



# 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

## 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

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

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
-----------------------------------	----

## 1. DCT Features (180 Dimensions)

### 1.1. K-Means

#### 1.1.1. K-Means 1 Cluster

```
----- Outputs for DCT features -----  
-----  
Kmeans 1 ModelScore = 66.95 %  
Time elapsed for KMeans 1 Clusters:: 15.96078021400001 sec  
Confusion Matix :  
[[183 2 1 0 0 0 1 0 11 2]  
 [ 0 200 0 0 0 0 0 0 0 0]  
 [ 8 7 152 3 6 0 7 2 12 3]  
 [ 1 8 10 113 0 0 1 1 63 3]  
 [ 0 10 0 0 66 0 2 0 1 121]  
 [ 2 42 2 33 4 0 5 0 92 20]  
 [ 2 7 0 0 0 0 188 0 3 0]  
 [ 0 7 5 0 8 0 0 160 1 19]  
 [ 1 15 1 18 4 0 2 0 146 13]  
 [ 0 10 0 0 44 0 0 14 1 131]]
```

Figure 1-1: DCT K-Means 1 Cluster

#### 1.1.2. K-Means 4 Clusters

```
----- Outputs for DCT features -----  
-----  
Kmeans 4 ModelScore = 89.85 %  
Time elapsed for KMeans 4 Clusters:: 30.72274223699999 sec  
Confusion Matix :  
[[195 2 0 0 0 2 1 0 0 0]  
 [ 0 200 0 0 0 0 0 0 0 0]  
 [ 8 0 160 1 13 1 0 4 13 0]  
 [ 0 0 2 171 0 13 0 0 12 2]  
 [ 0 2 0 0 173 1 2 1 3 18]  
 [ 4 4 0 14 1 166 5 0 6 0]  
 [ 3 0 0 0 0 0 197 0 0 0]  
 [ 0 1 4 0 0 0 0 192 1 2]  
 [ 1 3 0 0 1 7 1 2 181 4]  
 [ 0 0 0 0 14 0 0 24 0 162]]
```

Figure 1-2: DCT K-Means 4 Clusters

#### 1.1.3. K-Means 16 Clusters

```
----- Outputs for DCT features -----  
-----  
Kmeans 16 ModelScore = 93.2 %  
Time elapsed for KMeans 16 Clusters:: 61.533118875999996 sec  
Confusion Matix :  
[[194 0 0 1 0 2 1 0 2 0]  
 [ 0 200 0 0 0 0 0 0 0 0]  
 [ 8 0 175 1 1 2 0 1 12 0]  
 [ 0 0 2 181 0 7 0 0 9 1]  
 [ 0 1 0 2 181 0 1 0 0 15]  
 [ 2 0 0 8 0 185 5 0 0 0]  
 [ 2 0 0 0 0 3 195 0 0 0]  
 [ 0 0 4 0 0 0 0 192 0 4]  
 [ 1 1 1 6 0 9 1 0 178 3]  
 [ 0 0 0 0 12 0 0 5 0 183]]
```

Figure 1-3: DCT K-Means 16 Clusters

## 1.2. GMM

### 1.2.1. 1 GMM

```
----- Outputs for DCT features -----  
-----  
GMM 1 ModelScore      = 65.0 %  
Time elapsed for GaussianMixture 1 Component:: 3.249191823999979 sec  
Confusion Matix :  
[[180  0  5  0  0  0  1  0 14  0]  
 [  0 169  0  0  0  0  0  0 31  0]  
 [  7  0 166  8  4  0  0  1 14  0]  
 [  0  0  9 145  0  0  0  0 45  1]  
 [  0  0  6  0 68  0  2  0 25 99]  
 [  2  0 10 45  4  0  1  0 138  0]  
 [  2  0 51  0  1  0 138  0  8  0]  
 [  0  0  6  0 22  0  0 147 19  6]  
 [  1  0  7 32  3  0  1  0 154  2]  
 [  0  0  0  0 41  0  0  12 14 133]]
```

Figure 1-4: DCT GMM 1

### 1.2.2. 4 GMM

```
----- Outputs for DCT features -----  
-----  
GMM 4 ModelScore      = 85.1 %  
Time elapsed for GaussianMixture 4 Component:: 13.423637904000003 sec  
Confusion Matix :  
[[183  0  0  0 12  3  1  0  1  0]  
 [  0 183  0  0  0  9  0  0  8  0]  
 [  3  0 159  3 24  1  0  1  9  0]  
 [  0  1  1 175  2 17  0  1  3  0]  
 [  0  0  0  0 170  8  2  2  3 15]  
 [  0  0  0 34  4 155  2  0  5  0]  
 [  2  0  1  0  7  0 189  0  1  0]  
 [  0  0  3  0 10  0  0 186  0  1]  
 [  0  0  1  2 12 23  1  0 161  0]  
 [  0  4  0  0 26  0  0 28  1 141]]
```

Figure 1-5: DCT GMM 4

### 1.2.3. 16 GMM

```
----- Outputs for DCT features -----  
-----  
GMM 16 ModelScore     = 92.45 %  
Time elapsed for GaussianMixture 16 Component:: 149.317887307 sec  
Confusion Matix :  
[[194  0  0  1  0  1  2  0  1  1]  
 [  0 195  0  1  0  0  0  1  3  0]  
 [  3  0 181  1  1  0  0  1  7  6]  
 [  0  0  2 185  0  4  0  0  7  2]  
 [  0  1  1  1 174  0  5  1  0 17]  
 [  1  1  1 12  0 182  2  0  1  0]  
 [  2  1  0  0  0  1 196  0  0  0]  
 [  0  0  8  0  0  0  0 185  0  7]  
 [  0  1  0  7  0 10  1  0 179  2]  
 [  0  0  1  0 11  0  0  8  2 178]]
```

