2. For the same dataset (2 classes, male and female)

a) Use LDA to reduce the dimension from d to d’. (Here d=128)

b) Choose the direction W to reduce the dimension d’ (select appropriate d’).

c) Use d’ features to classify the test cases (any classification algorithm will do, Bayes classifier, minimum distance classifier, and so on).

In [1]:

```
import pandas as pd
import numpy as np
import scipy as sp
```

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	-0.0
1	2	male	-0.030614	0.049667	0.008084	-0.050324	0.007649	-0.063818	-0.0
2	3	male	-0.096178	0.061127	0.035326	-0.035388	-0.090728	-0.018634	-0.0
3	4	male	-0.103057	0.085044	0.078333	-0.035873	-0.028163	0.004924	0.0
4	5	male	-0.125815	0.120046	0.023131	-0.042901	0.038215	-0.049677	-0.0
...	...	...	...	...	...	...	...	...	
795	796	female	-0.164731	0.064301	0.058630	-0.017420	-0.157600	-0.022536	0.0
796	797	female	-0.095308	0.051095	0.092913	-0.101745	-0.083153	-0.028159	0.0
797	798	female	-0.202852	0.037039	0.079731	-0.047156	-0.140062	-0.080246	0.0
798	799	female	-0.088300	0.063530	0.049627	-0.026011	-0.172773	0.086218	0.0
799	800	female	-0.156201	0.055165	0.142716	-0.115393	-0.128982	-0.139830	-0.0

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	-0.15
1	male	-0.030614	0.049667	0.008084	-0.050324	0.007649	-0.063818	-0.019530	-0.11
2	male	-0.096178	0.061127	0.035326	-0.035388	-0.090728	-0.018634	-0.024315	-0.13
3	male	-0.103057	0.085044	0.078333	-0.035873	-0.028163	0.004924	0.007829	-0.01
4	male	-0.125815	0.120046	0.023131	-0.042901	0.038215	-0.049677	-0.054258	-0.13
...	...	...	...	...	...	...	...	...	
795	female	-0.164731	0.064301	0.058630	-0.017420	-0.157600	-0.022536	0.002864	-0.07
796	female	-0.095308	0.051095	0.092913	-0.101745	-0.083153	-0.028159	0.009090	-0.11
797	female	-0.202852	0.037039	0.079731	-0.047156	-0.140062	-0.080246	0.057668	-0.12
798	female	-0.088300	0.063530	0.049627	-0.026011	-0.172773	0.086218	0.042710	-0.16
799	female	-0.156201	0.055165	0.142716	-0.115393	-0.128982	-0.139830	-0.037305	-0.10

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	-0.15846
1	male	-0.030614	0.049667	0.008084	-0.050324	0.007649	-0.063818	-0.019530	-0.11990
2	male	-0.096178	0.061127	0.035326	-0.035388	-0.090728	-0.018634	-0.024315	-0.13978
3	male	-0.103057	0.085044	0.078333	-0.035873	-0.028163	0.004924	0.007829	-0.01701
4	male	-0.125815	0.120046	0.023131	-0.042901	0.038215	-0.049677	-0.054258	-0.13075
...	...	...	...	...	...	...	...	...	.
795	female	-0.164731	0.064301	0.058630	-0.017420	-0.157600	-0.022536	0.002864	-0.07273
796	female	-0.095308	0.051095	0.092913	-0.101745	-0.083153	-0.028159	0.009090	-0.11451
797	female	-0.202852	0.037039	0.079731	-0.047156	-0.140062	-0.080246	0.057668	-0.12208
798	female	-0.088300	0.063530	0.049627	-0.026011	-0.172773	0.086218	0.042710	-0.16185
799	female	-0.156201	0.055165	0.142716	-0.115393	-0.128982	-0.139830	-0.037305	-0.10140

800 rows × 129 columns

In [7]:

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

In [8]:

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

In [12]:

```
df_train[0:390]
```

Out[12]:

