**Cairo University** 

**Faculty of engineering**

**Electronics and electrical communication department**

**DSP2 ELC459**

**Assignment 2**

**DSP**

**Submitted to Dr.: Mohsen Rashwan**

**Submitted by**

| **Name** |  | **section Bench No.** |
| --- | --- | --- |
| **Mohamed Maged Khalil** | **3** | **57** |
| **Mohamed Mahmoud Abdelmotaleb** | **3** | **63** |
| **Nour El-din Mohamed Sayed** | **4** | **39** |

1 | Page

Contents

1. DCT Features (180 Dimensions) ...........................................................................................5 1.1. K-Means..............................................................................................................................5 1.1.1. K-Means 1 Cluster .......................................................................................................5 1.1.2. K-Means 4 Clusters......................................................................................................5 1.1.3. K-Means 16 Clusters....................................................................................................5 1.2. GMM...................................................................................................................................6 1.2.1. 1 GMM.........................................................................................................................6 1.2.2. 4 GMM.........................................................................................................................6 1.2.3. 16 GMM.......................................................................................................................6 1.3. SVM ....................................................................................................................................7 1.3.1. SVM Linear Kernel......................................................................................................7 1.3.2. SVM Poly Kernel.........................................................................................................7 1.3.3. SVM RBF Kernel.........................................................................................................7 1.3.4. SVM Sigmoid Kernel...................................................................................................8 2. PCA Features (262 Dimensions)............................................................................................8 2.1. K-Means..............................................................................................................................8 2.1.1. K-Means 1 Cluster .......................................................................................................8 2.1.2. K-Means 4 Clusters......................................................................................................8 2.1.3. K-Means 16 Clusters....................................................................................................9 2.2. GMM...................................................................................................................................9 2.2.1. 1 GMM.........................................................................................................................9 2.2.2. 4 GMM.........................................................................................................................9 2.2.3. 16 GMM.....................................................................................................................10 2.3. SVM ..................................................................................................................................10 2.3.1. SVM Linear Kernel....................................................................................................10 2.3.2. SVM Poly Kernel.......................................................................................................10 2.3.3. SVM RBF Kernel.......................................................................................................11 2.3.4. SVM Sigmoid Kernel.................................................................................................11 3. Extra Trees Features.............................................................................................................12 3.1. K-Means............................................................................................................................12 3.1.1. K-Means 1 Cluster .....................................................................................................12

2 | Page

3.1.2. K-Means 4 Clusters....................................................................................................12 3.1.3. K-Means 16 Clusters..................................................................................................12 3.2. GMM.................................................................................................................................13 3.2.1. 1 GMM.......................................................................................................................13 3.2.2. 4 GMM.......................................................................................................................13 3.2.3. 16 GMM.....................................................................................................................13 3.3. SVM ..................................................................................................................................14 3.3.1. SVM Linear Kernel....................................................................................................14 3.3.2. SVM Poly Kernel.......................................................................................................14 3.3.3. SVM RBF Kernel.......................................................................................................14 3.3.4. SVM Sigmoid Kernel.................................................................................................15 4. Summary: .............................................................................................................................15 5. References............................................................................................................................16 5.1. PCA Features Colab notebook ..........................................................................................17 5.2. DCT Features Colab notebook..........................................................................................17 5.3. Extra Trees Features Colab notebook ...............................................................................17 5.4. PCA Features Code ...........................................................................................................18 5.5. DCT Features Code...........................................................................................................23 5.6. Extra Trees Features Extracted .........................................................................................31

**List of Figures**

Figure 1-1: DCT K-Means 1 Cluster..............................................................................................5 Figure 1-2: DCT K-Means 4 Clusters............................................................................................5 Figure 1-3: DCT K-Means 16 Clusters..........................................................................................5 Figure 1-4: DCT GMM 1 ..............................................................................................................6 Figure 1-5: DCT GMM 4 ..............................................................................................................6 Figure 1-6: DCT GMM 16 ............................................................................................................6 Figure 1-7: DCT SVM Linear Kernel............................................................................................7 Figure 1-8: DCT SVM Poly Kernel...............................................................................................7 Figure 1-9: DCT SVM RBF Kernel...............................................................................................7 Figure 1-10: DCT SVM Sigmoid Kernel.......................................................................................8 Figure 2-1: PCA K-Means 1 Cluster..............................................................................................8 Figure 2-2: PCA K-Means 4 Clusters............................................................................................8 Figure 2-3: PCA K-Means 16 Clusters..........................................................................................9

3 | Page

Figure 2-4: PCA GMM 1 ..............................................................................................................9 Figure 2-5: PCA GMM 4 ..............................................................................................................9 Figure 2-6: PCA GMM 16 ..........................................................................................................10 Figure 2-7: PCA SVM Linear Kernel ..........................................................................................10 Figure 2-8: PCA SVM Poly Kernel .............................................................................................10 Figure 2-9: PCA SVM RBF Kernel.............................................................................................11 Figure 2-10: PCA SVM Sigmoid Kernel.....................................................................................11 Figure 3-1: Extra Trees K-Means 1 Cluster.................................................................................12 Figure 3-2: Extra Trees K-Means 4 Clusters ...............................................................................12 Figure 3-3: Extra Trees K-Means 16 Clusters .............................................................................12 Figure 3-4: Extra Trees GMM 1 .................................................................................................13 Figure 3-5: Extra Trees GMM 4 .................................................................................................13 Figure 3-6: Extra Trees GMM 16 ...............................................................................................13 Figure 3-7: Extra Trees SVM Linear Kernel ...............................................................................14 Figure 3-8: Extra Trees SVM Poly Kernel ..................................................................................14 Figure 3-9: Extra Trees SVM RBF Kernel ..................................................................................14 Figure 3-10: Extra Trees SVM Sigmoid Kernel ..........................................................................15