Figure 1-6: DCT GMM 16

## 1.3. SVM

### 1.3.1. SVM Linear Kernel

```
----- Outputs for DCT features -----
SVM_LINEAR ModelScore = 93.85 %
Time elapsed for SVM Linear:: 1.9082992849999982 sec
Confusion Matix :
[[195  0  1  1  0  1  2  0  0  0]
 [  0 199  0  0  0  0  0  0  1  0]
 [  4  1 177  2  3  3  2  2  6  0]
 [  1  1  6 181  1  3  0  2  3  2]
 [  2  0  0  1 188  0  2  0  1  6]
 [  5  1  1  3  6 180  2  0  2  0]
 [  1  0  2  1  1  1 194  0  0  0]
 [  0  0  5  1  0  0  0 192  0  2]
 [  2  2  0  0  2  3  3  0 188  0]
 [  1  0  0  0 11  0  0  5  0 183]]
```

Figure 1-7: DCT SVM Linear Kernel

### 1.3.2. SVM Poly Kernel

```
----- Outputs for DCT features -----
SVM_POLY kernel Score = 96.95 %
Time elapsed for SVM Poly:: 1.7343109749999996 sec
Confusion Matix :
[[198  0  1  0  0  0  0  0  1  0]
 [  0 199  0  0  0  0  0  0  1  0]
 [  3  0 186  2  2  0  0  2  5  0]
 [  0  1  1 193  0  2  0  0  0  3]
 [  2  0  0  0 191  0  2  0  0  5]
 [  1  0  0  1  1 191  4  0  2  0]
 [  2  0  0  0  0  0 198  0  0  0]
 [  0  0  4  0  0  0  0 196  0  0]
 [  0  0  0  0  3  1  1  0 195  0]
 [  0  0  0  0  6  0  0  2  0 192]]
```

Figure 1-8: DCT SVM Poly Kernel

### 1.3.3. SVM RBF Kernel

```
----- Outputs for DCT features -----
SVM_RBF kernel Score = 97.25 %
Time elapsed for SVM RBF:: 2.3123254089999996 sec
Confusion Matix :
[[198  0  1  0  0  0  0  0  1  0]
 [  0 199  0  0  0  0  0  0  1  0]
 [  3  0 186  2  2  0  0  2  5  0]
 [  0  1  1 193  0  2  0  0  0  3]
 [  2  0  0  0 191  0  2  0  0  5]
 [  1  0  0  1  1 191  4  0  2  0]
 [  2  0  0  0  0  0 198  0  0  0]
 [  0  0  4  0  0  0  0 196  0  0]
 [  0  0  0  0  3  1  1  0 195  0]
 [  0  0  0  0  6  0  0  2  0 192]]
```

Figure 1-9: DCT SVM RBF Kernel

### 1.3.4. SVM Sigmoid Kernel

```
----- Outputs for DCT features -----
SVM_SIGMOID kernel Score = 86.9 %
Time elapsed for SVM Sigmoid:: 3.765142583999996 sec
Confusion Matix :
[[183  0  1  1  2 13  0  0  0  0]
 [  0 195  0  1  0  0  0  0  4  0]
 [  8  0 157  4  7  1  8  2 13  0]
 [  2  0  6 166  0 10  0  0 14  2]
 [  1  0  1  0 187  0  2  0  1  8]
 [  8  8  3 10 12 148  3  0  8  0]
 [  4  0  8  1  0  1 186  0  0  0]
 [  0  0  6  0  1  0  0 188  0  5]
 [  2  3  2  5  1 30  5  0 150  2]
 [  0  0  0  0  7  1  0 13  1 178]]
```

Figure 1-10: DCT SVM Sigmoid Kernel

## 2. PCA Features (262 Dimensions)

### 2.1. K-Means

#### 2.1.1. K-Means 1 Cluster

```
----- Outputs for PCA features -----
Kmeans 1 ModelScore = 67.55 %
Time elapsed for KMeans 1 Clusters:: 22.665666325999993 sec
Confusion Matix :
[[183  1  0  0  0  0  1  0 14  1]
 [  0 200  0  0  0  0  0  0  0  0]
 [  7  3 156  3  8  0  5  2 15  1]
 [  0  6 10 116  0  0  1  1 64  2]
 [  0  5  0  0 66  0  3  0  4 122]
 [  1 32  1 34  6  0  4  0 101 21]
 [  4  6  1  0  0  0 185  0  4  0]
 [  0  6  6  0 14  0  0 159  1 14]
 [  2  9  1 20  4  0  2  0 149 13]
 [  0  2  0  0 48  0  0 12  1 137]]
```

Figure 2-1: PCA K-Means 1 Cluster

#### 2.1.2. K-Means 4 Clusters

```
----- Outputs for PCA features -----
Kmeans 4 ModelScore = 85.9 %
Time elapsed for KMeans 4 Clusters:: 30.974526501 sec
Confusion Matix :
[[194  1  1  0  0  2  2  0  0  0]
 [  0 198  0  0  0  0  1  0  0  1]
 [  6  0 180  2  0  1  0  2  9  0]
 [  0  1  2 172  0 14  2  0  8  1]
 [  0  0  0  0 165  2  4  8  1 20]
 [  3  1  0 25  0 159  8  0  3  1]
 [  2  0  0  0  0  0 198  0  0  0]
 [  0  1  6  0  1  0  0 189  0  3]
 [  3  1  1 10  3 10 11  3 157  1]
 [  0  0  0  0 49  1  0 44  0 106]]
```

