Q4. Eigenfaces - Face classification using PCA (40 classes)

> **a) Use the following "face.csv" file to classify the faces of 40 different people.**

> **b) Do not use in-built function for implementing PCA.**

> **c) Use appropriate classifier taught in class (any classification algorithm taught in class like Bayes classifier, minimum distance clasifier, and so on)**

> **d) Refer to the following link for a description of the dataset**

> **https://towardsdatascience.com/eigenfaces-face-classification-in-python-7b8d2af3d3ea**

```
import numpy as np
import pandas as pd
```

```
from google.colab import files
uploaded = files.upload()
```

Choose files face.csv

- **face.csv**(text/csv) - 17088890 bytes, last modified: 25/03/2021 - 100% done

```
Saving face.csv to face.csv
```

```
df = pd.read_csv('face.csv')

print("ACTUAL DATASET\n--------------")
print(df)
print('\n-----------------------------------------------------------------------------------------------------------------------------------------')
```

```
ACTUAL DATASET
--------------
            0         1         2  ...      4094      4095  target
0    0.309917  0.367769  0.417355  ...  0.161157  0.157025       0
1    0.454545  0.471074  0.512397  ...  0.152893  0.152893       0
2    0.318182  0.400826  0.491736  ...  0.148760  0.152893       0
3    0.198347  0.194215  0.194215  ...  0.752066  0.739669       0
4    0.500000  0.545455  0.582645  ...  0.173554  0.173554       0
..        ...       ...       ...  ...       ...       ...     ...
395  0.400826  0.495868  0.570248  ...  0.157025  0.136364      39
396  0.367769  0.367769  0.351240  ...  0.512397  0.549587      39
397  0.500000  0.533058  0.607438  ...  0.148760  0.190083      39
398  0.214876  0.219008  0.219008  ...  0.590909  0.603306      39
399  0.516529  0.462810  0.280992  ...  0.355372  0.384298      39

[400 rows x 4097 columns]

-----------------------------------------------------------------------------------------------------------------------------------------
```

```
df.drop(df.columns[-1], axis = 1, inplace = True)
```

```
train_data_set = df.iloc[2:10]

for i in range(10, 400, 10):
  train_data_set = train_data_set.append(df.iloc[i + 2:i + 10], ignore_index = True)

print("TRAIN DATASET WITH DIMENSION = 4096\n----------------------------------")
print(train_data_set)
print('\n----------------------------------------------------------------------------------------------------------------------------------')
```

```
TRAIN DATASET WITH DIMENSION = 4096
-----------------------------------
           0         1         2  ...      4093      4094      4095
0    0.318182  0.400826  0.491736  ...  0.140496  0.148760  0.152893
1    0.198347  0.194215  0.194215  ...  0.752066  0.752066  0.739669
2    0.500000  0.545455  0.582645  ...  0.177686  0.173554  0.173554
3    0.549587  0.545455  0.541322  ...  0.714876  0.706612  0.702479
4    0.330578  0.305785  0.330578  ...  0.144628  0.152893  0.152893
..        ...       ...       ...  ...       ...       ...       ...
315  0.400826  0.495868  0.570248  ...  0.161157  0.157025  0.136364
316  0.367769  0.367769  0.351240  ...  0.487603  0.512397  0.549587
317  0.500000  0.533058  0.607438  ...  0.177686  0.148760  0.190083
318  0.214876  0.219008  0.219008  ...  0.574380  0.590909  0.603306
319  0.516529  0.462810  0.280992  ...  0.359504  0.355372  0.384298

[320 rows x 4096 columns]

----------------------------------------------------------------------------------------------------------------------------------
```

```
test_data_set = df.iloc[0:2]

for i in range(10, 400, 10):
  test_data_set = test_data_set.append(df.iloc[i:i + 2], ignore_index = True)

print("TEST DATASET WITH DIMENSION = 4096\n----------------------------------")
print(test_data_set)
print('\n----------------------------------------------------------------------------------------------------------------------------------')
```

```
TEST DATASET WITH DIMENSION = 4096
----------------------------------
           0         1         2  ...      4093      4094      4095
0   0.309917  0.367769  0.417355  ...  0.152893  0.161157  0.157025
1   0.454545  0.471074  0.512397  ...  0.152893  0.152893  0.152893
2   0.541322  0.586777  0.640496  ...  0.095041  0.111570  0.111570
3   0.644628  0.690083  0.702479  ...  0.111570  0.107438  0.119835
