**# TASK :PREDICTION USING SUPERVISED ML**

# Linear Regression with Python Scikit Learn

# Importing the libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

# Importing the CSV dataset using pandas

dataset=pd.read\_csv('C:/Users/Rai/Desktop/student\_scores.csv')

dataset.shape

(This means that our dataset has 25 rows and 2 columns)

dataset.head()

(This method retrieves the first 5 records from our dataset)

dataset.describe()

(This is used to see the statistical details of the dataset)

# Plotting of Graph

(To plot our points use the following script)

dataset.plot(x='Hours', y='Scores', style='o')

plt.title('Hours vs Percentage')

plt.xlabel('Hours Studied')

plt.ylabel('Percentage Score')

plt.show()

(From the obtained graph it is seen that there is a positive linear relation between the number of hours studied and percentage of score)

# Preparing the Data

(Now we extract the attributes and labels)

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 1].values

(Now we split this data into training and test sets by using Scikit-Learn's built-in train\_test\_split() method)

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 the Algorithm

(Finally training our algorithm)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

(Retrieving the intercept)

print(regressor.intercept\_)

(Retrieving the slope)

print(regressor.coef\_)

#Making Predictions

y\_pred = regressor.predict(X\_test)

(The y\_pred is a numpy array that contains all the predicted values for the input values in the X\_test series)

(Comparing the actual output values for X\_test with the predicted values)

df = pd.DataFrame({'Actual': y\_test, 'Predicted': y\_pred})

df

# Evaluating the Algorithm

(Finding the values for these metrics using our test data

from sklearn import metrics)

print('Mean Absolute Error:', metrics. mean\_absolute\_error (y\_test, y\_pred))

print('Mean Squared Error:', metrics. mean\_squared\_error (y\_test, y\_pred))

print('Root Mean Squared Error:', np.sqrt (metrics.mean\_squared\_error(y\_test, y\_pred)))

# THANK YOU