Figure 2-2: PCA K-Means 4 Clusters



### 2.1.3. K-Means 16 Clusters

```
----- Outputs for PCA features -----
Kmeans 16 ModelScore = 93.89999999999999 %
Time elapsed for KMeans 16 Clusters:: 67.450978574 sec
Confusion Matix :
[[195 0 0 0 0 2 2 0 1 0]
 [ 0 199 0 0 0 0 0 0 1 0]
 [ 6 0 176 1 2 2 0 1 12 0]
 [ 0 0 3 181 0 9 0 0 6 1]
 [ 0 0 0 0 182 0 1 2 1 14]
 [ 2 0 0 6 0 187 3 0 2 0]
 [ 1 0 0 0 0 0 0 199 0 0]
 [ 0 0 7 0 0 0 0 0 191 0]
 [ 0 0 0 3 0 9 1 1 185 1]
 [ 0 0 0 0 5 0 0 11 1 183]]
```

Figure 2-3: PCA K-Means 16 Clusters

## 2.2. GMM

### 2.2.1. 1 GMM

```
----- Outputs for PCA features -----
GMM 1 ModelScore = 64.64999999999999 %
Time elapsed for GaussianMixture 1 Component:: 7.327552824999998 sec
Confusion Matix :
[[182 7 0 0 2 0 1 0 8 0]
 [ 0 192 0 0 0 0 0 0 3 5]
 [ 7 4 155 4 14 0 0 1 15 0]
 [ 0 2 13 112 0 0 0 0 71 2]
 [ 0 58 0 0 84 0 2 0 1 55]
 [ 1 75 5 35 4 0 2 0 73 5]
 [ 4 4 3 0 23 0 161 0 5 0]
 [ 0 29 4 0 15 0 0 145 1 6]
 [ 3 34 1 20 5 0 1 0 132 4]
 [ 0 35 0 0 26 0 0 8 1 130]]
```

Figure 2-4: PCA GMM 1

### 2.2.2. 4 GMM

```
----- Outputs for PCA features -----
GMM 4 ModelScore = 84.65 %
Time elapsed for GaussianMixture 4 Component:: 25.398074971 sec
Confusion Matix :
[[193 0 0 0 0 6 1 0 0 0]
 [ 0 180 0 0 0 19 0 0 1 0]
 [ 3 0 176 5 3 3 0 1 9 0]
 [ 0 8 2 178 0 5 0 0 5 2]
 [ 0 1 2 0 149 16 2 3 0 27]
 [ 1 0 0 35 0 158 2 0 4 0]
 [ 3 0 0 0 0 1 196 0 0 0]
 [ 0 0 7 0 5 0 0 175 1 12]
 [ 0 4 1 8 2 18 1 0 165 1]
 [ 0 0 0 0 47 1 0 28 1 123]]
```

Figure 2-5: PCA GMM 4

### 2.2.3. 16 GMM

```
----- Outputs for PCA features -----
GMM 16 ModelScore = 91.95 %
Time elapsed for GaussianMixture 16 Component:: 95.88681789900002 sec
Confusion Matix :
[[195  0  0  1  0  1  2  0  1  0]
 [  0 197  0  0  0  0  1  1  1  0]
 [  4  0 178  3  1  1  0  1 12  0]
 [  0  2  2 182  0  7  0  0  6  1]
 [  0  0  1  2 176  0  0  1  2 18]
 [  1  0  0 10  0 182  5  0  2  0]
 [  3  0  0  0  0  2 195  0  0  0]
 [  0  0  4  0  3  0  0 181  4  8]
 [  1  1  0  6  1  9  1  0 181  0]
 [  0  0  0  0 14  1  0  5  8 172]]
```

Figure 2-6: PCA GMM 16

## 2.3. SVM

### 2.3.1. SVM Linear Kernel

```
----- Outputs for PCA features -----
SVM_LINEAR ModelScore = 93.85 %
Time elapsed for SVM Linear:: 2.4262926910000004 sec
Confusion Matix :
[[198  0  0  1  0  1  0  0  0  0]
 [  0 198  0  0  0  0  0  0  2  0]
 [  4  0 182  3  1  0  2  2  6  0]
 [  1  0  3 181  0  6  0  0  5  4]
 [  2  0  2  0 190  0  0  2  0  4]
 [  2  3  1  2  7 177  2  0  6  0]
 [  3  0  1  0  0  3 193  0  0  0]
 [  0  0  4  0  2  0  0 188  0  6]
 [  1  1  0  3  1  4  4  0 185  1]
 [  0  0  0  0 10  0  0  4  1 185]]
```

Figure 2-7: PCA SVM Linear Kernel

### 2.3.2. SVM Poly Kernel

```
----- Outputs for PCA features -----
SVM_POLY kernel Score = 97.5 %
Time elapsed for SVM Poly:: 6.5033481590000015 sec
Confusion Matix :
[[196  0  0  0  0  1  1  0  2  0]
 [  0 199  0  0  0  0  0  0  1  0]
 [  2  0 188  0  0  1  0  2  7  0]
 [  0  0  0 195  0  1  0  0  3  1]
 [  0  0  0  0 193  0  0  0  1  6]
 [  1  0  0  1  1 192  3  0  2  0]
 [  1  0  0  0  0  1 198  0  0  0]
 [  0  0  4  0  0  0  0 196  0  0]
 [  0  0  0  0  2  0  1  0 197  0]
 [  0  0  0  0  4  0  0  0  0 196]]
```