4   0.578512  0.603306  0.632231  ...  0.177686  0.161157  0.152893
..       ...       ...       ...  ...       ...       ...       ...
75  0.144628  0.219008  0.326446  ...  0.309917  0.210744  0.214876
76  0.252066  0.219008  0.227273  ...  0.157025  0.157025  0.185950
77  0.355372  0.392562  0.446281  ...  0.173554  0.181818  0.185950
78  0.545455  0.611570  0.640496  ...  0.173554  0.173554  0.181818
79  0.334711  0.404959  0.475207  ...  0.384298  0.376033  0.384298

[80 rows x 4096 columns]

----------------------------------------------------------------------------------------------------------------------------------
```

```
train_data_set = np.transpose(train_data_set)
```

```
test_data_set = np.transpose(test_data_set)
```

```
d = 4096

print("DIMENSION OF FEATURE VECTORS\n----------------------------")
print("d = " + str(d))
print('\n-----------------------------------------------------------------------------------------------------------------------')
```

```
DIMENSION OF FEATURE VECTORS
----------------------------
d = 4096

-----------------------------------------------------------------------------------------------------------------------
```

```
def mean_PCA(train_data_set):
    train_data_set_mean = np.mean(train_data_set, axis = 1)
    return np.array(train_data_set_mean)
```

```
train_data_set_mean = mean_PCA(train_data_set)

print("MEAN OF TRAIN DATASET\n---------------------")
print(train_data_set_mean)
print('\n-----------------------------------------------------------------------------------------------------------------------')
```

```
MEAN OF TRAIN DATASET
---------------------
[0.3958032  0.43013947 0.47068698 ... 0.33145661 0.3206095  0.31765238]

-----------------------------------------------------------------------------------------------------------------------
```

```
train_data_set_covariance_matrix = np.cov(train_data_set)

print("COVARIANCE MATRIX OF TRAIN DATASET\n---------------------------------")
print(train_data_set_covariance_matrix)
print('\n-----------------------------------------------------------------------------------------------------------------------')
```

```
COVARIANCE MATRIX OF TRAIN DATASET
---------------------------------
[[ 0.03328546  0.03264678  0.02912804 ... -0.00714147 -0.00647714
  -0.0052123 ]
 [ 0.03264678  0.03629656  0.03496155 ... -0.01009454 -0.00915113
  -0.00750577]
 [ 0.02912804  0.03496155  0.03907593 ... -0.01333223 -0.01213405
  -0.01032698]
 ...
 [-0.00714147 -0.01009454 -0.01333223 ...  0.03640213  0.03205362
   0.02864383]
 [-0.00647714 -0.00915113 -0.01213405 ...  0.03205362  0.03469588
   0.03253077]
 [-0.0052123  -0.00750577 -0.01032698 ...  0.02864383  0.03253077
   0.03435278]]

-----------------------------------------------------------------------------------------------------------------------
```

```
eigen_values, eigen_vectors = np.linalg.eigh(train_data_set_covariance_matrix)

eigen_pairs = [(np.abs(eigen_values[i]), eigen_vectors[:, i]) for i in range(len(eigen_values))]

eigen_pairs.sort(key = lambda x : x[0], reverse = True)

print("EIGEN VALUES IN INCREASING ORDER\n--------------------------------")
for i in eigen_pairs:
    print(i[0])
print('\n-----------------------------------------------------------------------------------------------------------------------')
```

```
EIGEN VALUES IN INCREASING ORDER
--------------------------------
19.10579609532137
12.207087351051667
6.166659790621007
3.7667089639907667
2.7302879284954256
2.4915051370167314
1.8374666031635383
1.5976670365069312
1.548968539068521
1.2999297469089832
1.2590005969942248
1.2064132670601169
0.9888074298812582
0.9039823703871799
0.8448375935936389
0.7914776352457463
0.7520090833717974
0.6738264107345473
0.6144049784693244
0.5890530925253853
0.549786559499683
0.49010529977275485
0.4717037695685212
0.46573046423032194
0.4616061844765105
0.41816962860239154
0.41277260412302547
0.3887264695980576
0.36928618319222534
0.34224362968088184
0.32189230221496157
0.30506745385263817
0.2984196735967231
0.2730476287731591
0.2702427520743912
0.26234168424278026
0.25308844869632535
0.2469772015345936
0.23877602487160937
0.22800216195907072
0.21984486761921196
0.21917955393241387
0.20256778179057885
0.1932323181959326
0.18919373732548217
0.18774921089735153
0.1826612824551405
```

```
0.17829927514928332
0.1765633076791835
0.16774808774377675
0.1636749827970262
0.1587019165206993