**List of tables**

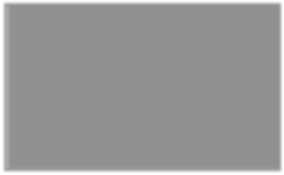
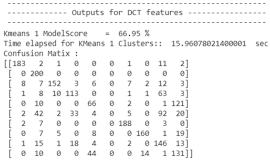
Table 1: Summary of results ........................................................................................................15

4 | Page

**1. DCT Features (180 Dimensions)**

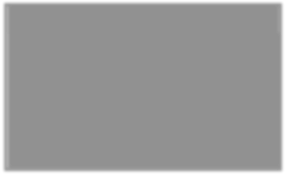
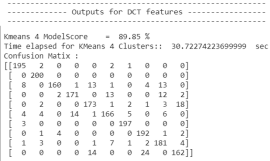
1.1. K-Means

1.1.1. K-Means 1 Cluster



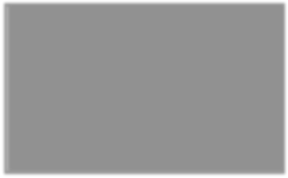
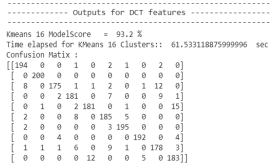
*Figure 1-1: DCT K-Means 1 Cluster*

1.1.2. K-Means 4 Clusters



*Figure 1-2: DCT K-Means 4 Clusters*

1.1.3. K-Means 16 Clusters



*Figure 1-3: DCT K-Means 16 Clusters*

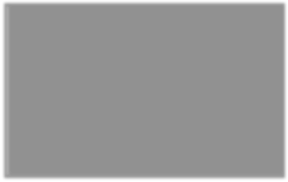
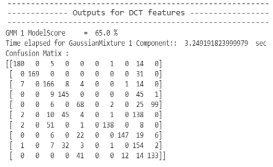
5 | Page

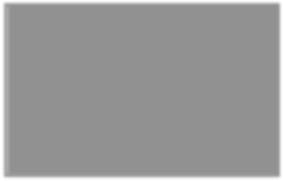
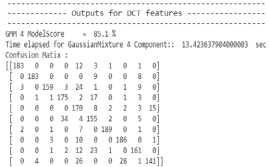
1.2. GMM

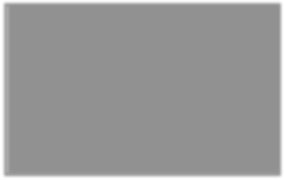
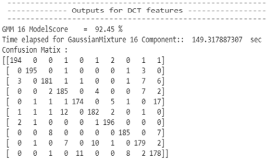
1.2.1. 1 GMM