Figure 2-8: PCA SVM Poly Kernel

### 2.3.3. SVM RBF Kernel

```
----- Outputs for PCA features -----  
-----  
SVM_RBF kernel Score = 97.65 %  
Time elapsed for SVM RBF:: 4.327395884999998 sec  
Confusion Matix :  
[[196  0  0  0  0  1  1  0  2  0]  
[  0 199  0  0  0  0  0  0  1  0]  
[  2  0 188  0  0  1  0  2  7  0]  
[  0  0  0 195  0  1  0  0  3  1]  
[  0  0  0  0 193  0  0  0  1  6]  
[  1  0  0  1  1 192  3  0  2  0]  
[  1  0  0  0  0  1 198  0  0  0]  
[  0  0  4  0  0  0  0 196  0  0]  
[  0  0  0  0  2  0  1  0 197  0]  
[  0  0  0  0  4  0  0  0  0 196]]
```

Figure 2-9: PCA SVM RBF Kernel

### 2.3.4. SVM Sigmoid Kernel

```
----- Outputs for PCA features -----  
-----  
SVM_SIGMOID kernel Score = 90.9 %  
Time elapsed for SVM Sigmoid:: 3.1046003639999995 sec  
Confusion Matix :  
[[190  0  2  0  2  4  0  1  1  0]  
[  0 198  0  0  0  0  0  0  2  0]  
[  9  1 161  3  5  1  6  2 12  0]  
[  1  2  4 179  0  4  0  0  7  3]  
[  0  0  1  0 191  0  1  0  1  6]  
[  8  4  1  6  9 165  4  0  3  0]  
[  4  0  7  1  0  3 183  0  2  0]  
[  0  0  6  0  2  0  0 188  0  4]  
[  4  2  1  0  1  4  4  0 184  0]  
[  0  0  0  0  8  1  0 10  2 179]]
```

Figure 2-10: PCA SVM Sigmoid Kernel

### 3. Extra Trees Features

#### 3.1. K-Means

##### 3.1.1. K-Means 1 Cluster

```
----- Outputs for ExtraTree features -----
Kmeans 1 ModelScore = 73.0 %
Time elapsed for KMeans 1 Clusters:: 14.253578875000002 sec
Confusion Matix :
[[170 1 0 0 0 29 0 0 0 0]
 [ 0 200 0 0 0 0 0 0 0 0]
 [ 8 4 146 0 6 12 2 2 20 0]
 [ 0 3 7 148 0 8 0 1 33 0]
 [ 0 3 0 0 162 33 2 0 0 0]
 [ 2 6 2 44 8 130 1 4 3 0]
 [ 2 1 1 0 1 30 165 0 0 0]
 [ 0 3 3 0 3 4 0 181 6 0]
 [ 1 6 1 6 6 18 4 0 158 0]
 [ 0 0 0 0 137 12 0 41 10 0]]
```

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

##### 3.1.2. K-Means 4 Clusters

```
----- Outputs for ExtraTree features -----
Kmeans 4 ModelScore = 90.5 %
Time elapsed for KMeans 4 Clusters:: 33.520312679 sec
Confusion Matix :
[[191 1 2 0 0 4 2 0 0 0]
 [ 0 198 0 0 0 0 0 0 2 0]
 [ 7 0 168 1 0 2 1 3 13 5]
 [ 0 0 1 167 0 17 0 0 12 3]
 [ 0 0 4 0 165 5 2 3 4 17]
 [ 2 1 1 4 0 181 7 0 4 0]
 [ 1 0 0 0 0 0 199 0 0 0]
 [ 0 0 4 0 0 0 0 0 194 0]
 [ 0 0 0 0 0 5 3 2 183 7]
 [ 0 0 13 0 8 1 0 14 0 164]]
```

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

##### 3.1.3. K-Means 16 Clusters

```
----- Outputs for ExtraTree features -----
Kmeans 16 ModelScore = 93.89999999999999 %
Time elapsed for KMeans 16 Clusters:: 59.199444457 sec
Confusion Matix :
[[196 1 0 0 0 1 1 0 1 0]
 [ 0 197 0 0 0 0 0 0 3 0]
 [ 7 0 176 0 3 0 0 4 8 2]
 [ 0 0 3 185 0 4 0 1 4 3]
 [ 2 0 0 0 181 0 1 0 2 14]
 [ 4 1 1 7 0 183 2 0 2 0]
 [ 1 0 0 0 0 1 198 0 0 0]
 [ 0 0 6 0 0 0 0 191 1 2]
 [ 0 0 1 0 2 3 2 1 189 2]
 [ 0 0 0 0 11 0 0 5 2 182]]
```

