Assignment No. 1

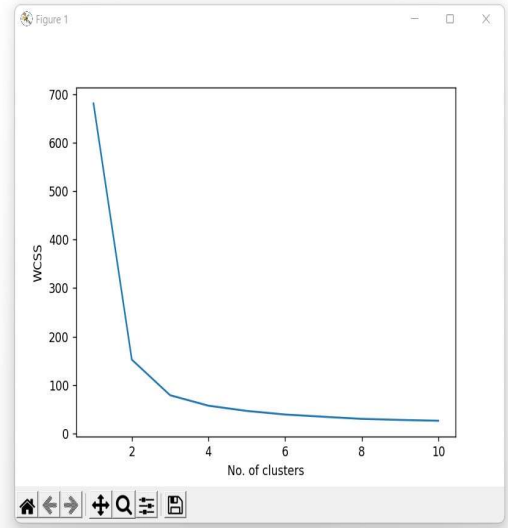
Code:-

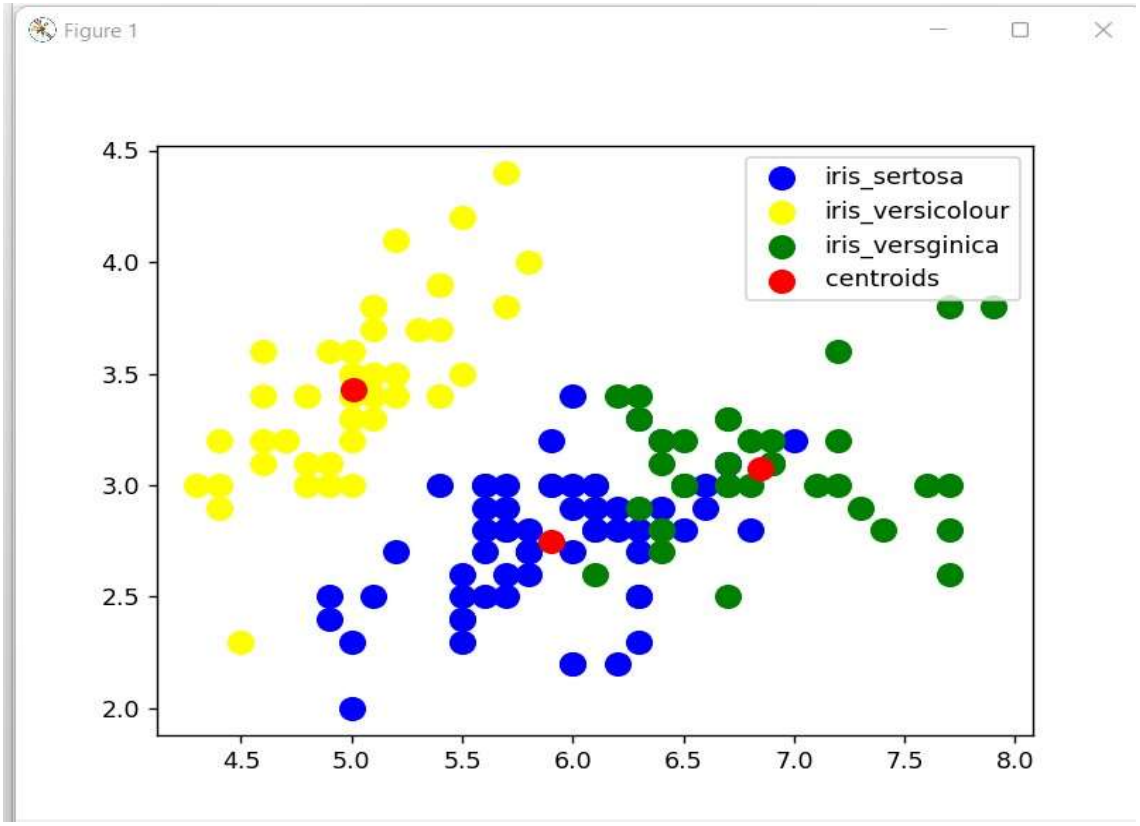
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans

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_versicolo
ur')
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='c
entroids')
plt.legend()
plt.show()
```

Output:-

```
Python 3.10.3 (tags/v3.10.3:a342a49, Mar 16 2022, 13:07:40) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========== RESTART: D:/Yash/BE Lab Assignments/ML/KMeans_Assign1.py ==========

Warning (from warnings module):
  File "C:\Python310\lib\site-packages\sklearn\cluster\_kmeans.py", line 1146
    self._check_params(X)
RuntimeWarning: Explicit initial center position passed: performing only one init in KMeans instead of n_init=10.
Labels after trainig: [0 0 0 0 1 1 1 1]
P6 belongs to  1 cluster
Population around centroid 2(P6) is  4
New values of centroids are: [[0.1225 0.765 ]
 [0.2475 0.275 ]]
```





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:-