0.15285677861848895
0.14831918120427018
0.14542047142485356
0.14373850818133344
0.1406424006354548
```

```
def get_reduced_dimension(eigen_pairs, eigen_values, percent):
    eigen_values_sum = np.sum(eigen_values)
    eigen_values_sum_reduced_dimension = percent * eigen_values_sum / 100
    sum = 0.0
    count = 0
    for i in eigen_pairs:
        if(sum < eigen_values_sum_reduced_dimension):
            sum += i[0]
            count += 1
        else:
            break
    return count
```

```
d_prime = get_reduced_dimension(eigen_pairs, eigen_values, 95)

print("REDUCED DIMENSION OF FEATURE VECTORS\n-----------------------------------")
print("d' = " + str(d_prime))
print('\n-----------------------------------------------------------------------------------------------------------------------')
```

```
REDUCED DIMENSION OF FEATURE VECTORS
-----------------------------------
d' = 111

-----------------------------------------------------------------------------------------------------------------------
```

```
matrix_w = eigen_pairs[0][1].reshape(d, 1)

for i in range(1, d_prime):
    matrix_w = np.hstack((matrix_w, eigen_pairs[i][1].reshape(d, 1)))

print("REDUCED DIMENSION MATRIX W\n--------------------------")
print(matrix_w)
print('\n-----------------------------------------------------------------------------------------------------------------------')
```

```
REDUCED DIMENSION MATRIX W
--------------------------
[[-0.00274755  0.02907084  0.00040754 ...  0.0010451   0.05855982
  -0.03557526]
 [-0.00543887  0.03428108 -0.00151739 ...  0.00890516  0.04227196
  -0.03798427]
 [-0.00760075  0.0390819  -0.00233185 ...  0.00400687 -0.00442485
  -0.02779936]
 ...
 [-0.00294621 -0.02996215 -0.01302683 ... -0.00583255 -0.01525501
  -0.02431535]
```

```
 [ 0.0008393  -0.02740332 -0.01025922 ...  0.03723325 -0.02880468
  -0.0042277 ]
 [ 0.0015609  -0.02503931 -0.00867573 ...  0.05040073  0.00684414
   0.03448373]]

-----------------------------------------------------------------------------------------------------------------
```

```
train_data_set_transformed = np.transpose(matrix_w.T.dot(train_data_set))

print("TRAIN DATASET WITH DIMENSION = " + str(d_prime) + "\n----------------------------------")
print(train_data_set_transformed)
print('\n-----------------------------------------------------------------------------------------------------------------')
```

```
TRAIN DATASET WITH DIMENSION = 111
----------------------------------
[[-3.84278500e+01  1.05496914e+01  2.76257385e+00 ... -1.49311834e-01
   2.64169294e-01  4.94872391e-01]
 [-3.80958302e+01 -2.36530844e+00  2.77395344e+00 ...  1.07242465e-01
  -2.50675502e-01  2.44819087e-01]
 [-3.70319559e+01  1.22432369e+01  6.31392191e+00 ... -1.36547366e-01
  -4.01670809e-02  3.49316185e-01]
 ...
 [-3.11444304e+01  1.01462209e+01  5.96225995e+00 ... -6.70871573e-02
  -1.75148272e-01  3.32349524e-01]
 [-3.88338294e+01  9.01930153e-01  1.98477303e+00 ... -2.30756990e-02
  -3.69142681e-01  3.00644319e-01]
 [-3.45162687e+01  6.32423245e+00  3.80672052e+00 ... -3.53049751e-01
   2.48143830e-01  2.12767436e-01]]

-----------------------------------------------------------------------------------------------------------------
```

```
test_data_set_transformed = np.transpose(matrix_w.T.dot(test_data_set))

print("TEST DATASET WITH DIMENSION = " + str(d_prime) + "\n--------------------------------")
print(test_data_set_transformed)
print('\n-----------------------------------------------------------------------------------------------------------------')
```

```
TEST DATASET WITH DIMENSION = 111
--------------------------------
[[-3.95811272e+01  9.73374957e+00  1.16472961e+00 ...  1.14704260e-02
   2.30335788e-02  1.73677045e-01]
 [-3.39897567e+01  1.54595402e+01  4.52888350e+00 ... -4.59040975e-01
  -1.37985001e-01  4.52161510e-01]
 [-3.42602930e+01  9.76087593e+00 -1.34650656e+00 ... -1.48750983e-01
   2.39329589e-01  3.29394425e-01]
 ...
 [-2.30611479e+01  9.42401369e+00  1.71149373e+00 ... -8.64847903e-02
  -4.13596812e-01  5.43861931e-01]
 [-3.18795206e+01  1.02691521e+01  5.97999216e+00 ... -2.37681112e-02
   6.24611657e-02  2.73599124e-01]
 [-3.73080804e+01  6.39686729e+00  4.28431021e+00 ... -2.95259371e-01
  -4.42425011e-01  2.70357075e-01]]

-----------------------------------------------------------------------------------------------------------------
```

