Task 3

Apply LDA algorithm to select the appropriate data from the given data set. Use XG boost algorithm for classification.

Tools: Rapid Miner, Python, Scikitlearn, Anaconda navigator

Algorithm

1. Compute the within class and between class scatter matrices
2. Compute the eigenvectors and corresponding eigenvalues for the scatter matrices
3. Sort the eigenvalues and select the top k
4. Create a new matrix containing eigenvectors that map to the k eigenvalues
5. Obtain the new features (i.e. LDA components) by taking the dot product of the data and the matrix from step 4

Code

```
# In[ ]:
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from numpy import linalg as lg
get_ipython().magic(u'matplotlib inline')
df =
pd.read_csv('C:/Users/vyoms/Desktop/SCLC_study_output_filtered_2.csv',header
=None)
df1 = df.drop(df.index[0])
df2 = df1.drop(df.columns[0], axis=1)
```

```
df3 = df2
df3_1 = df2.values[0:20,:]
df3_2 = df2.values[20:, : ]
m_1 = df3_1.mean(axis = 0)
m_2 = df3_2.mean(axis = 0)
mean_all = df2.mean(axis = 0)
mean_1 = m_1.reshape(1,19)
mean_1 = np.repeat(mean_1,20,axis = 0)
mean_2 = m_2.reshape(1,19)
mean_2 = np.repeat(mean_2,20,axis = 0)
within_class_scatter = np.zeros((19,19))
wcs_1 = np.zeros((19,19))
wcs_1 = np.matmul((np.transpose(df3_1 - mean_1 )), (df3_1 - mean_1))
wcs_2 = np.zeros((19,19))
wcs_2 = np.matmul((np.transpose(df3_2 - mean_2 )), (df3_2 - mean_2))
within_class_scatter = np.add(wcs_1,wcs_2)
bcs_1 = np.multiply(len(df3_1),np.outer((m_1 - mean_all),(m_1 - mean_all)))
bcs_2 = np.multiply(len(df3_2),np.outer((m_2 - mean_all),(m_2 - mean_all)))
between_class_scatter = np.add(bcs_1,bcs_2)
e_val, e_vector =
np.linalg.eig(np.dot(lg.inv(within_class_scatter),between_class_scatter))
for e in range (len(e_val)):
e_scatter = e_vector[:,e].reshape(19,1)
print(e_val[e].real)
print(between_class_scatter)
eig_pairs = [(np.abs(e_val[i]).real, e_vector[:,i].real) for i in range(len(e_val))]
eig_pairs = sorted(eig_pairs, key=lambda k: k[0], reverse=True)
```

```
print('Eigenvalues in decreasing order:\n')
for i in eig_pairs:
print(i[0])
W= eig_pairs[0][1].reshape(19,1)
W
lda_project = np.dot(df2,W)
lda_project
# In[177]:
#plot
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.set_title('LDA')
ax.plot(lda_project[0:20], np.zeros(20), linestyle='None', marker='o', color='blue',
label='NSCLC')
ax.plot(lda_project[20:40], np.zeros(20), linestyle='None', marker='o', color='red',
label='SCLC')
fig.show()
# In[185]:
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
y1_ = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
# LDA
sklearn_lda = LDA(n_components=1)
X_lda_sklearn = sklearn_lda.fit_transform(df2, y1_)
X_lda_sklearn= -X_lda_sklearn
print(X_lda_sklearn)
#plot
```

```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.set_title('LDA')
ax.plot(X_lda_sklearn[0:20], np.zeros(20), linestyle='None', marker='o',
color='blue', label='NSCLC')
ax.plot(X_lda_sklearn[20:40], np.zeros(20), linestyle='None', marker='o',
color='red', label='SCLC')
fig.show()
```

Output

```
[ 3.15378176]
[ 4.86686796]
[ 2.81120157]
[ 3.93543558]
[ 3.39771836]
[ 3.25628819]
[ 3.21596383]
[-4.49065733]
[-4.6115194 ]
[-3.45215203]
[-2.75643608]
[-3.83408221]
[-3.54607243]
[-3.25235288]
[-3.40306303]
```

[-4.04197759]

[-4.35872643]

[-5.76347876]

[-5.589837]

[-5.74304609]

[-3.85936631]

[-3.13967526]

