GROUP 10

R SHREJA - COE18B043

RITHIC KUMAR N - COE18B044

SREEDHAR ARUMUGAM - COE18B051

1. 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) Display the eigenvalue based on increasing order, select the d’ of the corresponding eigenvector which is the appropriate dimension d’ (select d’ S.T first 95% of λ values of the covariance matrix are considered).

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)

In [1]:

```
import pandas as pd
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
```

In [2]:

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

In [3]:

```
df
```

Out[3]:

	Unnamed: 0	Unnamed: 1	0	1	2	3	4	5
0	1	male	-0.066420	0.151611	0.027740	0.052771	-0.066105	-0.041232
1	2	male	-0.030614	0.049667	0.008084	-0.050324	0.007649	-0.063818
2	3	male	-0.096178	0.061127	0.035326	-0.035388	-0.090728	-0.018634
3	4	male	-0.103057	0.085044	0.078333	-0.035873	-0.028163	0.004924
4	5	male	-0.125815	0.120046	0.023131	-0.042901	0.038215	-0.049677
...	...	...	...	...	...	...	...	...
795	796	female	-0.164731	0.064301	0.058630	-0.017420	-0.157600	-0.022536
796	797	female	-0.095308	0.051095	0.092913	-0.101745	-0.083153	-0.028159
797	798	female	-0.202852	0.037039	0.079731	-0.047156	-0.140062	-0.080246
798	799	female	-0.088300	0.063530	0.049627	-0.026011	-0.172773	0.086218
799	800	female	-0.156201	0.055165	0.142716	-0.115393	-0.128982	-0.139830

800 rows × 130 columns

In [4]:

```
df['Unnamed: 1'].value_counts()
```

Out[4]:

```
female    401
male      399
Name: Unnamed: 1, dtype: int64
```

In [5]:

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

Out[5]:

	Unnamed: 1	0	1	2	3	4	5	6	
0	male	-0.066420	0.151611	0.027740	0.052771	-0.066105	-0.041232	-0.002637	-0
1	male	-0.030614	0.049667	0.008084	-0.050324	0.007649	-0.063818	-0.019530	-0
2	male	-0.096178	0.061127	0.035326	-0.035388	-0.090728	-0.018634	-0.024315	-0
3	male	-0.103057	0.085044	0.078333	-0.035873	-0.028163	0.004924	0.007829	-0
4	male	-0.125815	0.120046	0.023131	-0.042901	0.038215	-0.049677	-0.054258	-0
...	...	...	...	...	...	...	...	...	
795	female	-0.164731	0.064301	0.058630	-0.017420	-0.157600	-0.022536	0.002864	-0
796	female	-0.095308	0.051095	0.092913	-0.101745	-0.083153	-0.028159	0.009090	-0
797	female	-0.202852	0.037039	0.079731	-0.047156	-0.140062	-0.080246	0.057668	-0
798	female	-0.088300	0.063530	0.049627	-0.026011	-0.172773	0.086218	0.042710	-0
799	female	-0.156201	0.055165	0.142716	-0.115393	-0.128982	-0.139830	-0.037305	-0

800 rows × 129 columns

In [6]:

```
df.rename(columns={df.columns[0]:"Class"},inplace=True)
df
```

Out[6]:

	Class	0	1	2	3	4	5	6	
0	male	-0.066420	0.151611	0.027740	0.052771	-0.066105	-0.041232	-0.002637	-0.158
1	male	-0.030614	0.049667	0.008084	-0.050324	0.007649	-0.063818	-0.019530	-0.119
2	male	-0.096178	0.061127	0.035326	-0.035388	-0.090728	-0.018634	-0.024315	-0.139
3	male	-0.103057	0.085044	0.078333	-0.035873	-0.028163	0.004924	0.007829	-0.017
4	male	-0.125815	0.120046	0.023131	-0.042901	0.038215	-0.049677	-0.054258	-0.130
...	...	...	...	...	...	...	...	...	
795	female	-0.164731	0.064301	0.058630	-0.017420	-0.157600	-0.022536	0.002864	-0.072
796	female	-0.095308	0.051095	0.092913	-0.101745	-0.083153	-0.028159	0.009090	-0.114
797	female	-0.202852	0.037039	0.079731	-0.047156	-0.140062	-0.080246	0.057668	-0.122
798	female	-0.088300	0.063530	0.049627	-0.026011	-0.172773	0.086218	0.042710	-0.161
799	female	-0.156201	0.055165	0.142716	-0.115393	-0.128982	-0.139830	-0.037305	-0.101