1.2.2. 4 GMM

1.2.3. 16 GMM

*Figure 1-4: DCT GMM 1*

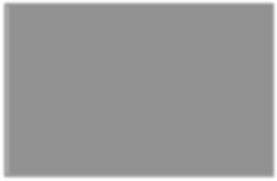
*Figure 1-5: DCT GMM 4*

*Figure 1-6: DCT GMM 16*

6 | Page

1.3. SVM

1.3.1. SVM Linear Kernel



*Figure 1-7: DCT SVM Linear Kernel*

1.3.2. SVM Poly Kernel



*Figure 1-8: DCT SVM Poly Kernel*

1.3.3. SVM RBF Kernel



*Figure 1-9: DCT SVM RBF Kernel*

7 | Page

1.3.4. SVM Sigmoid Kernel



*Figure 1-10: DCT SVM Sigmoid Kernel*

**2. PCA Features (262 Dimensions)**

2.1. K-Means

2.1.1. K-Means 1 Cluster



*Figure 2-1: PCA K-Means 1 Cluster*

2.1.2. K-Means 4 Clusters



*Figure 2-2: PCA K-Means 4 Clusters*

8 | Page

2.1.3. K-Means 16 Clusters



*Figure 2-3: PCA K-Means 16 Clusters*

2.2. GMM

2.2.1. 1 GMM



*Figure 2-4: PCA GMM 1*

2.2.2. 4 GMM



*Figure 2-5: PCA GMM 4*

9 | Page

2.2.3. 16 GMM 2.3. SVM

*Figure 2-6: PCA GMM 16*

2.3.1. SVM Linear Kernel



*Figure 2-7: PCA SVM Linear Kernel*

2.3.2. SVM Poly Kernel



*Figure 2-8: PCA SVM Poly Kernel*

10 | Page

2.3.3. SVM RBF Kernel



*Figure 2-9: PCA SVM RBF Kernel*

2.3.4. SVM Sigmoid Kernel



*Figure 2-10: PCA SVM Sigmoid Kernel*

11 | Page

**3. Extra Trees Features**

3.1. K-Means

3.1.1. K-Means 1 Cluster



*Figure 3-1: Extra Trees K-Means 1 Cluster*

3.1.2. K-Means 4 Clusters



*Figure 3-2: Extra Trees K-Means 4 Clusters*

3.1.3. K-Means 16 Clusters



*Figure 3-3: Extra Trees K-Means 16 Clusters*

12 | Page

3.2. GMM

3.2.1. 1 GMM

3.2.2. 4 GMM

3.2.3. 16 GMM

*Figure 3-4: Extra Trees GMM 1*

*Figure 3-5: Extra Trees GMM 4*

*Figure 3-6: Extra Trees GMM 16*

13 | Page

3.3. SVM

3.3.1. SVM Linear Kernel



*Figure 3-7: Extra Trees SVM Linear Kernel*

3.3.2. SVM Poly Kernel



*Figure 3-8: Extra Trees SVM Poly Kernel*

3.3.3. SVM RBF Kernel



*Figure 3-9: Extra Trees SVM RBF Kernel*

14 | Page

| 3.3.4. SVM Sigmoid Kernel    *Figure 3-10: Extra Trees SVM Sigmoid Kernel*  **4. Summary:** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Features** | | | | | |
| **DCT** | | **PCA** | | **ExtraTree** | |
| **Classifier** | | **Accuracy** | **Processing Time** | **Accuracy** | **Processing Time** | **Accuracy** | **Processing Time** |
| **K-means Clustering** | **1** | 66.95 % | 15.961 sec | 67.55 % | 22.665 sec | 73.0 % | 14.253 sec |
| **4** | 89.85 % | 30.722 sec | 85.9 % | 30.974 sec | 90.5 % | 33.521 sec |
| **16** | 93.2 % | 61.533 sec | 93.89 % | 67.451 sec | 93.89 % | 59.199 sec |
| **GMM** | **1** | 65.0 % | 3.2491 sec | 64.649 % | 7.3275 sec | 73.3 % | 6.2704 sec |
| **4** | 85.1 % | 13.423 sec | 84.65 % | 25.398 sec | 88.75 % | 26.285 sec |
| **16** | 92.45 % | 149.31 sec | 91.95 % | 95.886 sec | 93.6 % | 83.579 sec |
| **SVM** | **Linear** | 93.85 % | 1.9082 sec | 93.85 % | 2.4262 sec | 92.45 % | 2.3834 sec |
| **Poly**  **Kernel** | 96.95 % | 1.7343 sec | 97.5 % | 6.5033 sec | 97.6 % | 2.3462 sec |
| **RBF**  **Kernel** | 97.25 % | 2.3123 sec | 97.65 % | 4.3273 sec | 97.75 % | 3.4911 sec |
| **Sigmoid Kernel** | 86.9 % | 3.7651 sec | 90.9 % | 3.1046 sec | 52.7 % | 5.7334 sec |
| *Table 1: Summary of results*  15 | Page | | | | | | | |

**5. References**

All codes for previous features can be accessed and run directly from google colab. Steps to run the notebook from a link:

1. Open the Colab link i.e.

https://colab.research.google.com/drive/1GxIH6fa2u5ezw8pjsCZcIcQzAIO65xXf#scrollTo=ed qNccj6xeMJ

2. Drag and drop the whole dataset file .zip as in the pictures

16 | Page



After these steps just run the whole cells and the code will work fine. We attached 3 notebooks each one represents the implementation of a feature.

5.1. PCA Features Colab notebook

Link:

https://colab.research.google.com/drive/1GxIH6fa2u5ezw8pjsCZcIcQzAIO65xXf#scrollTo=ed qNccj6xeMJ

5.2. DCT Features Colab notebook

Link:

https://colab.research.google.com/drive/1w\_c0aICJMq0rSem5vjQOIW91cfAl5MoO

5.3. Extra Trees Features Colab notebook

Link:

https://colab.research.google.com/drive/1okX4P1wQjarkOrfLF

ZRpHMfpjJlYDIB?usp=sharing

17 | Page

5.4. PCA Features Code

!pip uninstall scikit-learn -y

!pip install -U scikit-learn

import time

import sys

import pandas as pd

import numpy as np

import os

import tensorflow as tf

import cv2

import sklearn

import matplotlib.image as mpimg

import torch

import scipy

from sklearn import mixture

from sklearn.cluster import KMeans

from tensorflow import keras

from keras.layers.core import Dense

from tensorflow.keras.layers import InputLayer, Flatten

from tensorflow.keras import layers, Input

from tensorflow.keras.models import Sequential, Model

from matplotlib import pyplot as plt

from sklearn.decomposition import PCA

from sklearn import svm

from PIL import Image

from numpy import array

from sklearn.pipeline import make\_pipeline

from sklearn.svm import SVC

from sklearn.datasets import make\_classification

from sklearn.svm import LinearSVC

from sklearn.preprocessing import StandardScaler

from sklearn import metrics

from sklearn.metrics import cluster

from scipy import misc

from scipy.special import comb

from scipy.stats import mode

from sklearn.metrics import accuracy\_score

np.set\_printoptions(threshold=sys.maxsize)

import warnings

warnings.filterwarnings('ignore')

from sklearn.metrics import confusion\_matrix

%matplotlib inline

# Remove this when working on local version, it's just needed when working on colab !unzip '/content/Reduced MNIST Data.zip'

18 | Page

def create\_dataset(img\_folder):

img\_data\_array=[]

class\_name=[]

for dir1 in os.listdir(img\_folder):

for file in os.listdir(os.path.join(img\_folder, dir1)):

image\_path= os.path.join(img\_folder, dir1, file)

image= cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

image=np.array(image)

image = image.astype('double')

image /= 255

img\_data\_array.append(image)

class\_name.append(dir1)

return img\_data\_array, class\_name

def get\_labels\_from\_clustring(true\_labels, predicted\_labels, num\_clusters): labels = np.zeros\_like(predicted\_labels)

for i in range(num\_clusters):

mask = (predicted\_labels == i)

labels[mask] = mode(true\_labels[mask])[0]

return labels

# Training data set part

img\_data, class\_name = create\_dataset (r'/content/Reduced MNIST Data/Reduced Traingin g data')

target\_dict={k: v for v, k in enumerate(np.unique(class\_name))}

target\_val= [target\_dict[class\_name[i]] for i in range(len(class\_name))] X = np.array(img\_data).astype('double')

y = np.array(list(map(int,target\_val)))

# X matrix needs to be reshaped into (Nsamples \* (HxW))

nsamples, nx, ny = X.shape

X\_d2 = X.reshape((nsamples,nx\*ny))

pca = PCA(.95)

pca.fit(X\_d2)

eig\_scores = pca.transform(X\_d2)