Code for Plotting The Graphs

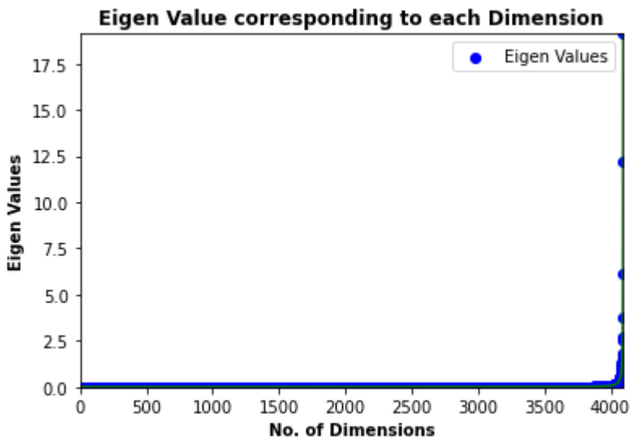
Graph 1 : X-axis - No. of Dimension; Y-axis - Eigen Values

Graph 2 : X-axis - No. of Dimension; Y-axis - Variance

```
import matplotlib.pyplot as plt
```

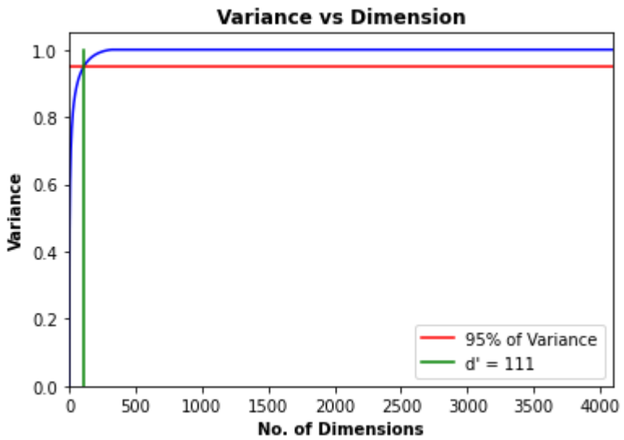
```
plot1 = plt.figure(1)
plt.scatter([i for i in range(d)], eigen_values, label = 'Eigen Values', color = 'blue')
plt.plot([i for i in range(d)], eigen_values, linestyle = '-', color = 'green')

plt.title('Eigen Value corresponding to each Dimension', fontweight = 'bold')
plt.xlabel('No. of Dimensions', fontweight = 'bold')
plt.ylabel('Eigen Values', fontweight = 'bold')

plt.axis([-1, d, -0.01, np.max(eigen_values) + 0.01])
plt.legend(loc = 'upper right')
```

```
<matplotlib.legend.Legend at 0x7f5c78bf6c50>
```




```
plot2 = plt.figure(2)

x_values = []
y_values = []

x_values.append(0)
y_values.append(0.0)

sum = 0.0
total_sum = np.sum(eigen_values)

for i in range(d):
    sum += eigen_pairs[i][0]
    x_values.append(i + 1)
    y_values.append(sum / total_sum)

fp_num = np.linspace(0.0, 1.0, 1000)

plt.plot(x_values, y_values, color = 'blue')
```

```
plt.plot([i for i in range(d)], [0.95 for i in range(d)], linestyle = '-', label = '95' + str('%') + ' of Variance', color = 'red')
plt.plot([d_prime for i in range(len(fp_num))], fp_num, linestyle = '-', label = 'd\' = ' + str(d_prime), color = 'green')

plt.axis([0, d, 0.0, 1.05])

plt.title('Variance vs Dimension', fontweight = 'bold')
plt.xlabel('No. of Dimensions', fontweight = 'bold')
plt.ylabel('Variance', fontweight = 'bold')

plt.legend(loc = 'lower right')
plt.show()
```