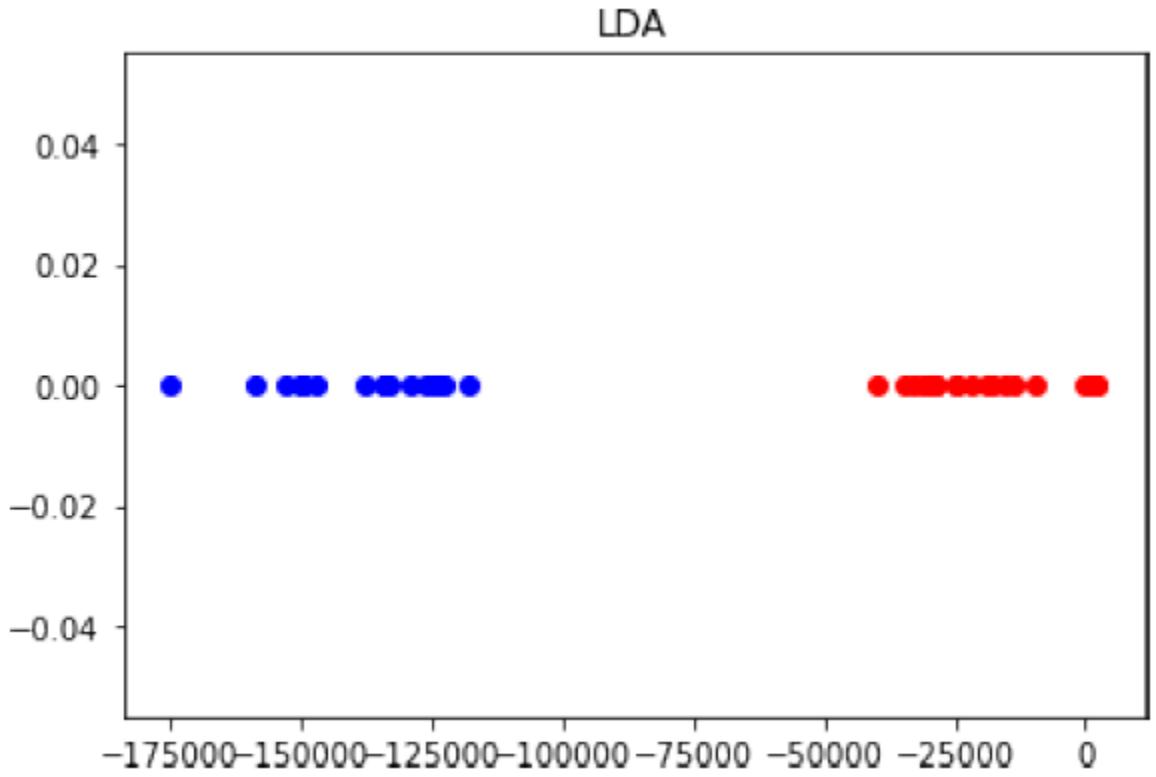
[-3.56371641]

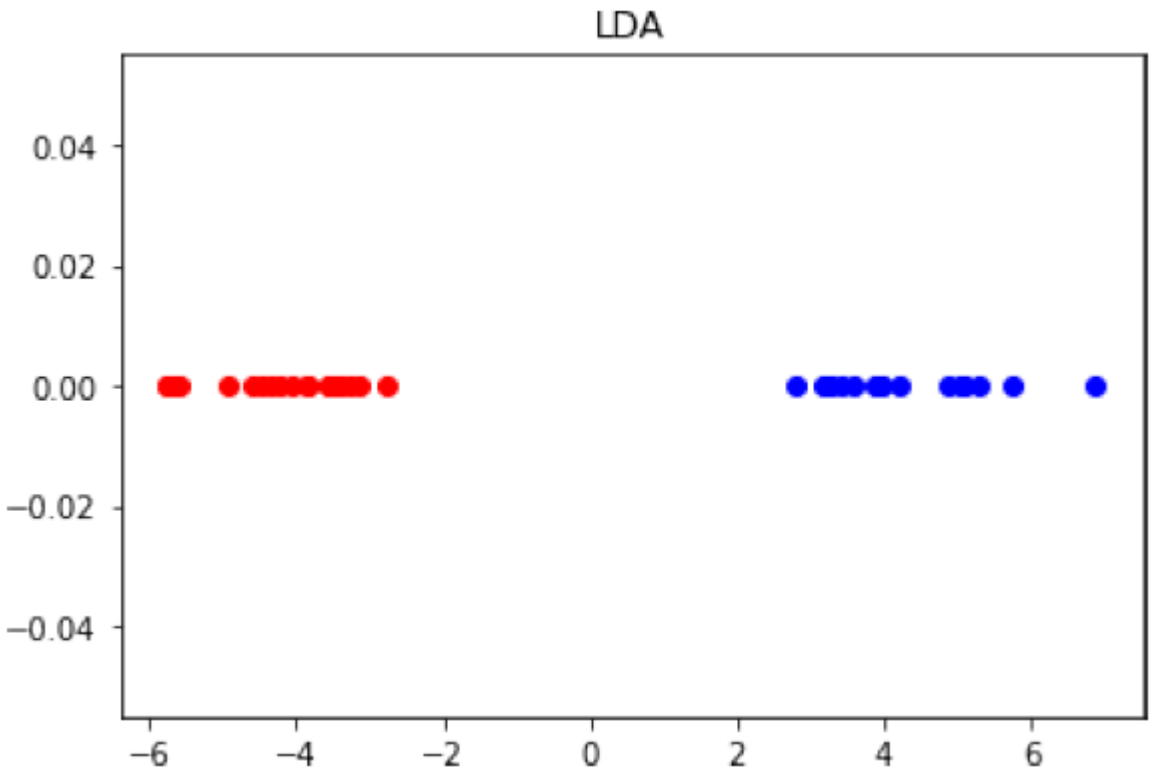
[-5.68605839]

[-4.9237813]

[-3.53641015]

[-4.242908]]




LDA
0.04
0.02
0.00
−0.02
−0.04
−6
−4
−2
0
2
4
6