**Assignment No. 1**

**Code:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cluster import KMeans

X=[[0.1,0.6],[0.15,0.71],[0.08,0.9],[0.16,0.85],[0.2,0.3],[0.25,0.5],[0.24,0.1],[0.3,0.2]]

centroids=np.array([[0.1,0.6],[0.25,0.5]])

kmeans=KMeans(n\_clusters=2,init=centroids)

kmeans.fit(X)

print("Labels after trainig:",kmeans.labels\_)

#Q1

print("P6 belongs to ",kmeans.labels\_[5],"cluster")

#Q2

print("Population around centroid 2(P6) is ",np.count\_nonzero(kmeans.labels\_==1))

#Q3

print("New values of centroids are:",kmeans.cluster\_centers\_)

from sklearn import datasets

iris=datasets.load\_iris()

iris\_x=iris.data

wcss=list()

for i in range(1,11):

kmeans=KMeans(n\_clusters=i,init='k-means++')

kmeans.fit(iris\_x)

wcss.append(kmeans.inertia\_)

plt.plot(range(1,11),wcss)

plt.xlabel('No. of clusters')

plt.ylabel('WCSS')

plt.show()

kmeans=KMeans(n\_clusters=3,init='k-means++')

kmeans.fit(iris\_x)

y\_means=kmeans.predict(iris\_x)

plt.scatter(iris\_x[y\_means==0,0],iris\_x[y\_means==0,1],c='blue',s=100,label='iris\_sertosa')

plt.scatter(iris\_x[y\_means==1,0],iris\_x[y\_means==1,1],c='yellow',s=100,label='iris\_versicolour')

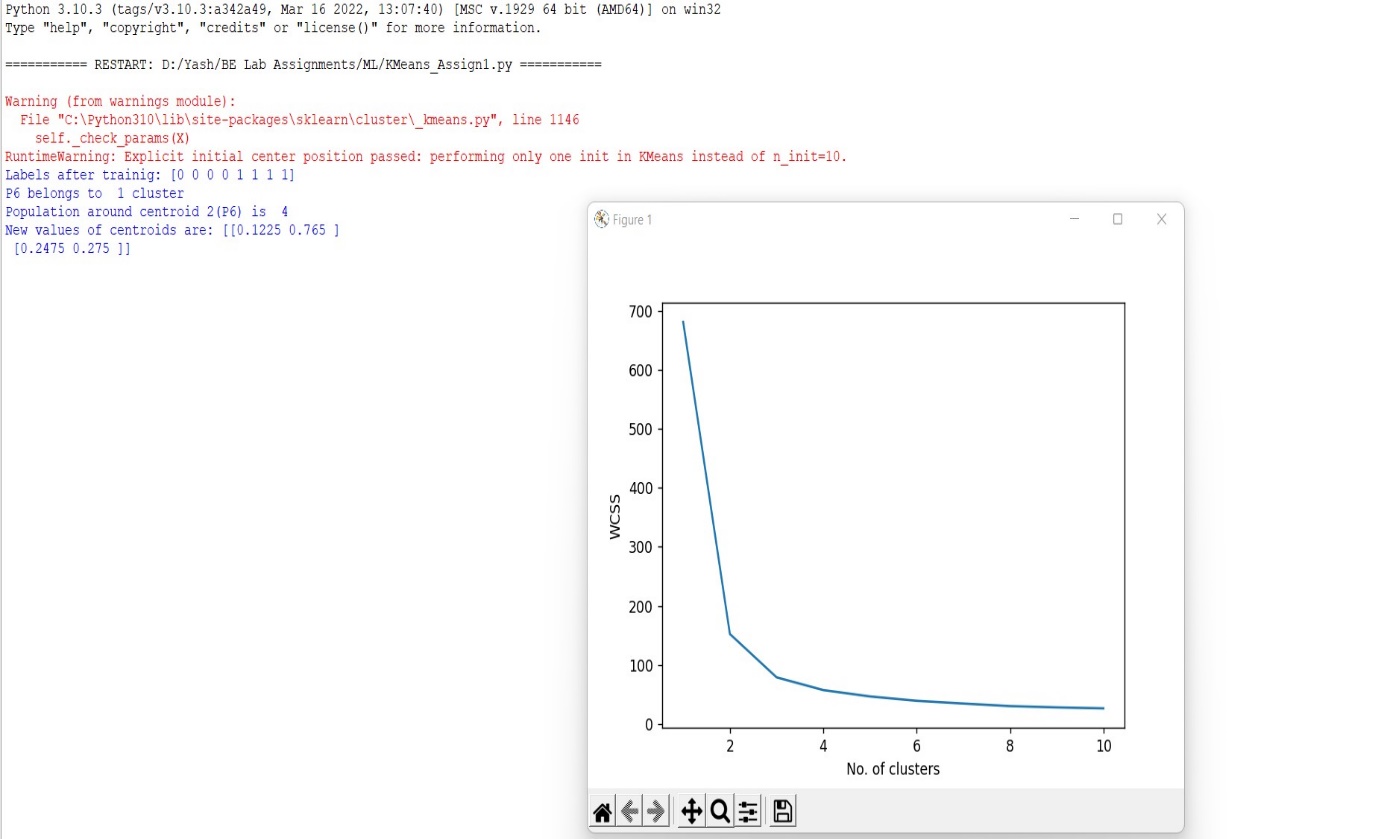
plt.scatter(iris\_x[y\_means==2,0],iris\_x[y\_means==2,1],c='green',s=100,label='iris\_versginica')

