Q1. Consider the 128- dimensional feature vectors (d=128) given in the “gender_feature_vectors.csv” file. (2 classes, male and female)

> **a) Use PCA to reduce the dimension from d to d’. (Here d=128)**
>
> **b) Select the appropriate dimension d’ (select d’ S.T first 95% of λ values)**
>
> **c) Use d’ features to classify the test cases (any classification algorithm taught in class like Bayes classifier, minimum distance classifier, and so on)**
>
> **Dataset Specifications:**
>
> > **Total number of samples = 800**
> >
> > **Number of classes = 2 (labeled as “male” and “female”)**
> >
> > **Samples from “1 to 400” belongs to class “male”**
> >
> > **Samples from “401 to 800” belongs to class “female”**
> >
> > **Number of samples per class = 400**
> >
> > **Number of dimensions = 128**
>
> **Use the following information to design classifier:**
>
> > **Number of test cases (first 10 in each class) = 20**
> >
> > **Number of training feature vectors (remaining 390 in each class) = 390**
> >
> > **Number of reduced dimensions = d’ (map 128 to d’ features vector)**

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

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

Choose files gender_featu…vectors.csv

- **gender_feature_vectors.csv**(text/csv) - 1279817 bytes, last modified: 25/03/2021 - 100% done

```
Saving gender_feature_vectors.csv to gender_feature_vectors.csv
```

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

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

```
ACTUAL DATASET
--------------
     Unnamed: 0 Unnamed: 1         0  ...       125       126       127
0             1       male -0.066420  ... -0.076400  0.107497  0.001567
1             2       male -0.030614  ...  0.017638  0.080610 -0.015930
2             3       male -0.096178  ...  0.017391  0.057652  0.086116
3             4       male -0.103057  ... -0.015100  0.161575  0.062462
4             5       male -0.125815  ...  0.028171  0.026041  0.084135
..          ...        ...       ...  ...       ...       ...       ...
795         796     female -0.164731  ...  0.031600  0.019064  0.004384
796         797     female -0.095308  ...  0.030732 -0.083713  0.064970
797         798     female -0.202852  ...  0.037384 -0.006257  0.039977
798         799     female -0.088300  ...  0.009701 -0.016942  0.048071
799         800     female -0.156201  ... -0.010298 -0.028856  0.075323

[800 rows x 130 columns]

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

```
df.drop(df.columns[[0, 1]], axis = 1, inplace = True)
```

```
train_data_set = df.iloc[10:399]
train_data_set = train_data_set.append(df.iloc[409:801], ignore_index = True)

print("TRAIN DATASET WITH DIMENSION = 128\n---------------------------------")
print(train_data_set)
print('\n---------------------------------------------------------------------------------------------------------------------')
```

```
TRAIN DATASET WITH DIMENSION = 128
---------------------------------
            0         1         2  ...       125       126       127
0   -0.101760  0.095119  0.022390  ...  0.045227  0.134832  0.053776
1   -0.126957  0.065444 -0.014750  ... -0.025286 -0.003429  0.057033
2    0.021787  0.047769  0.031156  ... -0.052743  0.034252  0.046343
3   -0.091019  0.042462 -0.061052  ... -0.026397  0.049204 -0.050450
4   -0.082929  0.058382  0.008007  ... -0.026378  0.048825 -0.025185
..        ...       ...       ...  ...       ...       ...       ...
775 -0.164731  0.064301  0.058630  ...  0.031600  0.019064  0.004384
776 -0.095308  0.051095  0.092913  ...  0.030732 -0.083713  0.064970
777 -0.202852  0.037039  0.079731  ...  0.037384 -0.006257  0.039977
778 -0.088300  0.063530  0.049627  ...  0.009701 -0.016942  0.048071
779 -0.156201  0.055165  0.142716  ... -0.010298 -0.028856  0.075323

[780 rows x 128 columns]