	Class	0	1	2	3	4	5	6	
0	male	-0.066420	0.151611	0.027740	0.052771	-0.066105	-0.041232	-0.002637	-0.15846
1	male	-0.030614	0.049667	0.008084	-0.050324	0.007649	-0.063818	-0.019530	-0.11990
2	male	-0.096178	0.061127	0.035326	-0.035388	-0.090728	-0.018634	-0.024315	-0.13978
3	male	-0.103057	0.085044	0.078333	-0.035873	-0.028163	0.004924	0.007829	-0.01701
4	male	-0.125815	0.120046	0.023131	-0.042901	0.038215	-0.049677	-0.054258	-0.13075
...	...	...	...	...	...	...	...	...	.
385	male	-0.064409	0.111391	0.204528	-0.061478	-0.007144	-0.185951	0.037198	-0.12249
386	male	0.002976	0.060769	-0.016651	0.053049	-0.099903	-0.082844	0.017824	-0.10680
387	male	-0.009311	0.077162	0.031294	-0.089573	-0.038689	-0.031663	-0.012356	-0.06409
388	male	-0.036479	0.063089	0.037658	-0.000695	-0.104741	-0.018407	0.030498	-0.08932
389	male	-0.201210	0.054217	0.173419	0.004309	-0.068915	-0.114538	0.124839	-0.06773

390 rows × 129 columns

In [13]:

```
df_train[390:]
```

Out[13]:

	Class	0	1	2	3	4	5	6	
400	female	0.001747	0.185678	0.073260	0.042142	-0.088674	0.028186	-0.027830	-0.06421
401	female	-0.091598	0.095340	0.072125	-0.092276	-0.079953	0.047782	-0.004701	-0.09200
402	female	-0.018751	0.088572	0.068894	-0.065700	-0.115126	0.024339	-0.028420	-0.15932
403	female	-0.130889	0.093262	0.122244	-0.110014	-0.157625	-0.036781	0.073908	-0.09857
404	female	-0.037433	0.078158	0.118061	-0.117658	-0.194807	-0.045464	-0.014104	-0.15882
...	...	...	...	...	...	...	...	...	.
785	female	-0.017931	0.045591	0.059826	-0.059833	-0.126832	0.070365	0.026347	-0.09353
786	female	-0.039572	0.049409	0.100950	-0.092940	-0.152356	-0.057714	-0.000538	-0.11599
787	female	-0.015170	-0.001116	0.132143	-0.078673	-0.134462	-0.155683	-0.030665	-0.16475
788	female	-0.151263	0.096725	0.075206	-0.047269	-0.233372	-0.019918	-0.072370	-0.11840
789	female	-0.179743	0.104404	0.052260	-0.074608	-0.166008	0.023287	0.006970	-0.14983

390 rows × 129 columns

In [15]:

```
df_test
```

Out[15]:

