Date. 05/1/2021...

Expt. No.....8..... Page No.....20.....

8 Apply EM algorithm to cluster a set of data stored in a csv file use the same dataset for clustering. using k-means algorithm. compare the result of these two algorithm and comment on the quality of clustering. You can add Java/python ML library classes.

```
import matplotlib.pyplot as plot.
from sklearn import datasets.
from sklearn.cluster import kmeans.
import sklearn.metrics as sm
import. pandas as pd.
import numpy as np.
iris = datasets.load_iris()
x = pd.DataFrame(iris.data)
x.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width'].
y = pd.DataFrame(iris.target)
y.columns = ['Target']

model = kmeans(n_clusters = 3)
model.fit(x)
model.labels_
plt.figure(figsize = (14,7))
colormap = np.array(['red', 'lime', 'black'])
plt.subplot(1,2,1).
plt.scatter(x.petal_length, x.petal_width, c=
colormap[y.targets], s=40)
plt.title('Real classification')
```

Teacher's Signature : .....

```
plt.subplot(1,2,2)
plt.scatter(x.petal_length, x.petal_width, c=colormap
    [model.labels_], s=40)
plt.title('k means classification')
plt.figsize = (14,7)
predY = np.choose(model.labels_,[0,1,2].astype.
(np.int64)
print(predY)
plt.subplot(1,2,1).
plt.scatter(x.petal_length,x.petal_width, c=
    Colormap[y.Targets], s=40)
plt.title('Real classification')
plt.subplot(1,2,2)
plt.scatter(x.petal_length,x.petal width, c=
colormap[predY], s=40)
plt.title('k means classification').
print('The accuracy score of k means:', sm.
accuracy_score(y, model.labels_))
print('The confusion matrix of k means:',
sm.confusion_matrix(y.model_labels))
from sklearn import preprocessing
Scaler = preprocessing.StandardScaler()
Scaler.fit(x)

xsa = Scaler.transform(x)
xs = pd.DataFrame(xsa, column = x.columns)
from sklearn.mixture import gaussian
mixture
gmm = GaussianMixture(n_components=3
```

```
gmm.fit(xs)
y_cluster_gmm = gmm.predict(xs)
plt.subplot(2,2,3)
plt.scatter(x.petal_length, x.petal_width, c=color.
    map[y_cluster_gmm], s=40).
plt.title('Gmm classification')
print('the accuracy score of EM:', sm.accuracy_
    score(y, y_cluster_gmm))
print('The confusion matrix of EM:', sm.confu
    sion matrix(y, y_cluster_gmm))
```

Output:

[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 2 2 2 2 0 2 2 2 2
2 2 0 0 2 2 2 2 2 0 2 0 2 0 2 0 2 2 0 0 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 2 0 2
2 0]

The accuracy score of k-means: 624.
The confusion of k-means: [[0 50 0]
[48 0 2]
[14 0 36]]

The accuracy score of EM: 0.96666666667
The confusion matrix: [[50 0 0]
[0 45 5]
[0 0 50]]

Date...18/12/24.....

Expt. No.....9:.....

Page No.....23.....

write a program to implement k-means Nearest neighbour algorithm to classify the Iris data set. print both correct & wrong predictions. Java/ python ML library classes can be used for the problem

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import kneighborsclassifier
from sklearn import datasets
iris = datasets.load_iris()
print("Iris Dataset loaded")
x_train, x_test, y_train, y_test = train_test_split(iris.
    data, iris.target, test_size = 0.1)
print("Dataset is split into training and testing...")
print("size of training data and its label", x_train.
    shape, y_train.shape)
print("size of training data and its label", x_test.
    shape, y_test.shape)
for i in range(len(iris.target_names)):
    print("label", i, "-", str(iris.target_names[i]))
classifier = kneighborsclassifier(n_neighbors=1)
classifier = fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print("Results of classification using knn with
    k=1")
for r in range(0, len(x_test)):
    print("sample:", str(x_test[r]), "Actual label:",
    str(y_test[r]), "predict label:", str(y_pred[r]))
print("classification using le accuracy:", classifier
    .score(x_test, y_test));
```

Teacher's Signature : .....

Date.....

Expt. No.....9..... Page No.....24.....