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

## 3.2. GMM

### 3.2.1. 1 GMM

```
----- Outputs for ExtraTree features -----
-----
GMM 1 ModelScore      = 73.3 %
Time elapsed for GaussianMixture 1 Component:: 6.27047513399998 sec
Confusion Matix :
[[180  0  0  1  1 17  0  0  1  0]
 [  0 133  0  0  0 62  0  0  0  5]
 [  7  0 152  1  5 12  2  1 17  3]
 [  0  1  9 149  0  8  0  0 29  4]
 [  0  0  0  0 143 32  2  0  0 23]
 [  3  1  2 53  4 114  1  0 12 10]
 [  2  0  2  1 11 21 163  0  0  0]
 [  0  0  3  0  4  9  0 168  9  7]
 [  1  0  1  7  4 25  2  0 151  9]
 [  0  0  0  0 60 13  0  6  8 113]]
```

Figure 3-4: Extra Trees GMM 1

### 3.2.2. 4 GMM

```
----- Outputs for ExtraTree features -----
-----
GMM 4 ModelScore      = 88.75 %
Time elapsed for GaussianMixture 4 Component:: 26.285010975000006 sec
Confusion Matix :
[[186  0  6  0  0  7  1  0  0  0]
 [  0 193  0  0  0  6  0  0  1  0]
 [  4  4 171  2  2  3  0  1 13  0]
 [  0  3  1 173  0 11  0  0 11  1]
 [  0  1  7  0 163 14  2  1  2 10]
 [  2  0  2  2  0 186  3  0  5  0]
 [  1  0  2  0  0  2 194  0  1  0]
 [  1  3 10  0  1  2  0 181  0  2]
 [  0  1  4  0  0  9  2  0 182  2]
 [  0  0 15  0 25  2  0 12  0 146]]
```

Figure 3-5: Extra Trees GMM 4

### 3.2.3. 16 GMM

```
----- Outputs for ExtraTree features -----
-----
GMM 16 ModelScore     = 93.60000000000001 %
Time elapsed for GaussianMixture 16 Component:: 83.57996379300002 sec
Confusion Matix :
[[196  0  0  0  0  1  1  0  2  0]
 [  0 197  0  0  0  0  0  0  3  0]
 [  5  0 169  1  2  1  0  6  9  7]
 [  0  0  2 184  0  6  0  1  7  0]
 [  3  0  0  0 178  1  0  0  5 13]
 [  3  0  1  4  0 186  2  0  3  1]
 [  1  0  0  0  0  1 198  0  0  0]
 [  0  0  2  0  0  0  0 192  2  4]
 [  0  0  1  0  0  6  0  0 192  1]
 [  0  0  0  0 12  0  0  5  3 180]]
```

Figure 3-6: Extra Trees GMM 16

### 3.3. SVM

#### 3.3.1. SVM Linear Kernel

```
----- Outputs for ExtraTree features -----  
-----  
SVM_LINEAR ModelScore = 92.45 %  
Time elapsed for SVM Linear:: 2.3834880390000004 sec  
Confusion Matix :  
[[198  0  0  0  0  1  0  0  1  0]  
[  0 198  0  0  0  0  0  0  2  0]  
[  7  1 176  2  2  0  2  2  8  0]  
[  2  0  0 185  1  2  0  2  4  4]  
[  2  0  1  0 187  1  1  1  1  6]  
[  5  1  2  8  5 166  6  0  6  1]  
[  2  0  3  0  1  5 187  0  2  0]  
[  0  0  6  0  3  0  0 187  0  4]  
[  0  2  0  5  2  3  4  0 183  1]  
[  0  0  0  2 10  0  0  5  1 182]]
```

Figure 3-7: Extra Trees SVM Linear Kernel

#### 3.3.2. SVM Poly Kernel

```
----- Outputs for ExtraTree features -----  
-----  
SVM_POLY kernel Score = 97.6 %  
Time elapsed for SVM Poly:: 2.346201763 sec  
Confusion Matix :  
[[199  0  0  0  0  0  0  0  1  0]  
[  0 199  0  0  0  0  0  0  1  0]  
[  3  0 192  0  0  0  0  2  3  0]  
[  0  0  0 196  0  1  0  1  1  1]  
[  1  0  2  0 191  0  0  0  0  6]  
[  1  0  3  1  0 193  2  0  0  0]  
[  1  0  0  0  0  0 198  0  1  0]  
[  0  0  4  0  0  0  0 196  0  0]  
[  0  0  0  1  0  1  3  0 195  0]  
[  0  0  0  0  4  0  0  2  1 193]]
```

Figure 3-8: Extra Trees SVM Poly Kernel

#### 3.3.3. SVM RBF Kernel

```
----- Outputs for ExtraTree features -----  
-----  
SVM_RBF kernel Score = 97.75 %  
Time elapsed for SVM RBF:: 3.4911927570000003 sec  
Confusion Matix :  
[[199  0  0  0  0  0  0  0  1  0]  
[  0 199  0  0  0  0  0  0  1  0]  
[  3  0 192  0  0  0  0  2  3  0]  
[  0  0  0 196  0  1  0  1  1  1]  
[  1  0  2  0 191  0  0  0  0  6]  
[  1  0  3  1  0 193  2  0  0  0]  
[  1  0  0  0  0  0 198  0  1  0]  
[  0  0  4  0  0  0  0 196  0  0]  
[  0  0  0  1  0  1  3  0 195  0]  
[  0  0  0  0  4  0  0  2  1 193]]
```