	Class	0	1	2	3	4	5	6	
390	male	-0.080683	0.097440	0.006550	0.018112	-0.114999	0.160041	-0.002373	-0.05795
391	male	-0.010301	0.135185	0.049710	-0.046424	-0.041742	0.016607	-0.041778	-0.04802
392	male	-0.112450	0.080098	0.030571	0.000952	-0.097450	-0.045070	-0.003641	-0.14282
393	male	-0.087855	0.100264	0.069775	0.037204	-0.047182	-0.047233	-0.013604	-0.18937
394	male	-0.139066	0.141988	0.070456	-0.003518	-0.065637	-0.037767	-0.094195	-0.19566
395	male	-0.129449	0.132177	0.055916	-0.009390	-0.080541	-0.072362	-0.067433	-0.19224
396	male	-0.158460	0.109948	0.019088	0.015506	-0.069668	0.032311	0.015062	-0.14081
397	male	-0.101499	0.119739	0.016951	-0.013677	-0.055524	0.028399	0.028164	-0.15210
398	male	-0.149516	0.081588	0.090796	-0.053116	-0.133314	0.001096	0.019941	-0.11780
399	female	0.039844	0.070357	0.130196	-0.007683	-0.077825	-0.021298	-0.024133	-0.08510
790	female	-0.184166	0.122040	0.064143	-0.116653	-0.140633	0.041252	0.014207	-0.10881
791	female	-0.128052	0.082184	0.148978	-0.074984	-0.113145	-0.031003	0.018468	-0.13028
792	female	-0.109618	0.094507	0.043757	-0.162004	-0.191951	0.027582	-0.033507	-0.11207
793	female	-0.201133	0.068051	0.088596	-0.084936	-0.055628	-0.069027	-0.003922	-0.19139
794	female	-0.185053	0.136105	0.096279	-0.139661	-0.244312	0.015234	-0.007418	-0.07761
795	female	-0.164731	0.064301	0.058630	-0.017420	-0.157600	-0.022536	0.002864	-0.07273
796	female	-0.095308	0.051095	0.092913	-0.101745	-0.083153	-0.028159	0.009090	-0.11451
797	female	-0.202852	0.037039	0.079731	-0.047156	-0.140062	-0.080246	0.057668	-0.12208
798	female	-0.088300	0.063530	0.049627	-0.026011	-0.172773	0.086218	0.042710	-0.16185
799	female	-0.156201	0.055165	0.142716	-0.115393	-0.128982	-0.139830	-0.037305	-0.10140

20 rows × 129 columns

Computing mean and scatter matrix

In [16]:

```python
df_male=df_train[[i for i in df.columns[1:]]][:390]
df_male
```

Out[16]:

	0	1	2	3	4	5	6	7	
0	-0.066420	0.151611	0.027740	0.052771	-0.066105	-0.041232	-0.002637	-0.158467	0.13
1	-0.030614	0.049667	0.008084	-0.050324	0.007649	-0.063818	-0.019530	-0.119905	0.18
2	-0.096178	0.061127	0.035326	-0.035388	-0.090728	-0.018634	-0.024315	-0.139786	0.05
3	-0.103057	0.085044	0.078333	-0.035873	-0.028163	0.004924	0.007829	-0.017016	0.11
4	-0.125815	0.120046	0.023131	-0.042901	0.038215	-0.049677	-0.054258	-0.130758	0.17
...	...	...	...	...	...	...	...	...	
385	-0.064409	0.111391	0.204528	-0.061478	-0.007144	-0.185951	0.037198	-0.122494	0.16
386	0.002976	0.060769	-0.016651	0.053049	-0.099903	-0.082844	0.017824	-0.106804	0.12
387	-0.009311	0.077162	0.031294	-0.089573	-0.038689	-0.031663	-0.012356	-0.064096	0.17
388	-0.036479	0.063089	0.037658	-0.000695	-0.104741	-0.018407	0.030498	-0.089321	0.13
389	-0.201210	0.054217	0.173419	0.004309	-0.068915	-0.114538	0.124839	-0.067737	0.10

390 rows × 128 columns

In [17]:

```python
df_female=df_train[[i for i in df.columns[1:]]][390:]
df_female
```

Out[17]:

	0	1	2	3	4	5	6	7	
400	0.001747	0.185678	0.073260	0.042142	-0.088674	0.028186	-0.027830	-0.064211	0.097
401	-0.091598	0.095340	0.072125	-0.092276	-0.079953	0.047782	-0.004701	-0.092005	0.222
402	-0.018751	0.088572	0.068894	-0.065700	-0.115126	0.024339	-0.028420	-0.159320	0.164
403	-0.130889	0.093262	0.122244	-0.110014	-0.157625	-0.036781	0.073908	-0.098571	0.12
404	-0.037433	0.078158	0.118061	-0.117658	-0.194807	-0.045464	-0.014104	-0.158824	0.15
...	...	...	...	...	...	...	...	...	
785	-0.017931	0.045591	0.059826	-0.059833	-0.126832	0.070365	0.026347	-0.093537	0.27
786	-0.039572	0.049409	0.100950	-0.092940	-0.152356	-0.057714	-0.000538	-0.115991	0.206
787	-0.015170	-0.001116	0.132143	-0.078673	-0.134462	-0.155683	-0.030665	-0.164758	0.105
788	-0.151263	0.096725	0.075206	-0.047269	-0.233372	-0.019918	-0.072370	-0.118408	0.19
789	-0.179743	0.104404	0.052260	-0.074608	-0.166008	0.023287	0.006970	-0.149833	0.15