Bayes Classifier Code

```
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
```

Bayes Classifier with dimesion = 4096

```
train_data_set = np.transpose(train_data_set)
test_data_set = np.transpose(test_data_set)

X_train = train_data_set
y_train = []

for i in range(40):
    for j in range(int(len(train_data_set)/ 40)):
        y_train.append(str(i))

X_test = test_data_set
y_test = []

for i in range(40):
```

```
    for j in range(int(len(test_data_set)/ 40)):
        y_test.append(str(i))

gnb = GaussianNB()

gnb.fit(X_train, y_train)

y_pred = gnb.predict(X_test)

overall_accuracy = metrics.accuracy_score(y_test, y_pred) * 100

print("BAYES CLASSIFIER - DIMENSION = " + str(d) + "\n----------------------------------")
print("Actual Classification = 0")
print("Bayes Classifier : " + str(y_pred[0:2]))
print('')
print("Actual Classification = 1")
print("Bayes Classifier : " + str(y_pred[2:4]))
print('')
print("Actual Classification = 2")
print("Bayes Classifier : " + str(y_pred[4:6]))
print('')
print("Actual Classification = 3")
print("Bayes Classifier : " + str(y_pred[6:8]))
print('')
print("Actual Classification = 4")
print("Bayes Classifier : " + str(y_pred[8:10]))
print('')
print("Actual Classification = 5")
print("Bayes Classifier : " + str(y_pred[10:12]))
print('')
print("Actual Classification = 6")
print("Bayes Classifier : " + str(y_pred[12:14]))
print('')
print("Actual Classification = 7")
print("Bayes Classifier : " + str(y_pred[14:16]))
print('')
print("Actual Classification = 8")
print("Bayes Classifier : " + str(y_pred[16:18]))
print('')
print("Actual Classification = 9")
print("Bayes Classifier : " + str(y_pred[18:20]))
print('')
print("Actual Classification = 10")
print("Bayes Classifier : " + str(y_pred[20:22]))
print('')
print("Actual Classification = 11")
print("Bayes Classifier : " + str(y_pred[22:24]))
print('')
print("Actual Classification = 12")
print("Bayes Classifier : " + str(y_pred[24:26]))
print('')
print("Actual Classification = 13")
print("Bayes Classifier : " + str(y_pred[26:28]))
print('')
print("Actual Classification = 14")
```

```
print("Bayes Classifier : " + str(y_pred[28:30]))
print('')
print("Actual Classification = 15")
print("Bayes Classifier : " + str(y_pred[30:32]))
print('')
print("Actual Classification = 16")
print("Bayes Classifier : " + str(y_pred[32:34]))
print('')
print("Actual Classification = 17")
print("Bayes Classifier : " + str(y_pred[34:36]))
print('')
print("Actual Classification = 18")
print("Bayes Classifier : " + str(y_pred[36:38]))
print('')
print("Actual Classification = 19")
print("Bayes Classifier : " + str(y_pred[38:40]))
print('')
print("Actual Classification = 20")
print("Bayes Classifier : " + str(y_pred[40:42]))
print('')
print("Actual Classification = 21")
print("Bayes Classifier : " + str(y_pred[42:44]))
print('')
print("Actual Classification = 22")
print("Bayes Classifier : " + str(y_pred[44:46]))
print('')
print("Actual Classification = 23")
print("Bayes Classifier : " + str(y_pred[46:48]))