Figure 3-9: Extra Trees SVM RBF Kernel

### 3.3.4. SVM Sigmoid Kernel

```

----- Outputs for ExtraTree features -----
-----
SVM_SIGMOID kernel Score = 52.7 %
Time elapsed for SVM Sigmoid:: 5.7334344260000005 sec
Confusion Matix :
[[112  0  0  1  1 79  2  0  5  0]
 [  0 134  0  4  0  1  0  0 61  0]
 [  9 35 105  1  1 16  9  1 15  8]
 [  3 15  3 76  0 81  1  1 18  2]
 [  0  3  1  0 122 37  2  2  5 28]
 [  2 13  1  7  2 108 12  0 47  8]
 [  4  5 12  0  0 76 96  0  4  3]
 [  1  2  4  0  3  2  0 163  2 23]
 [  0 90  1  0  0 50  4  1 54  0]
 [  1  2  0  0 25 53  0 35  0 84]]

```

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

## 5. References

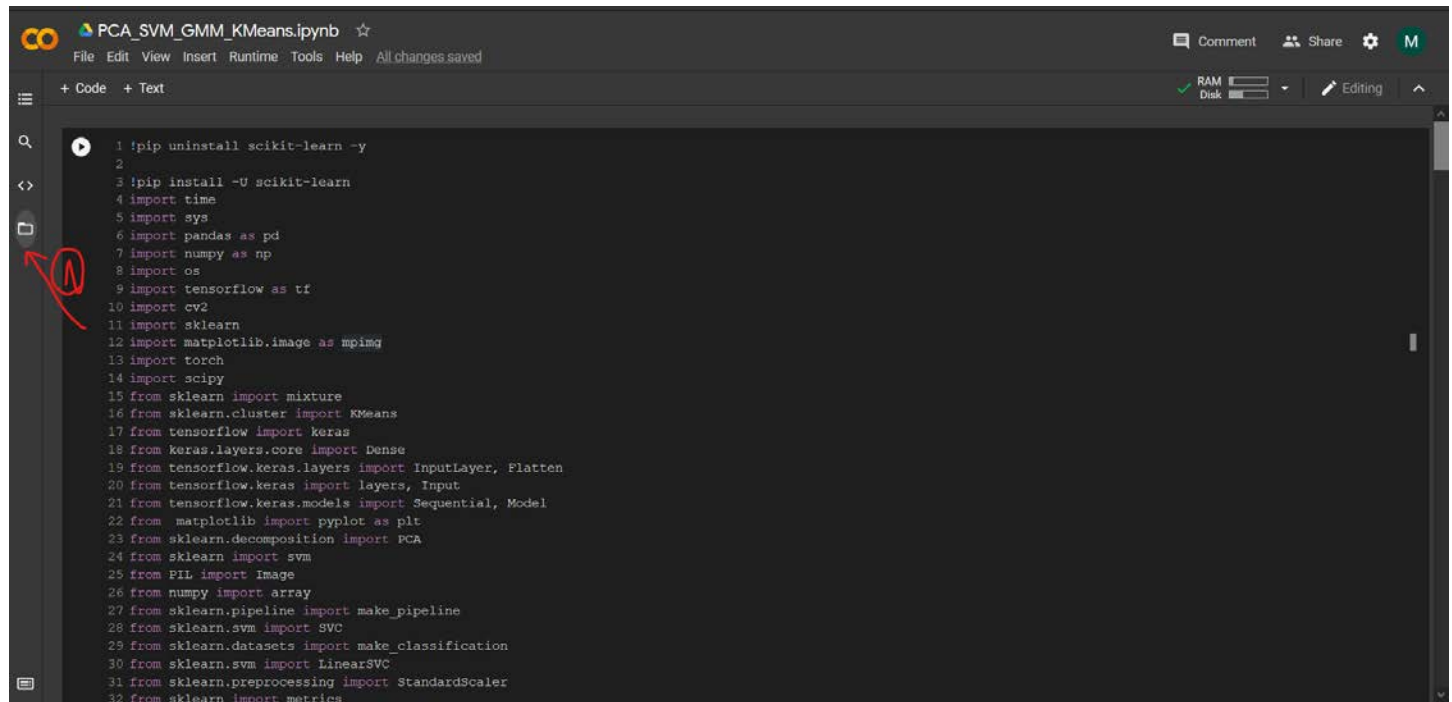
All codes for previous features can be accessed and run directly from google colab.

Steps to run the notebook from a link:

1. Open the Colab link i.e.

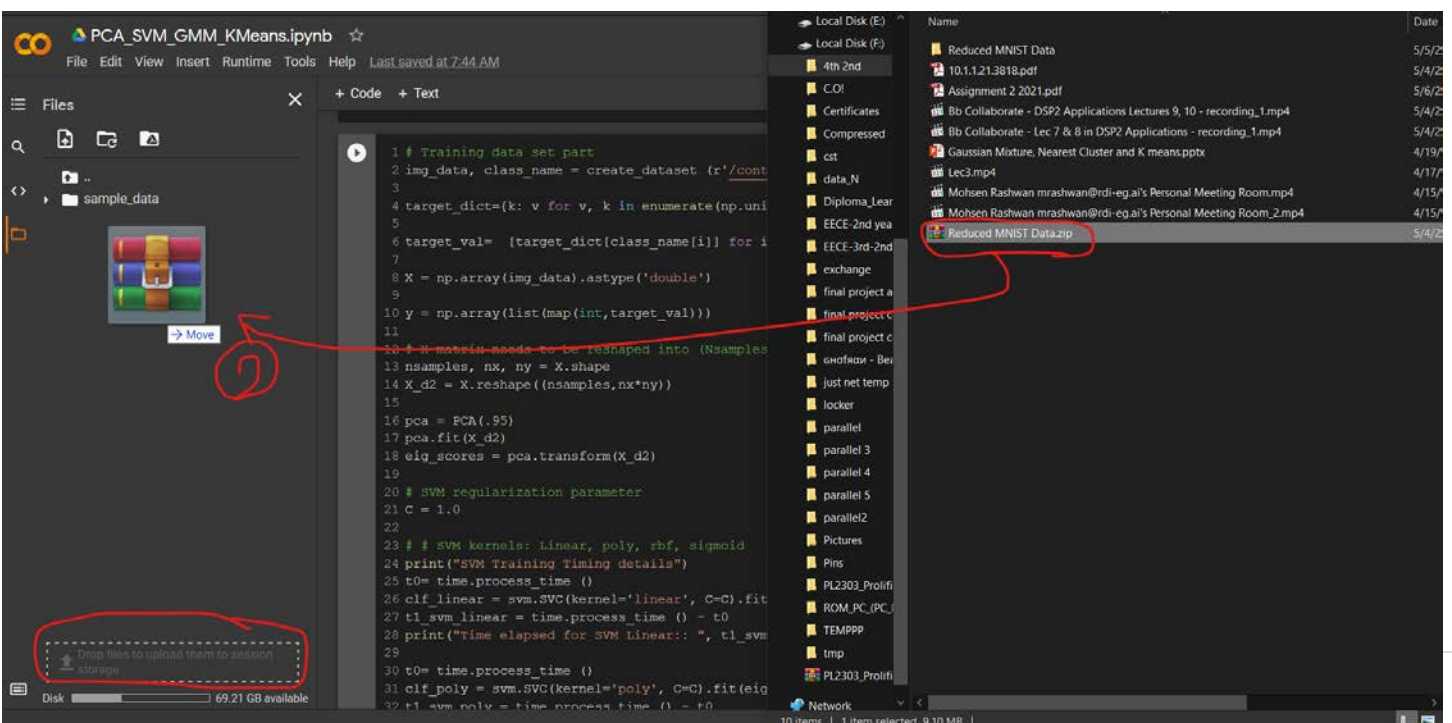
<https://colab.research.google.com/drive/1GxIH6fa2u5ezw8pjsCZcIcQzAIO65xXf#scrollTo=edqNccj6xeMJ>