390 rows × 128 columns

In [18]:

```
mean_male=df_male.mean()
mean_male
```

Out[18]:

```
0     -0.091137
1      0.092309
2      0.043384
3     -0.030786
4     -0.093000
         ...
123   -0.118805
124    0.021776
125   -0.025222
126    0.020244
127    0.040089
Length: 128, dtype: float64
```

In [19]:

```
mean_female=df_female.mean()
mean_female
```

Out[19]:

```
0     -0.116036
1      0.076929
2      0.081975
3     -0.077089
4     -0.122499
         ...
123   -0.086894
124    0.053799
125   -0.031701
126    0.004233
127    0.025536
Length: 128, dtype: float64
```

Sw matrix (within class)

In [20]:

```
#Scatter matrix 1

S1= (df_male.values - np.array(mean_male) ).T @ (df_male.values - np.array(mean_male))
```

In [21]:

```
S1
```

Out[21]:

```
array([[ 1.21590589, -0.22385172, -0.12872454, ...,  0.24384893,
        -0.07807342, -0.03553427],
       [-0.22385172,  0.89023903,  0.14525314, ..., -0.14402007,
        -0.11400393, -0.08026288],
       [-0.12872454,  0.14525314,  0.98592786, ..., -0.1457067 ,
         0.0365962 , -0.02872259],
       ...,
       [ 0.24384893, -0.14402007, -0.1457067 , ...,  0.951065  ,
         0.02352049,  0.00168977],
       [-0.07807342, -0.11400393,  0.0365962 , ...,  0.02352049,
         0.97078087,  0.10619016],
       [-0.03553427, -0.08026288, -0.02872259, ...,  0.00168977,
         0.10619016,  0.68554017]])
```

In [22]:

```
S2 = (df_female.values - np.array(mean_female)).T @ (df_female.values - np.array(mean_femal
```

In [23]:

```
S2
```

Out[23]:

```
array([[ 0.9621001 , -0.13917361,  0.14762725, ...,  0.06294964,
        -0.08545036, -0.05254243],
       [-0.13917361,  1.01438579,  0.01902008, ..., -0.17504062,
        -0.04697787,  0.01789532],
       [ 0.14762725,  0.01902008,  0.75556753, ..., -0.05615372,
        -0.07224368,  0.15315488],
       ...,
       [ 0.06294964, -0.17504062, -0.05615372, ...,  0.88627125,
        -0.04231172,  0.11945256],
       [-0.08545036, -0.04697787, -0.07224368, ..., -0.04231172,
         0.92043746, -0.1086309 ],
       [-0.05254243,  0.01789532,  0.15315488, ...,  0.11945256,
        -0.1086309 ,  1.0380166 ]])
```

In [24]:

```
Sw=S1+S2
```

For Sb (between class) we need mean of whole data

In [25]:

```
mean_whole=np.mean(df_train)
```

In [26]:

```
def Sb_i(mean_whole,mean_class,number_of_points,dim):
    return (number_of_points * np.subtract(mean_class,mean_whole).values.reshape(dim,1) @ r
```

In [27]:

```
Sb1=Sb_i(mean_whole,mean_male,390,128)
```

In [28]:

```
Sb2=Sb_i(mean_whole,mean_female,390,128)
```

In [29]:

```
Sb=Sb1+Sb2
```

In [30]:

```
eigen_vectors, eigen_values,_=np.linalg.svd(np.linalg.inv(Sw) @ Sb)
```