print('')
print("Actual Classification = 24")
print("Bayes Classifier : " + str(y_pred[48:50]))
print('')
print("Actual Classification = 25")
print("Bayes Classifier : " + str(y_pred[50:52]))
print('')
print("Actual Classification = 26")
print("Bayes Classifier : " + str(y_pred[52:54]))
print('')
print("Actual Classification = 27")
print("Bayes Classifier : " + str(y_pred[54:56]))
print('')
print("Actual Classification = 28")
print("Bayes Classifier : " + str(y_pred[56:58]))
print('')
print("Actual Classification = 29")
print("Bayes Classifier : " + str(y_pred[58:60]))
print('')
print("Actual Classification = 30")
print("Bayes Classifier : " + str(y_pred[60:62]))
print('')
print("Actual Classification = 31")
print("Bayes Classifier : " + str(y_pred[62:64]))
print('')
print("Actual Classification = 32")
print("Bayes Classifier : " + str(y_pred[64:66]))
```

```
print('')
print("Actual Classification = 33")
print("Bayes Classifier : " + str(y_pred[66:68]))
print('')
print("Actual Classification = 34")
print("Bayes Classifier : " + str(y_pred[68:70]))
print('')
print("Actual Classification = 35")
print("Bayes Classifier : " + str(y_pred[70:72]))
print('')
print("Actual Classification = 36")
print("Bayes Classifier : " + str(y_pred[72:74]))
print('')
print("Actual Classification = 37")
print("Bayes Classifier : " + str(y_pred[74:76]))
print('')
print("Actual Classification = 38")
print("Bayes Classifier : " + str(y_pred[76:78]))
print('')
print("Actual Classification = 39")
print("Bayes Classifier : " + str(y_pred[78:80]))
print('')

print("\nOverall Accuracy = " + str(overall_accuracy) + "%\n")
print('----------------------------------------------------------------------------------------------------------------------------')
```

```
     Bayes Classifier : ['21' '21']

     Actual Classification = 22
     Bayes Classifier : ['22' '22']

     Actual Classification = 23
     Bayes Classifier : ['23' '23']

     Actual Classification = 24
     Bayes Classifier : ['24' '24']

     Actual Classification = 25
     Bayes Classifier : ['25' '2']

     Actual Classification = 26
     Bayes Classifier : ['26' '26']

     Actual Classification = 27
     Bayes Classifier : ['27' '27']

     Actual Classification = 28
     Bayes Classifier : ['28' '28']

     Actual Classification = 29
     Bayes Classifier : ['29' '29']

     Actual Classification = 30
     Bayes Classifier : ['30' '30']

     Actual Classification = 31
     Bayes Classifier : ['31' '31']

     Actual Classification = 32
```

```
Actual Classification = 32
Bayes Classifier : ['32' '32']

Actual Classification = 33
Bayes Classifier : ['33' '33']

Actual Classification = 34
Bayes Classifier : ['34' '34']

Actual Classification = 35
Bayes Classifier : ['35' '35']

Actual Classification = 36
Bayes Classifier : ['36' '36']

Actual Classification = 37
Bayes Classifier : ['39' '37']

Actual Classification = 38
Bayes Classifier : ['38' '38']

Actual Classification = 39
Bayes Classifier : ['39' '39']


Overall Accuracy = 90.0%
```