800 rows × 129 columns

In [7]:

```
df_test=df.iloc[np.r_[390:400, 790:800]]
```

In [8]:

```
df_test
```

Out[8]:

	Class	0	1	2	3	4	5	6	
390	male	-0.080683	0.097440	0.006550	0.018112	-0.114999	0.160041	-0.002373	-0.057
391	male	-0.010301	0.135185	0.049710	-0.046424	-0.041742	0.016607	-0.041778	-0.048
392	male	-0.112450	0.080098	0.030571	0.000952	-0.097450	-0.045070	-0.003641	-0.142
393	male	-0.087855	0.100264	0.069775	0.037204	-0.047182	-0.047233	-0.013604	-0.189
394	male	-0.139066	0.141988	0.070456	-0.003518	-0.065637	-0.037767	-0.094195	-0.195
395	male	-0.129449	0.132177	0.055916	-0.009390	-0.080541	-0.072362	-0.067433	-0.192
396	male	-0.158460	0.109948	0.019088	0.015506	-0.069668	0.032311	0.015062	-0.140
397	male	-0.101499	0.119739	0.016951	-0.013677	-0.055524	0.028399	0.028164	-0.152
398	male	-0.149516	0.081588	0.090796	-0.053116	-0.133314	0.001096	0.019941	-0.117
399	female	0.039844	0.070357	0.130196	-0.007683	-0.077825	-0.021298	-0.024133	-0.085
790	female	-0.184166	0.122040	0.064143	-0.116653	-0.140633	0.041252	0.014207	-0.108
791	female	-0.128052	0.082184	0.148978	-0.074984	-0.113145	-0.031003	0.018468	-0.130
792	female	-0.109618	0.094507	0.043757	-0.162004	-0.191951	0.027582	-0.033507	-0.112
793	female	-0.201133	0.068051	0.088596	-0.084936	-0.055628	-0.069027	-0.003922	-0.191
794	female	-0.185053	0.136105	0.096279	-0.139661	-0.244312	0.015234	-0.007418	-0.077
795	female	-0.164731	0.064301	0.058630	-0.017420	-0.157600	-0.022536	0.002864	-0.072
796	female	-0.095308	0.051095	0.092913	-0.101745	-0.083153	-0.028159	0.009090	-0.114
797	female	-0.202852	0.037039	0.079731	-0.047156	-0.140062	-0.080246	0.057668	-0.122
798	female	-0.088300	0.063530	0.049627	-0.026011	-0.172773	0.086218	0.042710	-0.161
799	female	-0.156201	0.055165	0.142716	-0.115393	-0.128982	-0.139830	-0.037305	-0.101

20 rows × 129 columns

In [9]:

```
df_train=df.iloc[np.r_[0:390,400:790]]
```

In [10]:

```
df_train
```

Out[10]:

	Class	0	1	2	3	4	5	6	
0	male	-0.066420	0.151611	0.027740	0.052771	-0.066105	-0.041232	-0.002637	-0.158
1	male	-0.030614	0.049667	0.008084	-0.050324	0.007649	-0.063818	-0.019530	-0.119
2	male	-0.096178	0.061127	0.035326	-0.035388	-0.090728	-0.018634	-0.024315	-0.139
3	male	-0.103057	0.085044	0.078333	-0.035873	-0.028163	0.004924	0.007829	-0.017
4	male	-0.125815	0.120046	0.023131	-0.042901	0.038215	-0.049677	-0.054258	-0.130
...	...	...	...	...	...	...	...	...	
785	female	-0.017931	0.045591	0.059826	-0.059833	-0.126832	0.070365	0.026347	-0.093
786	female	-0.039572	0.049409	0.100950	-0.092940	-0.152356	-0.057714	-0.000538	-0.115
787	female	-0.015170	-0.001116	0.132143	-0.078673	-0.134462	-0.155683	-0.030665	-0.164
788	female	-0.151263	0.096725	0.075206	-0.047269	-0.233372	-0.019918	-0.072370	-0.118
789	female	-0.179743	0.104404	0.052260	-0.074608	-0.166008	0.023287	0.006970	-0.149