# SVM regularization parameter

C = 1.0

19 | Page

# # SVM kernels: Linear, poly, rbf, sigmoid

print("SVM Training Timing details")

t0= time.process\_time ()

clf\_linear = svm.SVC(kernel='linear', C=C).fit(eig\_scores, y) t1\_svm\_linear = time.process\_time () - t0

print("Time elapsed for SVM Linear:: ", t1\_svm\_linear, " sec")

t0= time.process\_time ()

clf\_poly = svm.SVC(kernel='poly', C=C).fit(eig\_scores, y) t1\_svm\_poly = time.process\_time () - t0

print("Time elapsed for SVM Poly:: ", t1\_svm\_poly, " sec")

t0= time.process\_time ()

clf\_rbf = svm.SVC(kernel='rbf', C=C).fit(eig\_scores, y) t1\_svm\_rbf = time.process\_time () - t0

print("Time elapsed for SVM RBF:: ", t1\_svm\_rbf, " sec")

t0= time.process\_time ()

clf\_sigmoid = svm.SVC(kernel='sigmoid', C=C).fit(eig\_scores, y) t1\_svm\_sigmoid = time.process\_time () - t0

print("Time elapsed for SVM Sigmoid:: ", t1\_svm\_sigmoid, " sec")

# # # # # # K Means classifier

print("KMeans Training Timing details")

t0= time.process\_time ()

kmeans16 = KMeans(n\_clusters=160, init='k

means++', max\_iter=300, n\_init=10, random\_state=0).fit(eig\_scores) t1\_km16 = time.process\_time () - t0

print("Time elapsed for KMeans 16 Clusters:: ", t1\_km16, " sec")

t0= time.process\_time ()

kmeans4 = KMeans(n\_clusters=40, init='k

means++', max\_iter=300, n\_init=10, random\_state=0).fit(eig\_scores) t1\_km4 = time.process\_time () - t0

print("Time elapsed for KMeans 4 Clusters:: ", t1\_km4, " sec")

t0= time.process\_time()

kmeans1 = KMeans(n\_clusters=10, init='k

means++', max\_iter=300, n\_init=10, random\_state=0).fit(eig\_scores) t1\_km1 = time.process\_time() - t0

print("Time elapsed for KMeans 1 Clusters:: ", t1\_km1, " sec")

# # # # # #GMM classifier

print("GMM Training Timing details")

t0= time.process\_time ()

20 | Page

GMM\_1 = sklearn.mixture.GaussianMixture(n\_components=10, max\_iter=300, tol=1e 4, random\_state=42, init\_params='kmeans', covariance\_type = 'spherical').fit(eig\_scor es)

t1\_gmm1 = time.process\_time () - t0

print("Time elapsed for GaussianMixture 1 Component:: ", t1\_gmm1, " sec")

t0= time.process\_time ()

GMM\_4 = sklearn.mixture.GaussianMixture(n\_components=40, max\_iter=300, tol=1e 4, random\_state=42, init\_params='kmeans', covariance\_type = 'spherical').fit(eig\_scor es)

t1\_gmm4 = time.process\_time () - t0

print("Time elapsed for GaussianMixture 4 Component:: ", t1\_gmm4, " sec")

t0= time.process\_time ()

GMM\_1\_6 = sklearn.mixture.GaussianMixture(n\_components=160, max\_iter=300, tol=1e 4, random\_state=42, init\_params='kmeans', covariance\_type = 'spherical').fit(eig\_scor es)

t1\_gmm16 = time.process\_time () - t0

print("Time elapsed for GaussianMixture 16 Component:: ", t1\_gmm16, " sec")

## Test dataset part

img\_data\_test, class\_name\_test = create\_dataset(r'/content/Reduced MNIST Data/Reduced Testing data')

target\_dict\_test={k: v for v, k in enumerate(np.unique(class\_name\_test))}

target\_val\_test= [target\_dict\_test[class\_name\_test[i]] for i in range(len(class\_name \_test))]

X\_test = np.array(img\_data\_test)

y = np.array(list(map(int, target\_val\_test)))

# Reshape data to get ready to get the eigen scores

nsamples, nx, ny = X\_test.shape

X\_d2\_test = X\_test.reshape((nsamples,nx\*ny))

eig\_scores = pca.transform(X\_d2\_test)

print("--------------------------------------------------------") print("------------- Outputs for PCA features -----------------") print("--------------------------------------------------------")

print("SVM\_LINEAR ModelScore = ", clf\_linear.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM Linear:: ", t1\_svm\_linear, " sec")

print("Confusion Matix :")

21 | Page

cm = confusion\_matrix(y,clf\_linear.predict(eig\_scores))

print(cm)

print(" ")

# plt.imshow(cm, cmap='binary')

print("SVM\_POLY kernel Score = ", clf\_poly.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM Poly:: ", t1\_svm\_poly, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_poly.predict(eig\_scores))

print(cm)

print(" ")

print("SVM\_RBF kernel Score = ", clf\_rbf.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM RBF:: ", t1\_svm\_rbf, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_poly.predict(eig\_scores))

print(cm)

print(" ")

print("SVM\_SIGMOID kernel Score = ", clf\_sigmoid.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM Sigmoid:: ", t1\_svm\_sigmoid, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_sigmoid.predict(eig\_scores))

print(cm)

print(" ")

predicted\_labels16 = get\_labels\_from\_clustring(y, kmeans16.predict(eig\_scores), 160) predicted\_labels4 = get\_labels\_from\_clustring(y, kmeans4.predict(eig\_scores), 40) predicted\_labels1 = get\_labels\_from\_clustring(y, kmeans1.predict(eig\_scores), 10)

print("Kmeans 1 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels1) \* 100 , "%")

print("Time elapsed for KMeans 1 Clusters:: ", t1\_km1, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels1)

print(cm)

print(" ")

print("Kmeans 4 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels4) \* 100 , "%")

print("Time elapsed for KMeans 4 Clusters:: ", t1\_km4, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels4)

print(cm)

print(" ")

print("Kmeans 16 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels16) \* 10 0, "%")