Bayes Classifier with dimesion = 111

```
X_train_reduced_dimension = train_data_set_transformed
y_train_reduced_dimension = []

for i in range(40):
    for j in range(int(len(train_data_set_transformed)/ 40)):
        y_train_reduced_dimension.append(str(i))

X_test_reduced_dimension = test_data_set_transformed
y_test_reduced_dimension = []

for i in range(40):
    for j in range(int(len(test_data_set_transformed)/ 40)):
        y_test_reduced_dimension.append(str(i))

gnb_reduced_dimension = GaussianNB()

gnb_reduced_dimension.fit(X_train_reduced_dimension, y_train_reduced_dimension)

y_pred_reduced_dimension = gnb_reduced_dimension.predict(X_test_reduced_dimension)

overall_accuracy_reduced_dimension = metrics.accuracy_score(y_test_reduced_dimension, y_pred_reduced_dimension) * 100

print("BAYES CLASSIFIER - DIMENSION = " + str(d_prime) + "\n----------------------------------")
print("Actual Classification = 0")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[0:2]))
print('')
```

```
print('')
print("Actual Classification = 1")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[2:4]))
print('')
print("Actual Classification = 2")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[4:6]))
print('')
print("Actual Classification = 3")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[6:8]))
print('')
print("Actual Classification = 4")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[8:10]))
print('')
print("Actual Classification = 5")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[10:12]))
print('')
print("Actual Classification = 6")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[12:14]))
print('')
print("Actual Classification = 7")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[14:16]))
print('')
print("Actual Classification = 8")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[16:18]))
print('')
print("Actual Classification = 9")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[18:20]))
print('')
print("Actual Classification = 10")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[20:22]))
print('')
print("Actual Classification = 11")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[22:24]))
print('')
print("Actual Classification = 12")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[24:26]))
print('')
print("Actual Classification = 13")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[26:28]))
print('')
print("Actual Classification = 14")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[28:30]))
print('')
print("Actual Classification = 15")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[30:32]))
print('')
print("Actual Classification = 16")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[32:34]))
print('')
print("Actual Classification = 17")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[34:36]))
print('')
print("Actual Classification = 18")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[36:38]))
print('')
print("Actual Classification = 19")
```

```
print("Actual Classification = 19")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[38:40]))
print('')
print("Actual Classification = 20")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[40:42]))
print('')
print("Actual Classification = 21")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[42:44]))
print('')
print("Actual Classification = 22")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[44:46]))
print('')
print("Actual Classification = 23")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[46:48]))
print('')
print("Actual Classification = 24")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[48:50]))
print('')
print("Actual Classification = 25")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[50:52]))
print('')
print("Actual Classification = 26")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[52:54]))
print('')
print("Actual Classification = 27")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[54:56]))
print('')
print("Actual Classification = 28")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[56:58]))
print('')
print("Actual Classification = 29")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[58:60]))
print('')
print("Actual Classification = 30")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[60:62]))
print('')
print("Actual Classification = 31")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[62:64]))
print('')
print("Actual Classification = 32")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[64:66]))
print('')
print("Actual Classification = 33")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[66:68]))
print('')
print("Actual Classification = 34")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[68:70]))
print('')
print("Actual Classification = 35")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[70:72]))
print('')
print("Actual Classification = 36")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[72:74]))
print('')
print("Actual Classification = 37")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[74:76]))
```

```
print('')
print("Actual Classification = 38")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[76:78]))
print('')
print("Actual Classification = 39")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[78:80]))
print('')

print("\nOverall Accuracy = " + str(overall_accuracy_reduced_dimension) + "%\n")
print('-----------------------------------------------------------------------------------------------------------------')
```

```
    Bayes Classifier : ['21' '21']

    Actual Classification = 22
    Bayes Classifier : ['22' '22']

    Actual Classification = 23
    Bayes Classifier : ['23' '23']

    Actual Classification = 24
    Bayes Classifier : ['24' '24']

    Actual Classification = 25
    Bayes Classifier : ['25' '25']

    Actual Classification = 26
    Bayes Classifier : ['26' '26']

    Actual Classification = 27
    Bayes Classifier : ['27' '27']

    Actual Classification = 28
    Bayes Classifier : ['28' '28']

    Actual Classification = 29
    Bayes Classifier : ['29' '29']

    Actual Classification = 30
    Bayes Classifier : ['30' '30']

    Actual Classification = 31
    Bayes Classifier : ['31' '31']

    Actual Classification = 32
    Bayes Classifier : ['32' '32']

    Actual Classification = 33
    Bayes Classifier : ['33' '33']

    Actual Classification = 34
    Bayes Classifier : ['7' '34']

    Actual Classification = 35
    Bayes Classifier : ['35' '35']


    Actual Classification = 36
    Bayes Classifier : ['36' '36']

    Actual Classification = 37
    Bayes Classifier : ['37' '37']
```

```
Bayes Classifier : [' 37   37 ]

Actual Classification = 38
Bayes Classifier : ['38' '38']

Actual Classification = 39
Bayes Classifier : ['39' '39']


Overall Accuracy = 97.5%
```

✓ 0s completed at 8:26 PM