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

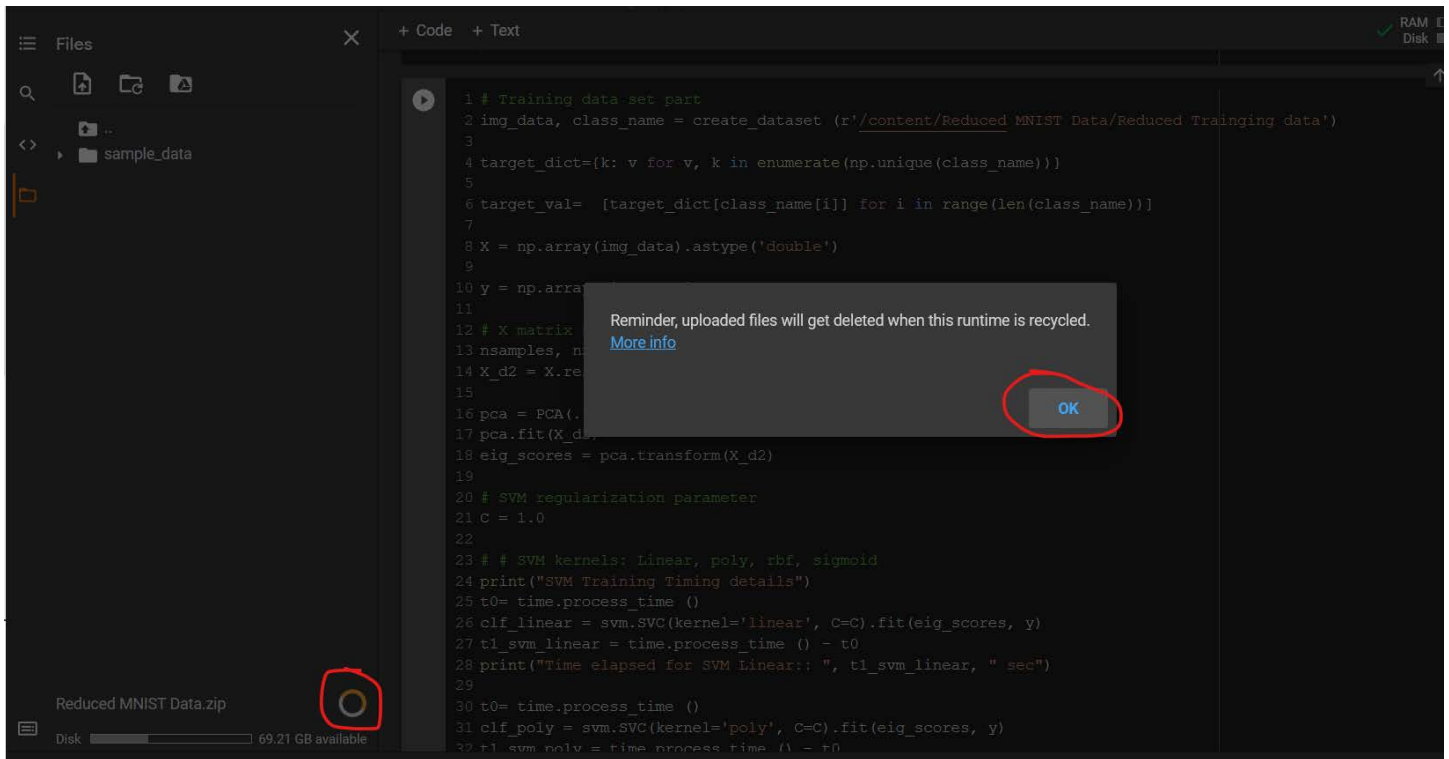


The screenshot shows the Google Colab interface with a notebook titled 'PCA\_SVM\_GMM\_KMeans.ipynb'. The code editor contains a list of imports and library setup commands. A red arrow points to the 'File' menu icon in the top left corner.

```
1 !pip uninstall scikit-learn -y
2
3 !pip install -U scikit-learn
4 import time
5 import sys
6 import pandas as pd
7 import numpy as np
8 import os
9 import tensorflow as tf
10 import cv2
11 import sklearn
12 import matplotlib.image as mpimg
13 import torch
14 import scipy
15 from sklearn import mixture
16 from sklearn.cluster import KMeans
17 from tensorflow import keras
18 from keras.layers.core import Dense
19 from tensorflow.keras.layers import InputLayer, Flatten
20 from tensorflow.keras import layers, Input
21 from tensorflow.keras.models import Sequential, Model
22 from matplotlib import pyplot as plt
23 from sklearn.decomposition import PCA
24 from sklearn import svm
25 from PIL import Image
26 from numpy import array
27 from sklearn.pipeline import make_pipeline
28 from sklearn.svm import SVC
29 from sklearn.datasets import make_classification
30 from sklearn.svm import LinearSVC
31 from sklearn.preprocessing import StandardScaler
32 from sklearn import metrics
```







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=edqNccj6xeMJ>

### 5.2. DCT Features Colab notebook

Link:

[https://colab.research.google.com/drive/1w\\_c0aICJMq0rSem5vjQOIW91cfAl5MoO](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>

## 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'
```

```

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_clustering(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 Training 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

```

```

# # 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_kml6 = time.process_time () - t0
print("Time elapsed for KMeans 16 Clusters:: ", t1_kml6, " 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_kml = time.process_time() - t0
print("Time elapsed for KMeans 1 Clusters:: ", t1_kml, " 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 ()
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 :")

```

```

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_clustering(y, kmeans16.predict(eig_scores), 160)
predicted_labels4 = get_labels_from_clustering(y, kmeans4.predict(eig_scores), 40)
predicted_labels1 = get_labels_from_clustering(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_kml, " 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_kml6, " sec")
print("Confusion Matix :")
cm = confusion_matrix(y,predicted_labels16)
print(cm)
print(" ")

predicted_labels16 = get_labels_from_clustering(y, GMM_1_6.predict(eig_scores), 160)
predicted_labels4 = get_labels_from_clustering(y, GMM_4.predict(eig_scores), 40)
predicted_labels1 = get_labels_from_clustering(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

```

```

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)

```



```

        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_clustering(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
            else:
                h = h + 1
                if (h == hmax):
                    v = v + 1
            i = i + 1
        else:
            v = v + 1
            if (v == vmax):
                h = h + 1
            else:
                v = v + 1
            i = i + 1
    return output

```

```

        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

```

```

    #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 Traingini
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")

```

```

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_kml6 = time.process_time () - t0
print("Time elapsed for KMeans 16 Clusters:: ", t1_kml6, " 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_kml = time.process_time() - t0
print("Time elapsed for KMeans 1 Clusters:: ", t1_kml, " 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 ()

```

```

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")

```

```

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_clustering(y, kmeans16.predict(eig_scores), 160)
predicted_labels4 = get_labels_from_clustering(y, kmeans4.predict(eig_scores), 40)
predicted_labels1 = get_labels_from_clustering(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_kml, " 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_kml6, " sec")
print("Confusion Matix :")
cm = confusion_matrix(y,predicted_labels16)
print(cm)
print(" ")

```

```

predicted_labels16 = get_labels_from_clustering(y, GMM_1_6.predict(eig_scores), 160)
predicted_labels4 = get_labels_from_clustering(y, GMM_4.predict(eig_scores), 40)
predicted_labels1 = get_labels_from_clustering(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

```

```

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

```



```

        img_data_array.append(image)
        class_name.append(dir1)
    return img_data_array, class_name

def get_labels_from_clustering(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

```

```

# # 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_kml6 = time.process_time () - t0
print("Time elapsed for KMeans 16 Clusters:: ", t1_kml6, " 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_kml = time.process_time() - t0
print("Time elapsed for KMeans 1 Clusters:: ", t1_kml, " 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 ()
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("-----")

```

```

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_clustering(y, kmeans16.predict(eig_scores), 160)
predicted_labels4 = get_labels_from_clustering(y, kmeans4.predict(eig_scores), 40)
predicted_labels1 = get_labels_from_clustering(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_kml, " 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) * 100, "%")
print("Time elapsed for KMeans 16 Clusters:: ", t1_kml6, " sec")
print("Confusion Matix :")
cm = confusion_matrix(y,predicted_labels16)
print(cm)
print(" ")

predicted_labels16 = get_labels_from_clustering(y, GMM_1_6.predict(eig_scores), 160)
predicted_labels4 = get_labels_from_clustering(y, GMM_4.predict(eig_scores), 40)
predicted_labels1 = get_labels_from_clustering(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(" ")

```