22 | Page

print("Time elapsed for KMeans 16 Clusters:: ", t1\_km16, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels16)

print(cm)

print(" ")

predicted\_labels16 = get\_labels\_from\_clustring(y, GMM\_1\_6.predict(eig\_scores), 160) predicted\_labels4 = get\_labels\_from\_clustring(y, GMM\_4.predict(eig\_scores), 40) predicted\_labels1 = get\_labels\_from\_clustring(y, GMM\_1.predict(eig\_scores), 10)

print("GMM 1 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels1) \* 100, "%")

print("Time elapsed for GaussianMixture 1 Component:: ", t1\_gmm1, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels1)

print(cm)

print(" ")

print("GMM 4 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels4) \* 100, "%")

print("Time elapsed for GaussianMixture 4 Component:: ", t1\_gmm4, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels4)

print(cm)

print(" ")

print("GMM 16 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels16) \* 100, "%")

print("Time elapsed for GaussianMixture 16 Component:: ", t1\_gmm16, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels16)

print(cm)

print(" ")

5.5. DCT Features Code

!pip uninstall scikit-learn -y

!pip install -U scikit-learn

import sys

import pandas as pd

import numpy as np

import os

import tensorflow as tf

23 | Page

import cv2

import sklearn

import matplotlib.image as mpimg

import torch

import scipy

np.set\_printoptions(threshold=sys.maxsize)

import time

import itertools

from sklearn import mixture

from sklearn.cluster import KMeans

from tensorflow import keras

from keras.layers.core import Dense

from tensorflow.keras.layers import InputLayer, Flatten from tensorflow.keras import layers, Input

from tensorflow.keras.models import Sequential, Model from matplotlib import pyplot as plt

from sklearn.decomposition import PCA

from sklearn import svm

from PIL import Image

from numpy import array

from sklearn.pipeline import make\_pipeline

from sklearn.svm import SVC

from sklearn.datasets import make\_classification from sklearn.svm import LinearSVC

from sklearn.preprocessing import StandardScaler from sklearn import metrics

from sklearn.metrics import cluster

from scipy import misc

from scipy.special import comb

from sklearn.metrics import accuracy\_score

from scipy.stats import mode

from scipy.fftpack import fft, dct

from sklearn.metrics import confusion\_matrix

%matplotlib inline

!unzip '/content/Reduced MNIST Data.zip'

def create\_dataset(img\_folder):

img\_data\_array=[]

class\_name=[]

for dir1 in os.listdir(img\_folder):

for file in os.listdir(os.path.join(img\_folder, dir1)): image\_path= os.path.join(img\_folder, dir1, file)

24 | Page

image= cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE) image=np.array(image)

image = image.astype('double')

image /= 255

img\_data\_array.append(image)

class\_name.append(dir1)

return img\_data\_array, class\_name

def get\_labels\_from\_clustring(true\_labels, predicted\_labels, num\_clusters): labels = np.zeros\_like(predicted\_labels)

for i in range(num\_clusters):

mask = (predicted\_labels == i)

labels[mask] = mode(true\_labels[mask])[0]

return labels

def zigzag(input):

#initializing the variables

#----------------------------------

h = 0

v = 0

vmin = 0

hmin = 0

vmax = input.shape[0]

hmax = input.shape[1]

#print(vmax ,hmax )

i = 0

output = np.zeros(( vmax \* hmax))

#----------------------------------

while ((v < vmax) and (h < hmax)):

if ((h + v) % 2) == 0: # going up

if (v == vmin):

#print(1)

output[i] = input[v, h] # if we got to the first line

if (h == hmax):

v = v + 1

else:

h = h + 1

25 | Page

i = i + 1

elif ((h == hmax -1 ) and (v < vmax)): # if we got to the last column #print(2)

output[i] = input[v, h]

v = v + 1

i = i + 1

elif ((v > vmin) and (h < hmax -1 )): # all other cases #print(3)

output[i] = input[v, h]

v = v - 1

h = h + 1

i = i + 1

else: # going down

if ((v == vmax -1) and (h <= hmax -1)): # if we got to the last line #print(4)

output[i] = input[v, h]

h = h + 1

i = i + 1

elif (h == hmin): # if we got to the first column #print(5)

output[i] = input[v, h]

if (v == vmax -1):

h = h + 1

else:

v = v + 1

i = i + 1

elif ((v < vmax -1) and (h > hmin)): # all other cases #print(6)

output[i] = input[v, h]

v = v + 1

h = h - 1

i = i + 1

if ((v == vmax-1) and (h == hmax-1)): # bottom right element #print(7)

output[i] = input[v, h]

break

26 | Page

#print ('v:',v,', h:',h,', i:',i)

return output

# implement 2D DCT

def dct2(a):

return dct(dct(a.T, norm='ortho').T, norm='ortho')

# Training data set part

img\_data, class\_name = create\_dataset (r'/content/Reduced MNIST Data/Reduced Traingin g data')

target\_dict={k: v for v, k in enumerate(np.unique(class\_name))}

target\_val= [target\_dict[class\_name[i]] for i in range(len(class\_name))] X = np.array(img\_data).astype('double')

y = np.array(list(map(int,target\_val)))

# X matrix needs to be reshaped into (Nsamples \* (HxW))

nsamples, nx, ny = X.shape

X\_d2 = X.reshape((nsamples,nx\*ny))

eig\_scores = np.empty((0,180), int)

for i in range(0, 10000):

dct\_out = dct2(X\_d2[i, :].reshape((1, 784)))

z\_out = np.array(zigzag(dct\_out))

z\_out1 = z\_out[:180].reshape(1, 180)

eig\_scores = np.append(eig\_scores, z\_out1, axis = 0)

