**18CSC305J-Artificial Intelligence**

**LAB 11 – Implementation of learning problems for an algorithm**

**Aim :**

a) Implementation of Linear Regression algorithm to predict students score using the given dataset.

b) Implementation of Support Vector Classification algorithm to classify the cases of breast cancer using the given dataset.

c) Implementation of K-means clustering algorithm to group the customers based on their demographic detail using the given dataset.

**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.
* Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
* K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

**CODE :**

**a.**

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

dataset=pd.read\_csv('student\_scores.csv')

dataset.head()

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)

print('X train shape: ', X\_train.shape)

print('Y train shape: ', y\_train.shape)

print('X test shape: ', X\_test.shape)

print('Y test shape: ', y\_test.shape)

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}) print(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)))

b.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

dataset = pd.read\_csv('diabetes.csv')

print(dataset.head())

dataset = pd.read\_csv('diabetes.csv')

print(dataset.head())

dataset.head()

def diagnosis(x):

if x=='M' :

return 1

if x=='B' :

return 0

dataset['DiabetesPedigreeFunction'] =

dataset['DiabetesPedigreeFunction'].apply(diagnosis)

print(dataset)

svc\_classifier=SVC(kernel='rbf')

svc\_classifier

Y = dataset['DiabetesPedigreeFunction']

X = dataset.drop(columns=['DiabetesPedigreeFunction'])

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=9)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=9)

print('X train shape: ', X\_train.shape)

print('Y train shape: ', Y\_train.shape)

print('X test shape: ', X\_test.shape)

print('Y test shape: ', Y\_test.shape)

svc\_classifier= SVC(kernel='poly')

c.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

%matplotlib inline

data=pd.read\_csv('mall\_customers.csv')

print(data.head())

inVsout=data.iloc[:,[3,4]]

inVsout

plt.scatter(inVsout.iloc[:0],inVsout.iloc[:,1])

kmeans=KMeans(n\_clusters=5)

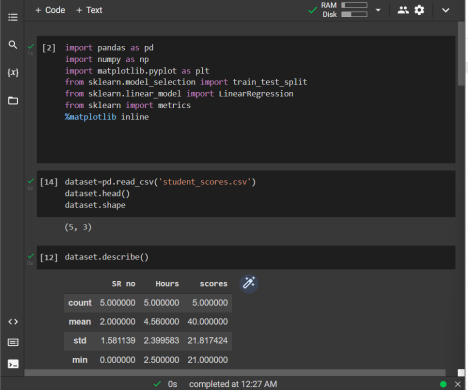
kmeans.fit(inVsout)

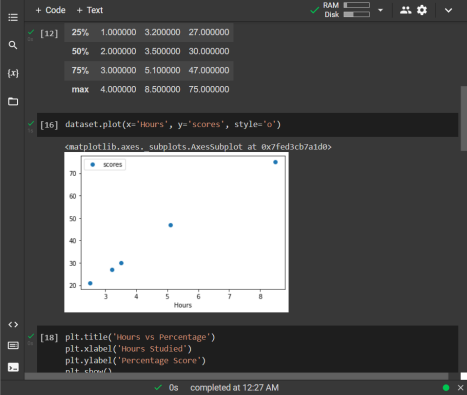
plt.scatter(inVsout.iloc[:0],inVsout.iloc[:,1],c=kmeans.labels\_,cmap='rain bow')

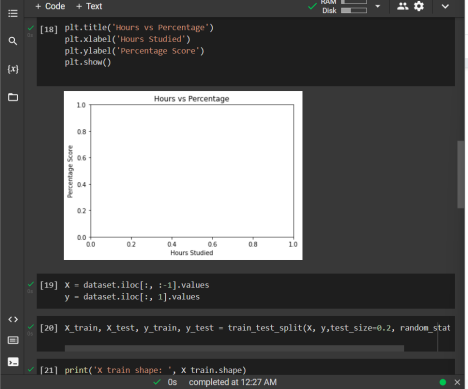
plt.show()