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

```
test_data_set = df.iloc[0:10]
test_data_set = test_data_set.append(df.iloc[399:409], ignore_index = True)

print("TEST DATASET WITH DIMENSION = 128\n---------------------------------")
print(test_data_set)
print('\n---------------------------------------------------------------------------------------------------------------------')
```

```
TEST DATASET WITH DIMENSION = 128
---------------------------------
```

```
             0         1         2  ...       125       126       127
0  -0.066420  0.151611  0.027740  ... -0.076400  0.107497  0.001567
1  -0.030614  0.049667  0.008084  ...  0.017638  0.080610 -0.015930
2  -0.096178  0.061127  0.035326  ...  0.017391  0.057652  0.086116
3  -0.103057  0.085044  0.078333  ... -0.015100  0.161575  0.062462
4  -0.125815  0.120046  0.023131  ...  0.028171  0.026041  0.084135
5  -0.149119  0.125288  0.142323  ... -0.141460  0.019018  0.085765
6  -0.139035  0.073513 -0.001770  ...  0.024194  0.062180  0.036039
7  -0.074126 -0.000669  0.004166  ... -0.026687 -0.017523 -0.038310
8  -0.166220  0.042769 -0.031647  ... -0.101063  0.061373  0.062176
9  -0.185770  0.154008  0.073184  ...  0.012158  0.032304  0.085996
10  0.039844  0.070357  0.130196  ...  0.080940  0.011467 -0.021999
11  0.001747  0.185678  0.073260  ... -0.017690  0.067028  0.036452
12 -0.091598  0.095340  0.072125  ...  0.016900 -0.081676  0.022809
13 -0.018751  0.088572  0.068894  ...  0.045943  0.010856  0.100522
14 -0.130889  0.093262  0.122244  ... -0.029204 -0.020707  0.031028
15 -0.037433  0.078158  0.118061  ... -0.032240  0.037601 -0.020016
16 -0.048322  0.063833  0.110804  ...  0.016019  0.016852  0.140859
17 -0.102973  0.046464  0.019684  ... -0.088858  0.049312  0.019009
18 -0.134824  0.093314  0.103505  ... -0.102133  0.014161  0.011314
19 -0.086950  0.104945  0.093125  ... -0.081236  0.073335  0.056886

[20 rows x 128 columns]

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

```
train_data_set = np.transpose(train_data_set)

test_data_set = np.transpose(test_data_set)
```

```
d = 128

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

```
DIMENSION OF FEATURE VECTORS
----------------------------
d = 128

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

```
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
---------------------
[-0.10448215  0.0846967   0.06285182 -0.05377628 -0.10830336 -0.01167129
 -0.00994616 -0.11886804  0.15334178 -0.07944194  0.18306372 -0.0453969
```

```
 -0.25141473 -0.02430834 -0.03412226  0.13462243 -0.15832291 -0.14540593
 -0.10982075 -0.07133358  0.01157879  0.04059486  0.0326039   0.03006838
 -0.1405689  -0.31671607 -0.06716424 -0.08551999  0.04336038 -0.07480046
  0.01769201  0.0670517  -0.18083406 -0.0405829   0.03638948  0.07284776
 -0.05261509 -0.08427783  0.20824122  0.01069204 -0.19344419 -0.0086605
  0.06124188  0.23737607  0.20656424 -0.00951447  0.02057955 -0.0752162
  0.11976795 -0.27134077  0.03545796  0.14539967  0.10378021  0.07455416
  0.07328733 -0.15244713  0.01723554  0.13915943 -0.18769971  0.06354377
  0.06837423 -0.10685182 -0.0376842  -0.03818284  0.16364505  0.09341991
 -0.09944617 -0.14446323  0.17806468 -0.14802508 -0.04075468  0.08026415
 -0.0970617  -0.16725921 -0.25098869  0.02069878  0.37289889  0.12125218
 -0.16338624  0.01543119 -0.07524929 -0.00924997  0.01921729  0.05471224
 -0.06624514 -0.04771193 -0.09156152  0.0181578   0.20037147 -0.03194873
 -0.01953999  0.22709499  0.0092173  -0.01161148  0.04122744  0.05648975
 -0.11785021 -0.00878284 -0.11891118 -0.02018789  0.03625375 -0.08768928
  0.00958594  0.09198727 -0.17684283  0.16309919 -0.02626004 -0.02582713
 -0.00473226 -0.03358308 -0.07092522 -0.0105671   0.16842126 -0.24869811
  0.19738621  0.17303481  0.00344205  0.14384933  0.05831965  0.05650954
  0.0077072  -0.00974738 -0.13218422 -0.10290807  0.03786203 -0.02808483
  0.01126394  0.03291232]

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

```python
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
----------------------------------
[[ 2.95869391e-03 -3.68920510e-04 -2.38206410e-04 ...  4.13989632e-04
  -1.15668930e-04 -3.17586004e-05]
 [-3.68920510e-04  2.48361909e-03  3.39760593e-05 ... -3.87729269e-04
  -1.52644986e-04 -3.33336342e-05]
 [-2.38206410e-04  3.39760593e-05  2.58699428e-03 ... -3.14120189e-04
  -1.94740059e-04  2.17204952e-05]
 ...
 [ 4.13989632e-04 -3.87729269e-04 -3.14120189e-04 ...  2.34645158e-03
   1.15149015e-05  1.51869998e-04]
 [-1.15668930e-04 -1.52644986e-04 -1.94740059e-04 ...  1.15149015e-05
   2.44623721e-03  2.32836604e-05]
 [-3.17586004e-05 -3.33336342e-05  2.17204952e-05 ...  1.51869998e-04
   2.32836604e-05  2.25952471e-03]]

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