# # SVM regularization parameter

C = 1.0

# # SVM kernels: Linear, poly, rbf, sigmoid

print("SVM Training Timing details")

t0= time.process\_time ()

clf\_linear = svm.SVC(kernel='linear', C=C).fit(eig\_scores, y)

t1\_svm\_linear = time.process\_time () - t0

print("Time elapsed for SVM Linear:: ", t1\_svm\_linear, " sec")

t0= time.process\_time ()

clf\_poly = svm.SVC(kernel='poly', C=C).fit(eig\_scores, y)

t1\_svm\_poly = time.process\_time () - t0

print("Time elapsed for SVM Poly:: ", t1\_svm\_poly, " sec")

27 | Page

t0= time.process\_time ()

clf\_rbf = svm.SVC(kernel='rbf', C=C).fit(eig\_scores, y)

t1\_svm\_rbf = time.process\_time () - t0

print("Time elapsed for SVM RBF:: ", t1\_svm\_rbf, " sec")

t0= time.process\_time ()

clf\_sigmoid = svm.SVC(kernel='sigmoid', C=C).fit(eig\_scores, y)

t1\_svm\_sigmoid = time.process\_time () - t0

print("Time elapsed for SVM Sigmoid:: ", t1\_svm\_sigmoid, " sec")

# # # # # # K Means classifier

print("KMeans Training Timing details")

t0= time.process\_time ()

kmeans16 = KMeans(n\_clusters=160, init='k

means++', max\_iter=300, n\_init=10, random\_state=0).fit(eig\_scores) t1\_km16 = time.process\_time () - t0

print("Time elapsed for KMeans 16 Clusters:: ", t1\_km16, " sec")

t0= time.process\_time ()

kmeans4 = KMeans(n\_clusters=40, init='k

means++', max\_iter=300, n\_init=10, random\_state=0).fit(eig\_scores) t1\_km4 = time.process\_time () - t0

print("Time elapsed for KMeans 4 Clusters:: ", t1\_km4, " sec")

t0= time.process\_time()

kmeans1 = KMeans(n\_clusters=10, init='k

means++', max\_iter=300, n\_init=10, random\_state=0).fit(eig\_scores) t1\_km1 = time.process\_time() - t0

print("Time elapsed for KMeans 1 Clusters:: ", t1\_km1, " sec")

# # # # # #GMM classifier

print("GMM Training Timing details")

t0= time.process\_time ()

GMM\_1 = sklearn.mixture.GaussianMixture(n\_components=10, max\_iter=300, tol=1e 4, random\_state=42, init\_params='kmeans', covariance\_type = 'spherical').fit(eig\_scor es)

t1\_gmm1 = time.process\_time () - t0

print("Time elapsed for GaussianMixture 1 Component:: ", t1\_gmm1, " sec")

t0= time.process\_time ()

GMM\_4 = sklearn.mixture.GaussianMixture(n\_components=40, max\_iter=300, tol=1e 4, random\_state=42, init\_params='kmeans', covariance\_type = 'spherical').fit(eig\_scor es)

t1\_gmm4 = time.process\_time () - t0

print("Time elapsed for GaussianMixture 4 Component:: ", t1\_gmm4, " sec") t0= time.process\_time ()

28 | Page

GMM\_1\_6 = sklearn.mixture.GaussianMixture(n\_components=160, max\_iter=300, tol=1e 4, random\_state=42, init\_params='kmeans', covariance\_type = 'spherical').fit(eig\_scor es)

t1\_gmm16 = time.process\_time () - t0

print("Time elapsed for GaussianMixture 16 Component:: ", t1\_gmm16, " sec")

## Test dataset part

img\_data\_test, class\_name\_test = create\_dataset(r'/content/Reduced MNIST Data/Reduced Testing data')

target\_dict\_test={k: v for v, k in enumerate(np.unique(class\_name\_test))}

target\_val\_test= [target\_dict\_test[class\_name\_test[i]] for i in range(len(class\_name \_test))]

X\_test = np.array(img\_data\_test)

y = np.array(list(map(int, target\_val\_test)))

# Reshape data to get ready to get the eigen scores

nsamples, nx, ny = X\_test.shape

X\_d2\_test = X\_test.reshape((nsamples,nx\*ny))

eig\_scores = np.empty((0,180), int)

for i in range(0, 2000):

dct\_out = dct2(X\_d2\_test[i, :].reshape((1, 784)))

z\_out = np.array(zigzag(dct\_out))

z\_out1 = z\_out[:180].reshape(1, 180)

eig\_scores = np.append(eig\_scores, z\_out1, axis = 0)

print("--------------------------------------------------------") print("------------- Outputs for DCT features -----------------") print("--------------------------------------------------------")

print("SVM\_LINEAR ModelScore = ", clf\_linear.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM Linear:: ", t1\_svm\_linear, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_linear.predict(eig\_scores))

print(cm)

print(" ")

# plt.imshow(cm, cmap='binary')

print("SVM\_POLY kernel Score = ", clf\_poly.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM Poly:: ", t1\_svm\_poly, " sec")

29 | Page

print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_poly.predict(eig\_scores))

print(cm)

print(" ")

print("SVM\_RBF kernel Score = ", clf\_rbf.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM RBF:: ", t1\_svm\_rbf, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_poly.predict(eig\_scores))

print(cm)

print(" ")

print("SVM\_SIGMOID kernel Score = ", clf\_sigmoid.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM Sigmoid:: ", t1\_svm\_sigmoid, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_sigmoid.predict(eig\_scores))

print(cm)

print(" ")

predicted\_labels16 = get\_labels\_from\_clustring(y, kmeans16.predict(eig\_scores), 160) predicted\_labels4 = get\_labels\_from\_clustring(y, kmeans4.predict(eig\_scores), 40) predicted\_labels1 = get\_labels\_from\_clustring(y, kmeans1.predict(eig\_scores), 10)

print("Kmeans 1 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels1) \* 100 , "%")

print("Time elapsed for KMeans 1 Clusters:: ", t1\_km1, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels1)

print(cm)

print(" ")

print("Kmeans 4 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels4) \* 100 , "%")

print("Time elapsed for KMeans 4 Clusters:: ", t1\_km4, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels4)

print(cm)

print(" ")

print("Kmeans 16 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels16) \* 10 0, "%")

print("Time elapsed for KMeans 16 Clusters:: ", t1\_km16, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels16)

print(cm)

print(" ")