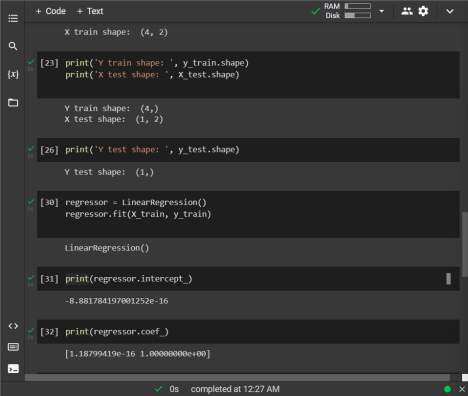
**OUTPUT :**

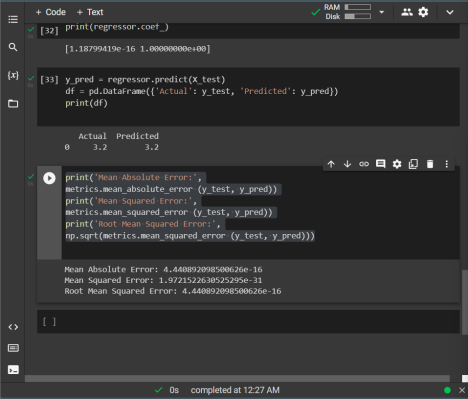
**a.**



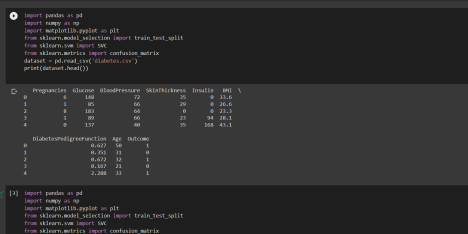


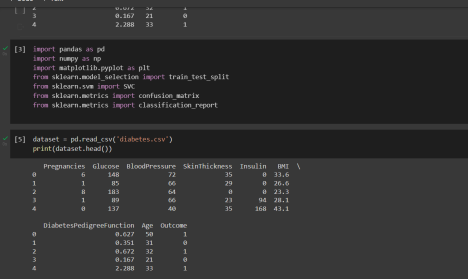


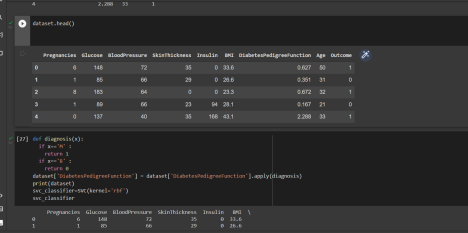


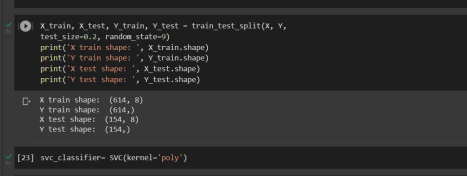
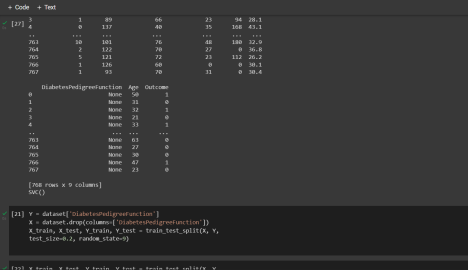


**b.**

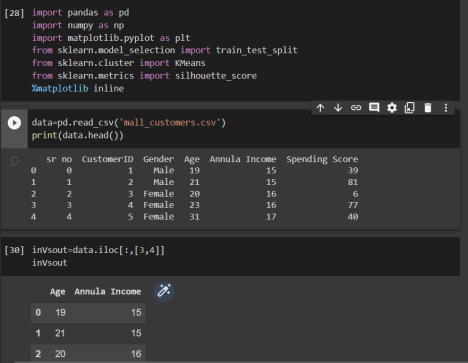


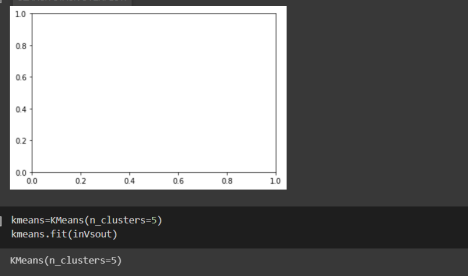


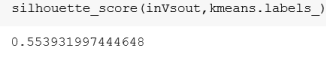




**c.**







**RESULT :**

Hence we have successfully fulfilled implementation of learning algorithms for an example.