice-versa.

In [9]:
```
data.plot("Hours", "Scores", linestyle = ":", marker = "o", markersize = "15", markerfacecolor = "green", figsize = (10, 8))
```

Out[9]:
```
<AxesSubplot:xlabel='Hours'>
```



In [10]:
```
data.plot(x = 'Hours', y = 'Scores', style = 'o', figsize = (10, 8))
plt.title('Hours vs Pencentage(%)')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



LaTex: α^2

Preparing Data

In [11]:
```
X = data.iloc[:, :-1].values
Y = data.iloc[:, 1].values
```

In [12]:
```
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)
```

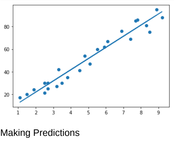
Training Algorithm

In [13]:
```
from sklearn.linear_model import LinearRegression
```

In [14]:
```
regressor = LinearRegression()
regressor.fit(X_train, Y_train)
print("Training Complete!")
```

```
Training Complete!
```

In [15]:
```
line = regressor.coef_*X + regressor.intercept_
plt.scatter(X, Y)
plt.plot(X, line);
plt.show()
```



Making Predictions

In [16]:
```
print(X_test)
Y_pred = regressor.predict(X_test)
```

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

In [17]:
```
df = pd.DataFrame({'Actual': Y_test, 'Predicted': Y_pred})
df
```

Out[17]:

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

In [18]:
```
hours = 9.25
own_pred = regressor.predict([[hours]])
print("Number of Hours = {}".format(hours))
print("Predicted Score = {}".format(own_pred[0]))
```

```
Number of Hours = 9.25
Predicted Score = 93.69173248737538
```

Evaluating the Model

In [19]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, Y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```

Thank You!