30 | Page

predicted\_labels16 = get\_labels\_from\_clustring(y, GMM\_1\_6.predict(eig\_scores), 160) predicted\_labels4 = get\_labels\_from\_clustring(y, GMM\_4.predict(eig\_scores), 40) predicted\_labels1 = get\_labels\_from\_clustring(y, GMM\_1.predict(eig\_scores), 10)

print("GMM 1 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels1) \* 100, "%")

print("Time elapsed for GaussianMixture 1 Component:: ", t1\_gmm1, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels1)

print(cm)

print(" ")

print("GMM 4 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels4) \* 100, "%")

print("Time elapsed for GaussianMixture 4 Component:: ", t1\_gmm4, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels4)

print(cm)

print(" ")

print("GMM 16 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels16) \* 100, "%")

print("Time elapsed for GaussianMixture 16 Component:: ", t1\_gmm16, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels16)

print(cm)

print(" ")

5.6. Extra Trees Features Extracted

!pip uninstall scikit-learn -y

!pip install -U scikit-learn

import time

import sys

import pandas as pd

import numpy as np

import os

import tensorflow as tf

import cv2

import sklearn

import matplotlib.image as mpimg

import torch

import scipy

from sklearn import mixture

31 | Page

from sklearn.cluster import KMeans

from tensorflow import keras

from keras.layers.core import Dense

from tensorflow.keras.layers import InputLayer, Flatten from tensorflow.keras import layers, Input

from tensorflow.keras.models import Sequential, Model from matplotlib import pyplot as plt

from sklearn.decomposition import PCA

from sklearn import svm

from PIL import Image

from numpy import array

from sklearn.pipeline import make\_pipeline

from sklearn.svm import SVC

from sklearn.datasets import make\_classification from sklearn.svm import LinearSVC

from sklearn.preprocessing import StandardScaler from sklearn import metrics

from sklearn.metrics import cluster

from scipy import misc

from scipy.special import comb

from scipy.stats import mode

from sklearn.metrics import accuracy\_score

np.set\_printoptions(threshold=sys.maxsize)

import warnings

warnings.filterwarnings('ignore')

from sklearn.feature\_selection import SelectPercentile from sklearn.feature\_selection import chi2 , f\_classif from sklearn.metrics import confusion\_matrix

%matplotlib inline

!unzip '/content/Reduced MNIST Data.zip'

def create\_dataset(img\_folder):

img\_data\_array=[]

class\_name=[]

for dir1 in os.listdir(img\_folder):

for file in os.listdir(os.path.join(img\_folder, dir1)): image\_path= os.path.join(img\_folder, dir1, file) image= cv2.imread(image\_path, -1)

image=np.array(image)

image = image.astype('double')

image /= 255

32 | Page

img\_data\_array.append(image)

class\_name.append(dir1)

return img\_data\_array, class\_name

def get\_labels\_from\_clustring(true\_labels, predicted\_labels, num\_clusters): labels = np.zeros\_like(predicted\_labels)

for i in range(num\_clusters):

mask = (predicted\_labels == i)

labels[mask] = mode(true\_labels[mask])[0]

return labels

# Training data set part

from sklearn.manifold import TSNE

from sklearn.feature\_selection import SelectKBest

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.feature\_selection import SelectFromModel

img\_data, class\_name = create\_dataset (r'/content/Reduced MNIST Data/Reduced Traingin g data')

target\_dict={k: v for v, k in enumerate(np.unique(class\_name))}

target\_val= [target\_dict[class\_name[i]] for i in range(len(class\_name))] X = np.array(img\_data).astype('double')

y = np.array(list(map(int,target\_val)))

# X matrix needs to be reshaped into (Nsamples \* (HxW))

nsamples, nx, ny = X.shape

X\_d2 = X.reshape((nsamples,nx\*ny))

clf = ExtraTreesClassifier(n\_estimators=50)

clf = clf.fit(X\_d2, y)

clf.feature\_importances\_

model = SelectFromModel(clf, prefit=True)

X = model.transform(X\_d2)

eig\_scores = X

# SVM regularization parameter

C = 1.0

33 | Page

# # SVM kernels: Linear, poly, rbf, sigmoid

print("SVM Training Timing details")

t0= time.process\_time ()

clf\_linear = svm.SVC(kernel='linear', C=C).fit(eig\_scores, y) t1\_svm\_linear = time.process\_time () - t0

print("Time elapsed for SVM Linear:: ", t1\_svm\_linear, " sec")

t0= time.process\_time ()

clf\_poly = svm.SVC(kernel='poly', C=C).fit(eig\_scores, y) t1\_svm\_poly = time.process\_time () - t0

print("Time elapsed for SVM Poly:: ", t1\_svm\_poly, " sec")

t0= time.process\_time ()

clf\_rbf = svm.SVC(kernel='rbf', C=C).fit(eig\_scores, y) t1\_svm\_rbf = time.process\_time () - t0

print("Time elapsed for SVM RBF:: ", t1\_svm\_rbf, " sec")

t0= time.process\_time ()

clf\_sigmoid = svm.SVC(kernel='sigmoid', C=C).fit(eig\_scores, y) t1\_svm\_sigmoid = time.process\_time () - t0

print("Time elapsed for SVM Sigmoid:: ", t1\_svm\_sigmoid, " sec")