```
from sklearn.metrics import classification_report
confusion matrix

from sklearn matrices import accuracy_scan.
print ('confusion matrix').
print(confusion_matrix(y_test, y pred))
print ('accuracy matrices').
printf (classification_report (y_test, y_pred))
printf ("correct predictions", accuracy_score
    (y_test, y_pred)
printf ("wrong prediction", 1- accuracy_score.
                (y_test, y_pred)).
```

Teacher's Signature : .....

Date.....

Expt. No.....

Page No.....

output:

IRIS Dataset loaded.
Dataset is split into training & testing...
Size of training data and its label (135,4) (135,)
Size of training data and its label (15,4) (15,)

label 0 - setosa
label 1 - versicolor
label - 2 - virginica

Result of classification using knn with k=1

Sample: [6.5 3.0 5.0, 1.8]. Actual label:2 predicted label:2
Sample: [7.3 2.9 6.3, 1.8] Actual label:2. predicted label:2
Sample [7.1, 3. 5.9, 2.1]. Actual label:2 predicted label:2
Sample [7.7 3. 6.1 2.3] Actual label:2 predicted label:2
Sample: [6.4 3. 5. 1.7] Actual label: 1 predicted label:1
Sample [6.5. 2.6. 4.4, 1.2] Actual label:1 predicted label:1
Sample [5.3 3.7 1.5 0.2] Actual label: 0 predicted label:0
Sample [6.4 3.2 5.3 2.3] Actual label. : 2 predicted label:2
Sample [5.7 3.8 1.7. 0.3] Actual label. : 0 predicted label:0
Sample [4.9 2.5 4.5. 1.7] Actual label : 2 predicted label:2
Sample [5. 2. 3.5 1.7]. Actual label : 1 predicted label:1
Sample [5.2 2.7 3.9 1.4] Actual label : 1 predicted label:1
Sample [5.2 3.4 1.4 0.2] Actual label: 0 predicted label:0
Sample [5.1 2.5 3. 1.1]. Actual label: 1 predicted label:1
Sample [6.3 2.3 4.4 1.3] Actual label 1 predicted label:1

Date.....

Expt. No.....

Page No:.....

classification Accuracy : 0.9333333333333333

Confusion matrix.

[[3 0 0]
 [0 6 0]
 [0 1 5]].

accuracy metrics.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3
1	0.86	1.00	1.92	6
2	1.00	0.83	0.91	6
accuracy			0.93	15
macro avg	0.95	0.94	0.94	15
weighted avg	0.94	0.93	0.93	15

correct prediction. 0.9333333333333333.

wrong prediction 0.06666666666666665.

Date. 05/01/2021

Expt. No. 10

Page No. 25

Implement the non-parametric locally weighted Regression algorithm in order to fit data points. Select the appropriate dataset for your experiment & draw graphs.

```
import matplotlib.pylab as plt
from sklego.linear_model import LowessRegression

n = 100
xs = np.linspace(0, np.pi, n)
ys = 1 + np.sin(xs) + np.cos(xs**2) + np.random.
normal(0, 0.1, n).

mod = LowessRegression(sigma = 0.01, span = 0.5).
      fit(xs.reshape(-1, 1), ys)

xs_new = np.linspace(-1, np.pi+1, n*2)
preds = mod.predict(xs_new.reshape(-1,1))

plt.figure(figsize = (12, 4))
plt.scatter(xs, ys).
plt.plot(xs_new, preds, color = "orange")
```

Teacher's Signature : .....

Expt. No.....

Date.....

Page No.....