780 rows × 129 columns

Covariance

In [11]:

```
df_dim=df_train[[i for i in df.columns[1:]]]
```

In [12]:

```
df_dim.shape
```

Out[12]:

```
(780, 128)
```

In [13]:

```
cov=np.cov(df_dim.T)
cov
```

Out[13]:

```
array([[ 2.95108480e-03, -3.70158184e-04, -2.16257469e-04, ...,
         4.34218359e-04, -1.10121464e-04, -2.23638279e-05],
       [-3.70158184e-04,  2.50417089e-03,  6.23082137e-05, ...,
        -3.84633702e-04, -1.45010282e-04, -2.40364718e-05],
       [-2.16257469e-04,  6.23082137e-05,  2.60834145e-03, ...,
        -3.21716028e-04, -2.00431637e-04,  1.91564799e-05],
       ...,
       [ 4.34218359e-04, -3.84633702e-04, -3.21716028e-04, ...,
         2.36909122e-03,  1.84584670e-06,  1.79111827e-04],
       [-1.10121464e-04, -1.45010282e-04, -2.00431637e-04, ...,
         1.84584670e-06,  2.49192480e-03,  5.51925391e-05],
       [-2.23638279e-05, -2.40364718e-05,  1.91564799e-05, ...,
         1.79111827e-04,  5.51925391e-05,  2.26553560e-03]])
```

In [14]:

```
cov.shape
```

Out[14]:

```
(128, 128)
```

getting eigen values

In [15]:

```
def sorted_eig(A):  # For now we sort 'by convention'. For PCA the sorting is key.
    lambdas, vs = np.linalg.eig(A)
    # Next line just sorts values & vectors together in order of decreasing eigenvalues
    lambdas, vs = zip(*sorted(zip(list(lambdas), list(vs.T)),key=lambda x: x[0], revers
e=True))
    return lambdas, np.array(vs).T  # un-doing the list-casting from the previous line
```

In [16]:

```
lambdas,vs=sorted_eig(cov)
print("Lambda values in sorted order")
lambdas
```

Out[16]:

```
(0.04137574503460549,
 0.02360849711562386,
 0.017077038743770187,
 0.014451291776124938,
 0.012633313809123732,
 0.011714705691204017,
 0.010189552276534721,
 0.009087057091511646,
 0.008482770886136562,
 0.008039865359807597,
 0.00760030905912849,
 0.007145776669589233,
 0.006771403338885542,
 0.006509757015262208,
 0.0062636178632257625,
 0.005842316913291985,
 0.005564646916617724,
 0.005510877415737265,
 0.005215948451115262,
 0.005052886212394662,
 0.004814955569113215,
 0.004563226306426189,
 0.0044474704013512225,
 0.004268294677809414,
 0.004229219580396081,
 0.003926267119388022,
 0.0038579422169492264,
 0.003691394876422616,
 0.003477780068960983,
 0.0033675627685272884,
 0.003202721657676988,
 0.0030656341277170433,
 0.0030408421988791347,
 0.0028325439794659106,
 0.0027758324222281914,
 0.002700415370836653,
 0.0026746600113390874,
 0.0025363760389129028,
 0.002515055042077224,
 0.0024227300289765487,
 0.002357721538014705,
 0.002321156393347481,
 0.0022633127203457093,
 0.002177699886331758,
 0.0020620451101881617,
 0.0020097613332662047,
 0.0019102064874178648,
 0.001856074146743188,
 0.0018327018105405322,
 0.0017736034036216462,
 0.0016454665395786428,
 0.0016042741311068861,
 0.0015387653014735222,
 0.001473453780781087,
 0.0014437964392814296,
 0.0013681760840532897,
 0.0012991285501252154,
 0.0012360134625372231,
 0.0011793992314895738,
```