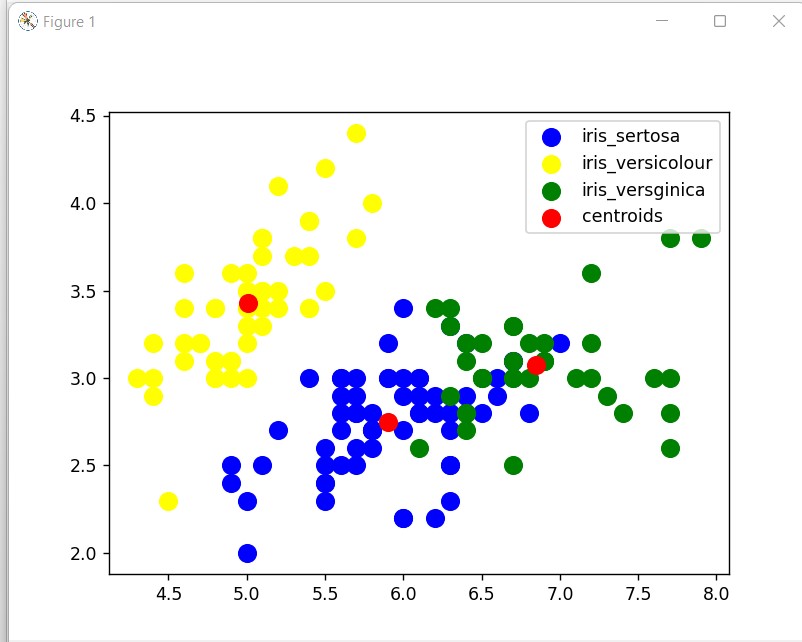
plt.scatter(kmeans.cluster\_centers\_[:,0],kmeans.cluster\_centers\_[:,1],c='red',s=100,label='centroids')

plt.legend()

plt.show()

**Output:-**





**Assignment No. 3**

**Code:-**

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.datasets import load\_diabetes

from sklearn.linear\_model import LinearRegression, Ridge

from sklearn.model\_selection import cross\_val\_score

diabetes = load\_diabetes()

lr = LinearRegression(normalize = True)

lr\_scores = cross\_val\_score(lr, diabetes.data, diabetes.target,cv=10)

print("Scores obtained by Linear Regression: ",lr\_scores)

print("\nMean: ",lr\_scores.mean())

***Output:-***

**Scores obtained by Linear Regression: [0.55614411 0.23056092 0.35357777 0.62190498 0.26587602 0.61819338**

**0.41815916 0.43515232 0.43436983 0.68568514]**

**Mean: 0.4619623619583371**

## Ridge

rg = Ridge(0.005, normalize=True)

rg\_scores = cross\_val\_score(rg, diabetes.data, diabetes.target, cv=10)

print("Scores obtained by Ridge Regression: ",rg\_scores)

print("\nMean: ",rg\_scores.mean())

***Output:-***

**Scores obtained by Ridge Regression: [0.55014745 0.24000807 0.36373382 0.61657429 0.2695777 0.62172635**

**0.42346862 0.42813504 0.43117332 0.68303604]**

**Mean: 0.462758069707298**

## RidgeCV for set of alpha values, alpha\_ to extract alpha value

from sklearn.linear\_model import RidgeCV

rg = RidgeCV(alphas=(1.0, 0.1, 0.01, 0.005, 0.0025, 0.001, 0.00025),normalize=True)

rg.fit(diabetes.data, diabetes.target)

print(rg.alpha\_)

rg\_cv\_scores = cross\_val\_score(rg, diabetes.data, diabetes.target, cv=10)

print("Scores obtained by RidgeCV Regression: ",rg\_cv\_scores)

print("\nMean: ",rg\_scores.mean())