# # # # # # K Means classifier

print("KMeans Training Timing details")

t0= time.process\_time ()

kmeans16 = KMeans(n\_clusters=160, init='k

means++', max\_iter=300, n\_init=10, random\_state=0).fit(eig\_scores) t1\_km16 = time.process\_time () - t0

print("Time elapsed for KMeans 16 Clusters:: ", t1\_km16, " sec")

t0= time.process\_time ()

kmeans4 = KMeans(n\_clusters=40, init='k

means++', max\_iter=300, n\_init=10, random\_state=0).fit(eig\_scores) t1\_km4 = time.process\_time () - t0

print("Time elapsed for KMeans 4 Clusters:: ", t1\_km4, " sec")

t0= time.process\_time()

kmeans1 = KMeans(n\_clusters=10, init='k

means++', max\_iter=300, n\_init=10, random\_state=0).fit(eig\_scores) t1\_km1 = time.process\_time() - t0

print("Time elapsed for KMeans 1 Clusters:: ", t1\_km1, " sec")

# # # # # #GMM classifier

print("GMM Training Timing details")

t0= time.process\_time ()

34 | Page

GMM\_1 = sklearn.mixture.GaussianMixture(n\_components=10, max\_iter=300, tol=1e 4, random\_state=42, init\_params='kmeans', covariance\_type = 'spherical').fit(eig\_scor es)

t1\_gmm1 = time.process\_time () - t0

print("Time elapsed for GaussianMixture 1 Component:: ", t1\_gmm1, " sec")

t0= time.process\_time ()

GMM\_4 = sklearn.mixture.GaussianMixture(n\_components=40, max\_iter=300, tol=1e 4, random\_state=42, init\_params='kmeans', covariance\_type = 'spherical').fit(eig\_scor es)

t1\_gmm4 = time.process\_time () - t0

print("Time elapsed for GaussianMixture 4 Component:: ", t1\_gmm4, " sec")

t0= time.process\_time ()

GMM\_1\_6 = sklearn.mixture.GaussianMixture(n\_components=160, max\_iter=300, tol=1e 4, random\_state=42, init\_params='kmeans', covariance\_type = 'spherical').fit(eig\_scor es)

t1\_gmm16 = time.process\_time () - t0

print("Time elapsed for GaussianMixture 16 Component:: ", t1\_gmm16, " sec")

## Test dataset part

img\_data\_test, class\_name\_test = create\_dataset(r'/content/Reduced MNIST Data/Reduced Testing data')

target\_dict\_test={k: v for v, k in enumerate(np.unique(class\_name\_test))}

target\_val\_test= [target\_dict\_test[class\_name\_test[i]] for i in range(len(class\_name \_test))]

X\_test = np.array(img\_data\_test)

y = np.array(list(map(int, target\_val\_test)))

# Reshape data to get ready to get the eigen scores

nsamples, nx, ny = X\_test.shape

X\_d2\_test = X\_test.reshape((nsamples,nx\*ny))

X = model.transform(X\_d2\_test)

eig\_scores=X

print("--------------------------------------------------------") print("---------- Outputs for ExtraTree features --------------") print("--------------------------------------------------------")

35 | Page

print("SVM\_LINEAR ModelScore = ", clf\_linear.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM Linear:: ", t1\_svm\_linear, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_linear.predict(eig\_scores))

print(cm)

print(" ")

# plt.imshow(cm, cmap='binary')

print("SVM\_POLY kernel Score = ", clf\_poly.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM Poly:: ", t1\_svm\_poly, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_poly.predict(eig\_scores))

print(cm)

print(" ")

print("SVM\_RBF kernel Score = ", clf\_rbf.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM RBF:: ", t1\_svm\_rbf, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_poly.predict(eig\_scores))

print(cm)

print(" ")

print("SVM\_SIGMOID kernel Score = ", clf\_sigmoid.score(eig\_scores, y) \* 100, "%") print("Time elapsed for SVM Sigmoid:: ", t1\_svm\_sigmoid, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,clf\_sigmoid.predict(eig\_scores))

print(cm)

print(" ")

predicted\_labels16 = get\_labels\_from\_clustring(y, kmeans16.predict(eig\_scores), 160) predicted\_labels4 = get\_labels\_from\_clustring(y, kmeans4.predict(eig\_scores), 40) predicted\_labels1 = get\_labels\_from\_clustring(y, kmeans1.predict(eig\_scores), 10)

print("Kmeans 1 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels1) \* 100 , "%")

print("Time elapsed for KMeans 1 Clusters:: ", t1\_km1, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels1)

print(cm)

print(" ")

print("Kmeans 4 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels4) \* 100 , "%")

print("Time elapsed for KMeans 4 Clusters:: ", t1\_km4, " sec")

print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels4)

print(cm)

36 | Page

print(" ")

print("Kmeans 16 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels16) \* 10 0, "%")

print("Time elapsed for KMeans 16 Clusters:: ", t1\_km16, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels16)

print(cm)

print(" ")

predicted\_labels16 = get\_labels\_from\_clustring(y, GMM\_1\_6.predict(eig\_scores), 160) predicted\_labels4 = get\_labels\_from\_clustring(y, GMM\_4.predict(eig\_scores), 40) predicted\_labels1 = get\_labels\_from\_clustring(y, GMM\_1.predict(eig\_scores), 10)

print("GMM 1 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels1) \* 100, "%")

print("Time elapsed for GaussianMixture 1 Component:: ", t1\_gmm1, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels1)

print(cm)

print(" ")

print("GMM 4 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels4) \* 100, "%")

print("Time elapsed for GaussianMixture 4 Component:: ", t1\_gmm4, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels4)

print(cm)

print(" ")

print("GMM 16 ModelScore = ", metrics.accuracy\_score(y, predicted\_labels16) \* 100, "%")

print("Time elapsed for GaussianMixture 16 Component:: ", t1\_gmm16, " sec") print("Confusion Matix :")

cm = confusion\_matrix(y,predicted\_labels16)

print(cm)

print(" ")

37 | Page