In [31]:

```
eigen_values
```

Out[31]:

```
array([2.72333430e+06, 4.14087275e-04, 3.92718028e-04, 3.47948322e-04,
       3.21549303e-04, 3.08722475e-04, 2.99873459e-04, 2.75389943e-04,
       2.72325531e-04, 2.63408566e-04, 2.50202781e-04, 2.41079159e-04,
       2.33192792e-04, 2.29902802e-04, 2.20424901e-04, 2.12390727e-04,
       2.08723331e-04, 2.02983943e-04, 1.98274952e-04, 1.92096411e-04,
       1.87930711e-04, 1.83259321e-04, 1.79973511e-04, 1.68708968e-04,
       1.64019504e-04, 1.57551063e-04, 1.55982351e-04, 1.52701669e-04,
       1.49555675e-04, 1.44957772e-04, 1.38982285e-04, 1.36554861e-04,
       1.35132546e-04, 1.33338857e-04, 1.28368942e-04, 1.24689578e-04,
       1.19399238e-04, 1.17035283e-04, 1.15960905e-04, 1.11093987e-04,
       1.08886529e-04, 1.04481965e-04, 1.01524049e-04, 1.00881439e-04,
       9.74491377e-05, 9.51172349e-05, 9.35910544e-05, 8.99262536e-05,
       8.88186053e-05, 8.76201235e-05, 7.99192625e-05, 7.83831228e-05,
       7.68043061e-05, 7.57674485e-05, 7.44863358e-05, 7.08509161e-05,
       6.98792249e-05, 6.81283604e-05, 6.65086484e-05, 6.41987831e-05,
       6.04101777e-05, 5.99595235e-05, 5.88472506e-05, 5.64822859e-05,
       5.57394138e-05, 5.39642613e-05, 5.13365336e-05, 5.08517143e-05,
       4.84559338e-05, 4.75352581e-05, 4.73064850e-05, 4.57833601e-05,
       4.35205221e-05, 4.14972441e-05, 4.12446418e-05, 3.94001245e-05,
       3.75535651e-05, 3.70542609e-05, 3.46971925e-05, 3.44804253e-05,
       3.31875332e-05, 3.15001859e-05, 3.06431124e-05, 2.94046780e-05,
       2.80340966e-05, 2.78905436e-05, 2.73364007e-05, 2.41553891e-05,
       2.37390224e-05, 2.28896573e-05, 2.25135106e-05, 2.07032761e-05,
       2.00756602e-05, 1.83019631e-05, 1.74186818e-05, 1.67906905e-05,
       1.61982034e-05, 1.50486336e-05, 1.44613895e-05, 1.40479915e-05,
       1.36479073e-05, 1.17312851e-05, 1.12881342e-05, 1.07155906e-05,
       1.01684656e-05, 9.03868353e-06, 8.60895383e-06, 7.69794773e-06,
       7.07619203e-06, 6.35861567e-06, 5.65950430e-06, 5.08306307e-06,
       4.42827147e-06, 3.77289525e-06, 3.67781457e-06, 3.41245691e-06,
       2.91492001e-06, 2.18146763e-06, 1.79744983e-06, 1.45753050e-06,
       1.14460294e-06, 1.02814399e-06, 7.10385451e-07, 6.55616874e-07,
       4.17230863e-07, 2.33276401e-07, 1.30299101e-07, 9.41889153e-08])
```

In [32]:

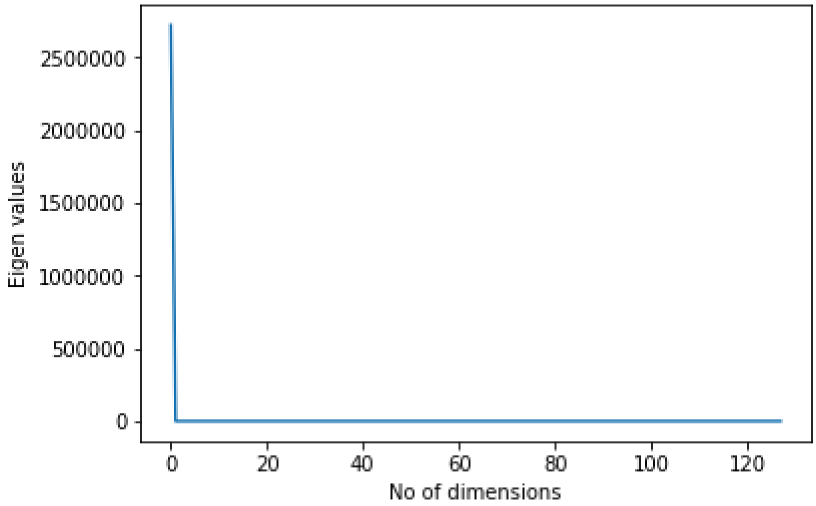
```
import matplotlib.pyplot as plt
```

In [33]:

```
plt.plot(eigen_values)
plt.xlabel("No of dimensions")
plt.ylabel("Eigen values")
```

Out[33]:

```
Text(0, 0.5, 'Eigen values')
```



We take only first eigen value (2 classes --> take (2-1) eigen value).

Reducing data

In [34]:

```
red_train_data=df_train[df.columns[1:]] @ eigen_vectors[0]
red_train_data
```

Out[34]:

```
0     -0.076645
1      0.011543
2      0.075068
3      0.100944
4     -0.005650
         ...
785   -0.060044
786    0.072923
787    0.010113
788    0.093721
789    0.084124
Length: 780, dtype: float64
```

In [35]:

```
red_test_data=df_test[[i for i in df_test.columns[1:]]] @ eigen_vectors[0]
red_test_data
```

Out[35]:

```
390    0.093422
391   -0.005016
392    0.149489
393    0.084264
394    0.063795
395    0.052293
396    0.160087
397    0.017396
398    0.045768
399    0.040492
790    0.118789
791    0.041542
792    0.105642
793    0.068019
794    0.089727
795    0.098056
796    0.043459
797    0.053029
798    0.056282
799    0.033971
dtype: float64
```

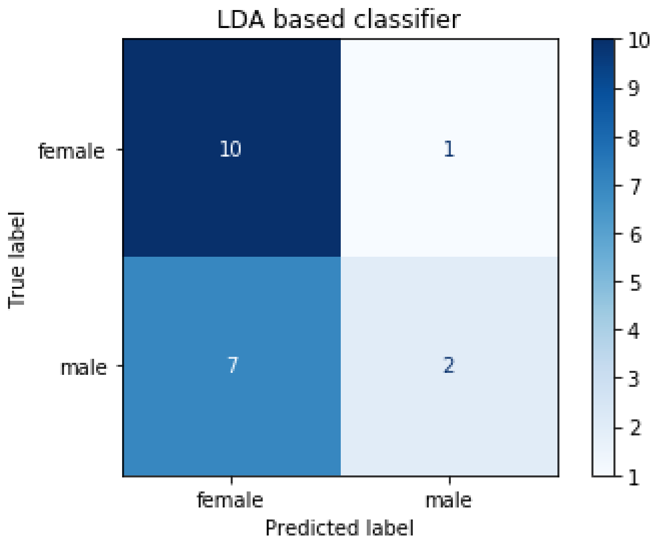
Classification

In [36]:

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

classifier = GaussianNB()
classifier.fit(np.array(red_train_data).reshape(-1,1),df_train['Class'])

disp = plot_confusion_matrix(classifier,np.array(red_test_data).reshape(-1,1) , df_test["Cl
                                 display_labels=["female","male"],
                                 cmap=plt.cm.Blues,
                                 )
disp.ax_.set_title("LDA based classifier")
plt.show()
```



3 Give the comparative study for the above results w.r.t to classification accuracy in terms of the confusion matrix.

Naive Bayes classifier gives 60 percent accuracy for LDA.Whereas for PCA we get 95 percent accuracy. We can see that PCA outperforms LDA in classification.

From the confusion matrix we see PCA based classifier only misclassifies one sample. LDA based classifier misclassifies 8 samples out of 20