***Output:-***

**Scores obtained by RidgeCV Regression: [0.52803256 0.23657595 0.3565488 0.607513 0.2695777 0.62047382**

**0.42149214 0.43894932 0.43138195 0.6642474 ]**

**Mean: 0.462758069707298**

## Lasso, LassoCV

from sklearn.linear\_model import Lasso, LassoCV

ls = Lasso(alpha=0.005, normalize=True)

ls\_scores = cross\_val\_score(ls, diabetes.data, diabetes.target, cv=10)

print("Scores obtained by Lasso Regression: ",ls\_scores)

print("\nMean: ",ls\_scores.mean())

***Output:-***

**Scores obtained by Lasso Regression: [0.55078146 0.23993097 0.36453647 0.61454396 0.26894036 0.62269952**

**0.42465645 0.42726522 0.43075777 0.68414508]**

**Mean: 0.46282572553668555**

from sklearn.linear\_model import LassoCV

ls\_cv = LassoCV(alphas=(1.0,0.1,0.01,0.005,0.0025,0.001,0.00025),normalize=True)

ls\_cv.fit(diabetes.data, diabetes.target)

ls\_cv.alpha\_

ls\_cv\_scores = cross\_val\_score(ls\_cv, diabetes.data, diabetes.target,cv=10)

print("Scores obtained by LassoCV Regression: ",ls\_cv\_scores)

print("\nMean: ",ls\_cv\_scores.mean())

***Output:-***

**Scores obtained by LassoCV Regression: [0.51796189 0.23824666 0.35415718 0.59752149 0.27503201 0.62269952**

**0.41851368 0.42047653 0.42562829 0.68231416]**

**Mean: 0.45525514142746404**

## ElasticNet, ElasticNetCV

from sklearn.linear\_model import ElasticNet, ElasticNetCV

en = ElasticNet(alpha=0.001,l1\_ratio=0.8, normalize=True)

en\_scores = cross\_val\_score(en, diabetes.data, diabetes.target, cv=10)

print("Scores obtained by ElasticNet Regression: ",en\_scores)

print("\nMean: ",en\_scores.mean())

***Output:-***

**Scores obtained by ElasticNet Regression: [0.53103739 0.24682675 0.38160097 0.60832995 0.2830996 0.62083992**

**0.43113636 0.43484866 0.43055758 0.6676087 ]**

**Mean: 0.46358858847836454**

encv = ElasticNetCV(alphas=(0.1, 0.01,0.005, 0.0025, 0.001),l1\_ratio=(0.1,0.25,0.5,0.75,

0.8), normalize=True)

encv.fit(diabetes.data, diabetes.target)

print(encv.alpha\_)

print(encv.l1\_ratio\_)

encv\_scores = cross\_val\_score(encv, diabetes.data, diabetes.target,cv=10)

print("Scores obtained by ElasticNetCV Regression: ",encv\_scores)

print("\nMean: ",encv\_scores.mean())

***Output:-***

**Scores obtained by ElasticNetCV Regression: [0.52796681 0.24682675 0.38160097 0.60743123 0.28526958 0.61931497**

**0.43113636 0.43907824 0.43137097 0.66429212]**

**Mean: 0.46342880159848276**

## Comparative analysis

#Linear Regression Score

print(lr\_scores.mean())

#Ridge Score

print(rg\_scores.mean())

#Lasso Score

print(ls\_scores.mean())

#ElasticNet Score

print(en\_scores.mean())

***Output:-* 0.4619623619583371**

**0.462758069707298**

**0.46282572553668555**

**0.46358858847836454**

import matplotlib.pyplot as plt

import numpy as np

objects = ('Linear','Ridge','Lasso','ElasticNet')

y\_pos = np.arange(len(objects))

p = [0.4619623619583371,0.4627580697072979,0.4628257255366856,0.46358858847836454]

plt.bar(y\_pos, p, align='center',alpha=0.5, color=['hotpink', 'yellow', 'lime','cyan'])

plt.xticks(y\_pos, objects)

plt.ylabel('Accuracy')

plt.title('Linear Models')

ElasticNetCV(alphas=(0.1, 0.01, 0.005, 0.0025, 0.001),

l1\_ratio=(0.1, 0.25, 0.5, 0.75, 0.8), normalize=True)

plt.show()

***Output:-***