```python
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
--------------------------------
0.04114896784526045
0.024219938485601743
0.01693009563154853
0.014519681570422775
0.012681332012793407
0.011769922572414025
0.010135807809185195
0.00911486678824767
0.008497732688160993
0.007983773751324325
0.007583311540613215
0.007170423103137166
0.0066762566195137495
0.00653850367879149
0.006306909862299599
0.005799921830560707
0.00557574008725447
0.005464143639838317
0.00527779483220967
0.005037880644236138
0.004793053744075445
0.004520346159081878
0.0044772720193894
0.004262265573473648
0.004192017068120541
0.004020535930228428
0.003816060302637334
0.003571591278761981
0.0034214114286942113
0.003317654659647653
0.0032297768986479882
0.00307048335601501
0.003028819480253576
0.0028529575155442238
0.002804243288295685
0.002708628302476543
0.002636417751408717
0.002516983108865679
0.002489503862698147
0.002392950098873742
0.0023798749379336727
0.002299285387925611
0.0022646261815810515
0.0021228689303826338
0.002032466921556844
0.001979086797269327
0.0019180637632538885
0.0018398973114664393
0.0018213152097611759
0.0017803961327181983
0.0016351106791738584
0.0015864271622794803
0.0015417123426378523
0.0014769893520887813
0.001427273337521619
0.001355922590954152
0.001306380270523626
0.0012240113328513826
```

```
def get_reduced_dimension(eigen_pairs, eigen_values, percent):
```

```
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' = 57

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

```
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
--------------------------
[[-0.07544724  0.06338625 -0.05958283 ...  0.13294506 -0.11575339
   0.02574427]
 [-0.03816975 -0.00993477  0.02846919 ...  0.0218059   0.01089124
   0.08336985]
 [ 0.09262621 -0.03630104 -0.11938941 ...  0.02284758 -0.00228281
   0.15058825]
 ...
 [-0.01555627 -0.00281145  0.03443676 ...  0.01955619 -0.11439354
  -0.07877596]
 [-0.04721488 -0.02785616 -0.08216391 ...  0.05315518 -0.15282619
   0.08812416]
 [-0.03216387  0.07497542 -0.06468661 ... -0.04518652 -0.08575702
   0.13859976]]

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

```
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 = 57
---------------------------------
[[ 0.16201447  0.03870741 -0.13590151 ... -0.0875488   0.18615724
  -0.03118321]
 [ 0.16369729  0.09623356  0.26956394 ... -0.13118405  0.16424507
  -0.03418276]
 [ 0.13568836  0.33517518  0.14235416 ... -0.04097319  0.20750583
  -0.07148731]
 ...
 [ 0.49782625  0.35616263  0.23839154 ... -0.05620864  0.2027782
   0.00444246]
 [ 0.53388546  0.38099249  0.1277724  ... -0.0451492   0.22881357
  -0.03173833]
 [ 0.54512133  0.05387415  0.26936994 ... -0.06487287  0.18464931
  -0.04050429]]