0.0011756810795875475,
0.0011101014875846825,
0.0010686852603078684,
0.0010080061093842574,
0.0009981204169890476,
0.0009303374948736985,
0.0008861924584951544,
0.0008578262872595781,
0.0008314607222984141,
0.0007511239148571716,
0.0007304558969953307,
0.0007090358458285729,
0.0006463624868829558,
0.0005869581784766859,
0.0005630997907502456,
0.00038966919693531906,
0.0003491875729506964,
0.00017423761449379173,
4.589566755200929e-06,
8.578427835079057e-07,
3.059268608435976e-07,
9.355187331571268e-08,
9.061524745412814e-09,
1.23602304949623e-09,
5.425056669644469e-10,
2.7954655365346586e-10,
9.449981920708562e-11,
2.3908232170159416e-11,
9.426540816467547e-12,
4.18888495467226e-12,
2.5704593491045257e-12,
7.83886053141938e-13,
6.746061771312841e-13,
3.000746804666171e-13,
1.6010753954830057e-13,
7.979081251285144e-14,
5.3879530947440807e-14,
4.361637600750955e-14,
3.0905150060949546e-14,
2.0321363896017388e-14,
1.1463095068504483e-14,
1.0643313860501104e-14,
8.132710422215743e-15,
6.9957037506392524e-15,
6.65297665108662e-15,
6.286844255079942e-15,
5.6095001472050975e-15,
5.328087543481544e-15,
5.071977414370759e-15,
4.9223449211747615e-15,
4.624848293407667e-15,
4.398807452238238e-15,
4.171949273130997e-15,
3.796127583127979e-15,
3.60845946278965e-15,
3.199356133416859e-15,
2.95836610690943e-15,
2.85193636134112e-15,
2.725376477179258e-15,
2.7001704617578923e-15,
2.4056221409956062e-15,

```
 2.3629505630551914e-15,
 2.177118779513895e-15,
 2.0409126404096537e-15,
 1.9264089736652567e-15,
 1.7448796144443188e-15,
 1.586320349179776e-15,
 1.369477040839034e-15,
 1.142219401624541e-15)
```

In [17]:

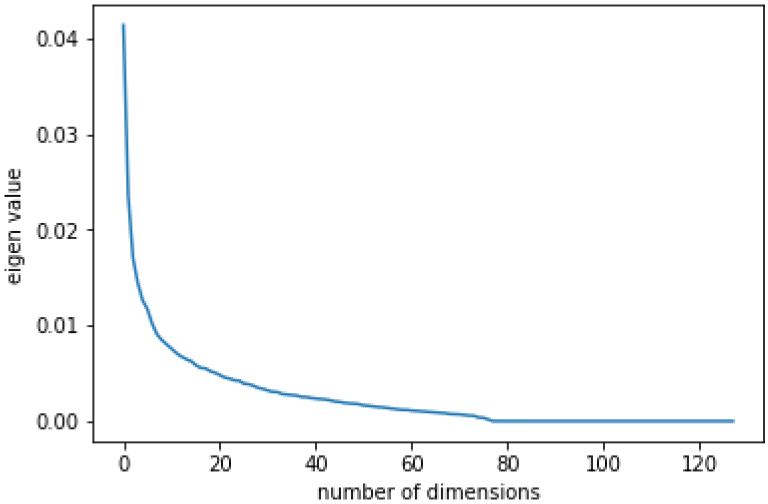
```
vs
```

Out[17]:

```
array([[-0.07127427, -0.06435239, -0.05941313, ...,  0.01756089,
        -0.06551164,  0.03609231],
       [-0.03849229,  0.00910902,  0.02515185, ...,  0.06466022,
        -0.1574927 ,  0.01232222],
       [ 0.09235571,  0.03575751, -0.11693621, ..., -0.03149788,
         0.0080926 ,  0.07226275],
       ...,
       [-0.01535881, -0.00290511,  0.04327725, ...,  0.10022143,
        -0.01055383, -0.12365049],
       [-0.04780239,  0.02650115, -0.08556776, ..., -0.02240944,
        -0.04477904, -0.18795663],
       [-0.03681228, -0.07247092, -0.07462774, ..., -0.02856651,
        -0.08375543, -0.028343  ]])
```

