**Experiment 11**

**Implementation of learning algorithms for an application**

**Aim:**  To implement Implementation of learning algorithms for an application.

**Problem Description:**

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

**Problem Formulation:**

While training the model we are given :

x: input training data (univariate – one input variable(parameter))

y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.

θ1: intercept

θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

**Source code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn import metrics

%matplotlib inline

from google.colab import files

uploaded = files.upload()

import io

dataset = pd.read\_csv(io.BytesIO(uploaded['student\_scores.csv']))

dataset.shape

dataset.describe()

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

plt.title('Hours vs Percentage')

plt.xlabel('Hours Studied')

plt.ylabel('Percentage Score')

plt.show()

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

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

regressor = LinearRegression()

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

print(regressor.intercept\_)

print(regressor.coef\_)

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

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

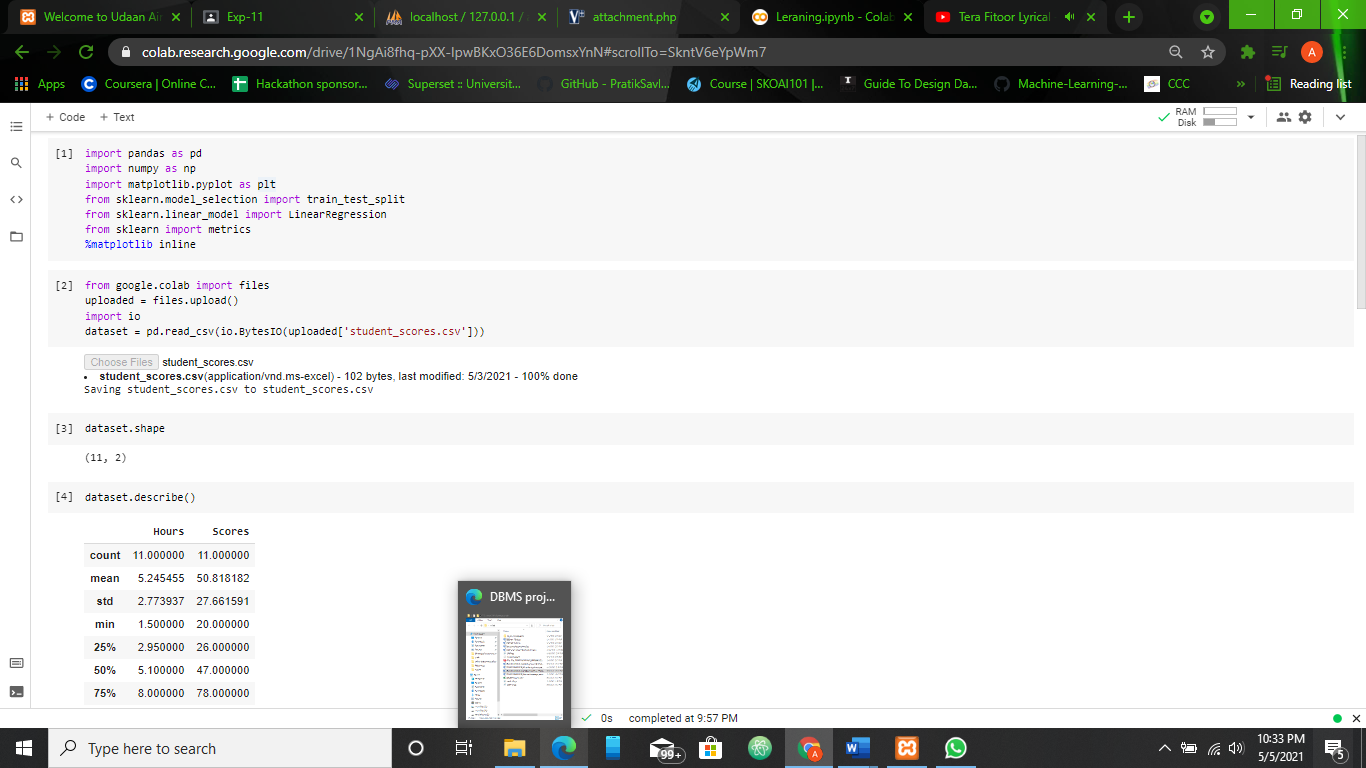
df

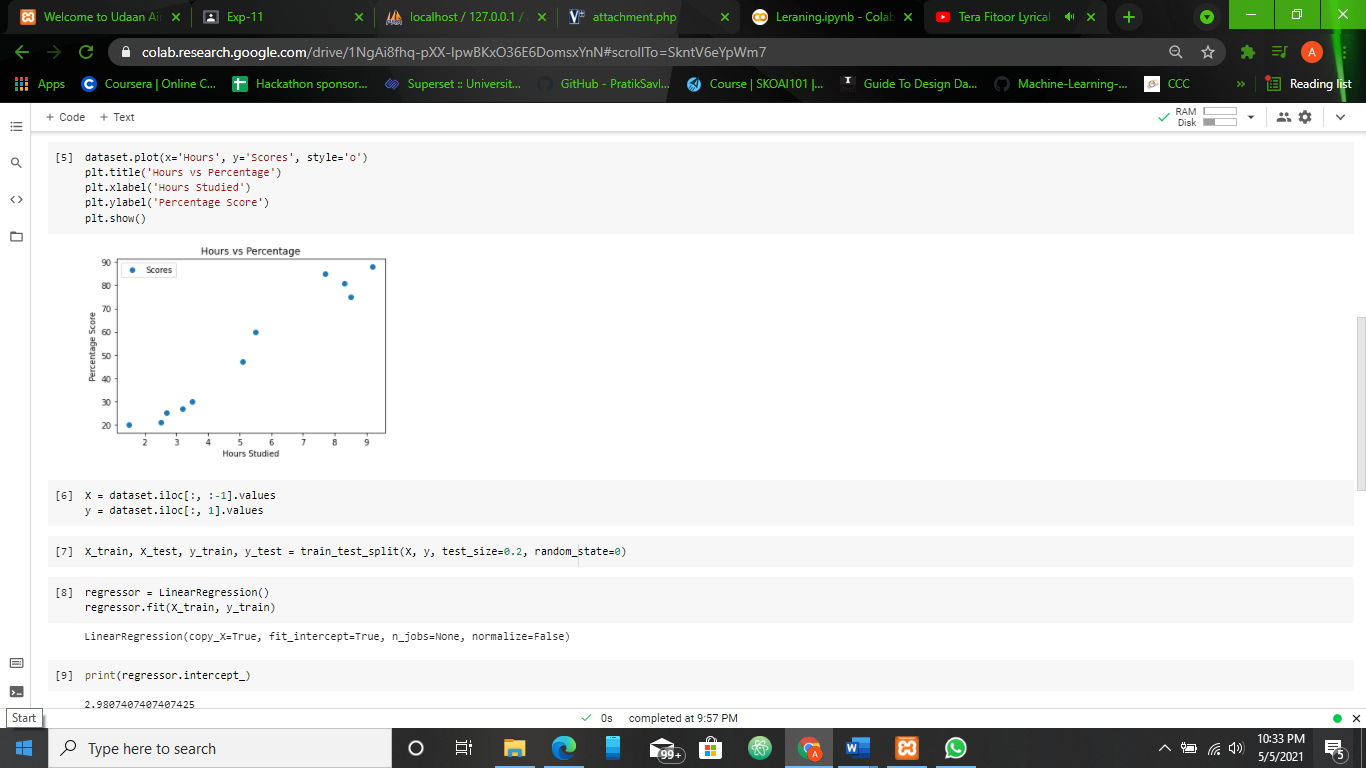
print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

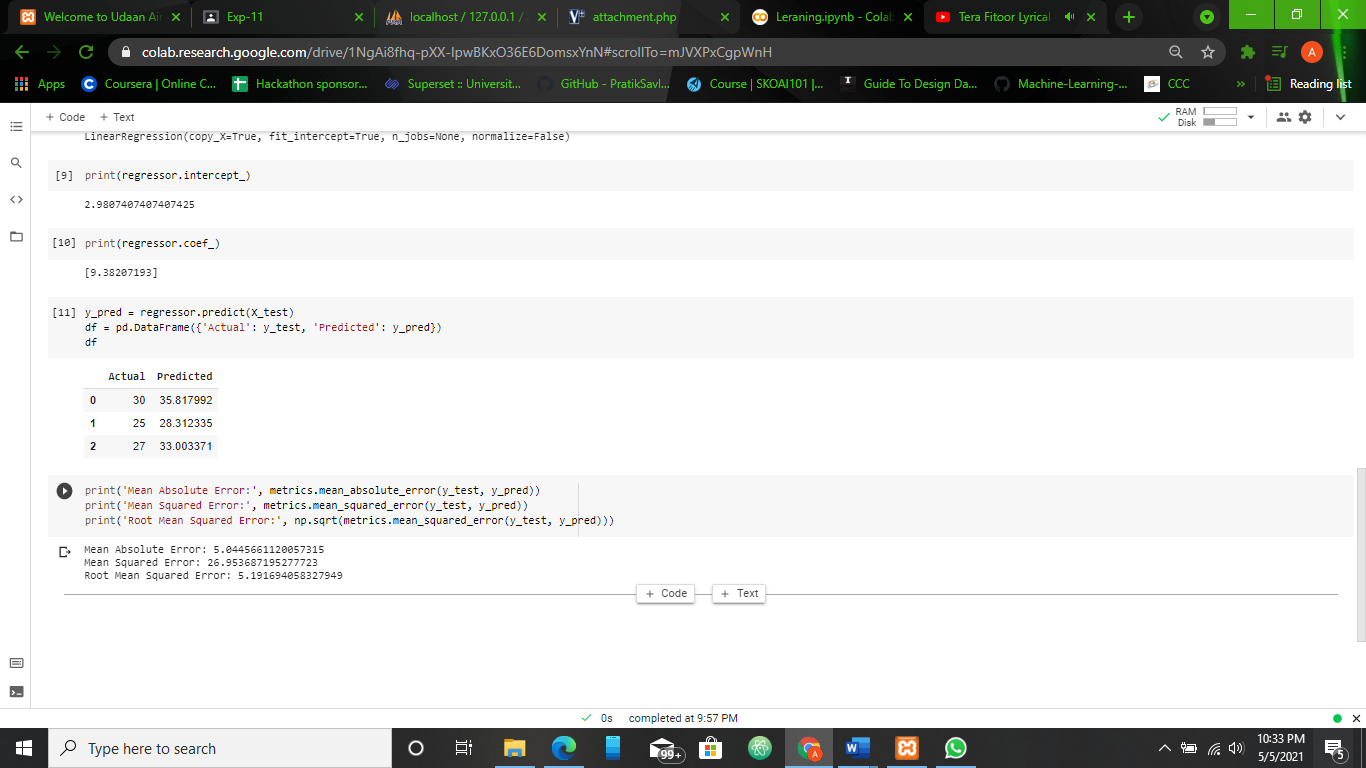
print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

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

**Verification:**







**Result:**

The Implementation of learning algorithms for an application problem., was analyzed and an optimal solution was devised. This solution was the coded & tested against various test cases and documented.