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

```
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 = 57
--------------------------------
[[ 0.09864139  0.18567741  0.18802642 ... -0.09239422  0.15614565
   0.0072273 ]
 [ 0.1110883   0.11430068  0.32885361 ... -0.09853505  0.20004937
  -0.05611838]
 [ 0.13898797  0.06899293  0.1259554  ... -0.08764153  0.21843502
   0.02000268]
 ...
 [ 0.57546444  0.02622321 -0.0915027  ... -0.09526622  0.20086673
   0.01346464]
 [ 0.51033098  0.32639377  0.20469761 ... -0.08536315  0.18382309
   0.01236624]
 [ 0.52183469  0.0644349  -0.01447099 ... -0.02715599  0.19544759
   0.08034389]]

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

Code for Plotting The Graphs

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

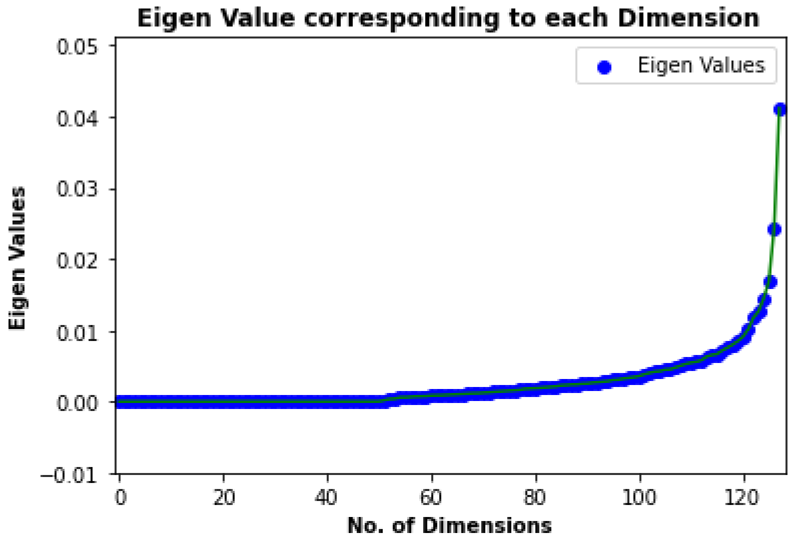
Graph 2 : X-axis - No. of Dimension; Y-axis - Variance

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

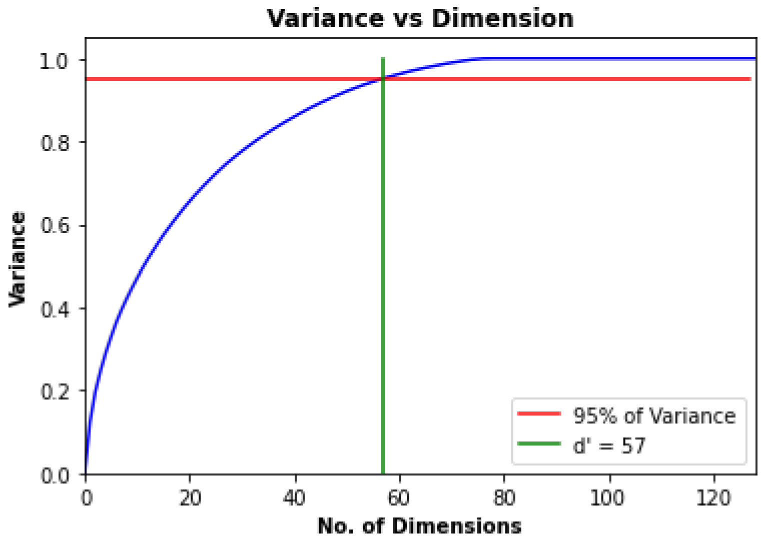
```
plot1 = plt.figure(1)
plt.scatter([i for i in range(d)], eigen_values, label = 'Eigen Values', color = 'blue')
plt.plot([i for i in range(d)], eigen_values, linestyle = '-', color = 'green')
```

```
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 0x7f960e5f1ad0>
```




```
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 = 128

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

X_train = train_data_set
y_train = []

variety = ['Male', 'Female']

for i in range(2):
    for j in range(int(len(train_data_set)/ 2)):
        y_train.append(variety[i])

X_test = test_data_set
y_test = []

for i in range(2):
    for j in range(int(len(test_data_set)/ 2)):
        y_test.append(variety[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 = Male")
print("Bayes Classifier : " + str(y_pred[0:10]))
print('')
print("Actual Classification = Female")
print("Bayes Classifier : " + str(y_pred[10:20]))
print('')

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

```
BAYES CLASSIFIER - DIMENSION = 128
----------------------------------
Actual Classification = Male
Bayes Classifier : ['Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Female' 'Male' 'Male']

Actual Classification = Female
Bayes Classifier : ['Male' 'Male' 'Female' 'Female' 'Female' 'Female' 'Female' 'Female'
 'Female' 'Female']


Overall Accuracy = 85.0%

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

Bayes Classifier with dimesion = 57

```
X_train_reduced_dimension = train_data_set_transformed
y_train_reduced_dimension = []

variety = ['Male', 'Female']

for i in range(2):
    for j in range(int(len(train_data_set_transformed)/ 2)):
        y_train_reduced_dimension.append(variety[i])

X_test_reduced_dimension = test_data_set_transformed
y_test_reduced_dimension = []

for i in range(2):
    for j in range(int(len(test_data_set_transformed)/ 2)):
        y_test_reduced_dimension.append(variety[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 = Male")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[0:10]))
print('')
print("Actual Classification = Female")
print("Bayes Classifier : " + str(y_pred_reduced_dimension[10:20]))
print('')

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

```
BAYES CLASSIFIER - DIMENSION = 57
---------------------------------
Actual Classification = Male
Bayes Classifier : ['Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Female' 'Male' 'Male']

Actual Classification = Female
Bayes Classifier : ['Male' 'Male' 'Female' 'Female' 'Female' 'Female' 'Female' 'Female'
 'Female' 'Female']


Overall Accuracy = 85.0%

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

Confusion Matrix

```
matrix = metrics.plot_confusion_matrix(gnb_reduced_dimension, X_test_reduced_dimension, y_test_reduced_dimension, cmap = plt.cm.Blues)

plt.title('Confusion Matrix')
plt.show()
```