In [18]:

```
plt.plot(lambdas)
plt.xlabel('number of dimensions')
plt.ylabel('eigen value');
```



Getting no of components

In [19]:

```
#consider values which explain 95% of variance

print("Total lambda values:",len(lambdas))

total_eigen=np.sum(lambdas)
r_sum=0

for i in range(len(lambdas)):
    if(r_sum/total_eigen >=.95):
        index=i
        break
    else:
        r_sum+=lambdas[i]

lambda_rd=lambdas[:index]
vs_rd=vs[:index]
```

```
Total lambda values: 128
```

In [20]:

```
print("Reduced lambda ",len(lambda_rd))
```

```
Reduced lambda  57
```

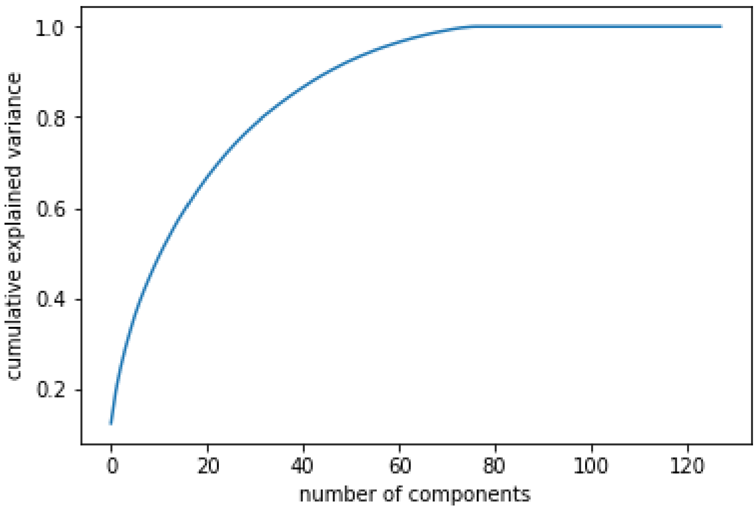
Verifying with inbuilt PCA for no of components

In [21]:

```
from sklearn.decomposition import PCA

pca = PCA().fit(df_dim)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');

#we can see for 0.95, we get around 60
```



Reducing the data

In [22]:

```
red_train_data=df_dim @ vs_rd.T
red_train_data
```

Out[22]:

	0	1	2	3	4	5	6	7	
0	-0.085922	-0.200630	0.193114	0.162222	0.005716	0.034162	0.029775	-0.081066	0.2
1	-0.055755	-0.124344	0.151680	0.157961	0.018568	0.030559	0.088235	-0.114077	0.2
2	-0.116070	-0.119059	0.114342	0.098051	0.045468	0.087185	0.125157	-0.029765	0.2
3	-0.082357	-0.187135	0.187303	0.177185	0.044054	-0.014291	0.103310	-0.039566	0.2
4	-0.076012	-0.152521	0.146189	0.112002	-0.001292	-0.000156	0.106822	-0.039447	0.1
...	...	...	...	...	...	...	...	...	
785	-0.166122	-0.143068	0.166056	0.105643	-0.020830	0.119008	0.096983	-0.106037	0.2
786	-0.018686	-0.175362	0.225020	0.080913	-0.040170	0.181357	0.142460	-0.044167	0.2
787	-0.078697	-0.162946	0.220727	0.122436	0.027959	0.120039	0.052178	-0.042097	0.2
788	-0.141495	-0.130481	0.247952	0.166986	0.015830	0.134146	0.090760	-0.086861	0.2
789	-0.062910	-0.129503	0.198673	0.111234	-0.076534	0.146725	0.146133	-0.050591	0.2

780 rows × 57 columns

In [23]:

```
red_test_data=df_test[[i for i in df_test.columns[1:]]] @ vs_rd.T
red_test_data
```

Out[23]:

	0	1	2	3	4	5	6	7	
390	-0.177132	-0.175115	0.269655	0.118365	0.038968	0.085124	0.057599	-0.065892	0.1
391	-0.103300	-0.160897	0.167615	0.201968	-0.008014	0.092952	0.045541	0.002151	0.2
392	-0.153435	-0.173594	0.193888	0.138280	0.034524	0.119990	0.119908	-0.088775	0.1
393	-0.115593	-0.050115	0.164323	0.055682	0.003437	0.064277	0.133570	-0.020624	0.1
394	-0.055370	-0.026352	0.158553	0.109222	0.025264	-0.003519	0.089221	0.029834	0.1
395	-0.075354	-0.085509	0.129584	0.130524	0.118713	0.045890	0.123421	-0.066907	0.1
396	-0.133736	-0.087822	0.116717	0.095699	0.107149	0.091118	0.136185	-0.025547	0.1
397	-0.009092	-0.165390	0.148819	0.110606	0.056945	0.022829	0.080098	-0.126040	0.1
398	-0.035924	-0.125528	0.161770	0.197797	-0.043080	0.037714	0.048948	-0.143283	0.1
399	-0.044933	-0.080137	0.083556	0.119618	0.004948	0.084383	0.091632	-0.087539	0.1
790	-0.106879	-0.175464	0.149705	0.078446	-0.025678	0.124027	0.123001	-0.064445	0.2
791	-0.036366	-0.197258	0.199854	0.082745	-0.076543	0.170693	0.068024	-0.083996	0.2
792	-0.126374	-0.273559	0.068183	0.110489	0.033757	0.167480	0.143893	-0.141644	0.2
793	-0.082626	-0.098502	0.064317	0.058600	0.027592	0.065021	0.152353	-0.007970	0.1
794	-0.124669	-0.244321	0.158432	0.100248	-0.001757	0.141562	0.168321	-0.042069	0.1
795	0.034578	-0.137226	0.070334	0.087678	-0.043030	0.057677	0.029719	-0.030424	0.2
796	-0.063201	-0.225409	0.158612	0.092467	0.023689	0.127564	0.171994	-0.043018	0.2
797	0.039070	-0.085903	0.126874	0.071063	-0.034993	0.092489	0.064239	-0.060878	0.1
798	-0.084401	-0.123668	0.107070	0.127319	-0.094750	0.087222	0.028819	-0.110238	0.2
799	-0.088090	-0.212623	0.193325	0.101219	0.022406	0.138563	0.145457	-0.036179	0.1

20 rows × 57 columns

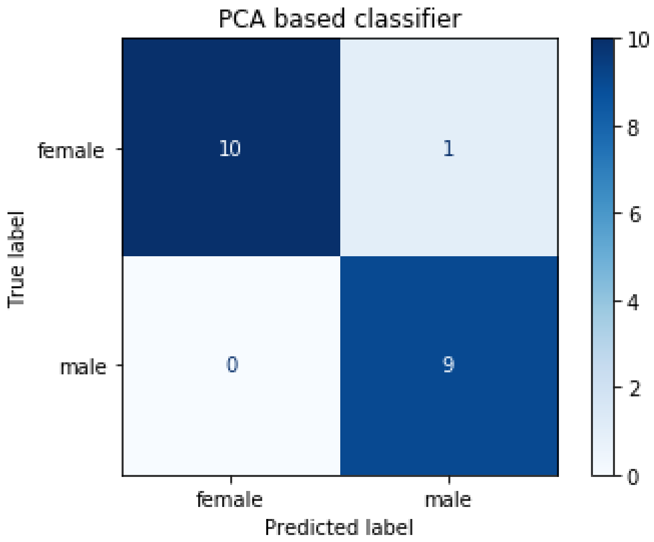
Classification

In [24]:

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

classifier = GaussianNB()
classifier.fit(np.array(red_train_data),df_train['Class'])

disp = plot_confusion_matrix(classifier,np.array(red_test_data) , df_test["Class"].to_l
ist(),
                                 display_labels=["female","male"],
                                 cmap=plt.cm.Blues,
                                 )
disp.ax_.set_title("PCA based classifier")
plt.show()
```



We can see we get good accuracy even after reducing the dimensions (almost half of original dimension)

In [ ]:

```

```