Comparison of LDA and QDA Classifiers in Python Using Suicide Rate Data

Burak Koryan | burak@koryan.ca | http://koryan.ca | Feb 9 2019

Objective:

The goal of this project is to compare and contrast Linear Discriminant Analysis(LDA) and Quadratic Discriminant Analysis(QDA) in Python using suicide rate data found on kaggle.com[3].

Introduction:

Linear discriminant analysis(LDA)[1] and quadratic discriminant analysis(QDA)[2] are widely used classifiers in machine learning and in the *sklearn* library to classify data.In this project,randomly chosen dataset on suicide data from multiple countries all over the world.The database has information on *year,sex,age,population,suicide rate per 100,000 people* etc.However,in order to make classification easier,only *year,population,and sex* data has been used on suicide rate.First 20 entry of the dataset used can be seen in Figure 1 below.

To investigate differences between both LDA and QDA,test size and random state has been changed of the classifiers.In the results section,there will be plots and tables shown to explain differences between them.As a python compiler Anaconda Navigator's *Spyder* has been used to classify and plot results.

country	year	sex	age	suicides_no	population	suicides/100k	pop
Albania	1987	male	15-24	years	21	312900	6.71
Albania	1987	male	35-54	years	16	308000	5.19
Albania	1987	female	15-24	years	14	289700	4.83
Albania	1987	male	75+	years	1	21800	4.59
Albania	1987	male	25-34	years	9	274300	3.28
Albania	1987	female	75+	years	1	35600	2.81
Albania	1987	female	35-54	years	6	278800	2.15
Albania	1987	female	25-34	years	4	257200	1.56
Albania	1987	male	55-74	years	1	137500	0.73
Albania	1987	female	5-14	years	0	311000	0
Albania	1987	female	55-74	years	0	144600	0
Albania	1987	male	5-14	years	0	338200	0
Albania	1988	female	75+	years	2	36400	5.49
Albania	1988	male	15-24	years	17	319200	5.33
Albania	1988	male	75+	years	1	22300	4.48
Albania	1988	male	35-54	years	14	314100	4.46
Albania	1988	male	55-74	years	4	140200	2.85
Albania	1988	female	15-24	years	8	295600	2.71
Albania	1988	female	55-74	years	3	147500	2.03

Figure 1 : Dataset screenshot of the first 20 entry

Procedure:

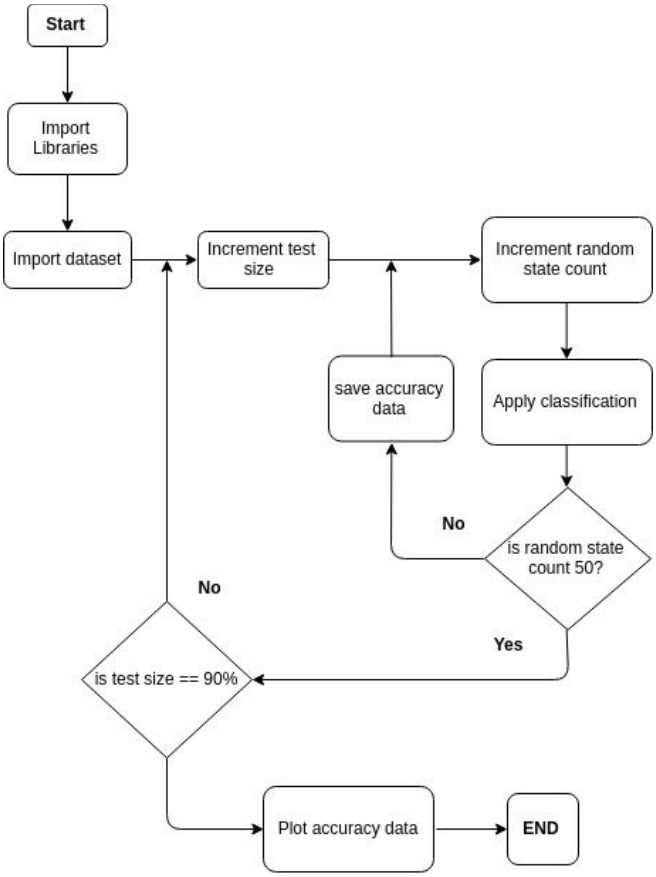

Figure 2 : Classification algorithm

Like any other classification system,first the necessary libraries (panda,matplot,and sklearn/lda,sklearn/qda) are imported.Then the dataset included into the compiler *spyder*.After importing the dataset,its separated into X and Y variables.The X variable has the data on *year* and *population*.The Y variable has the data on *sex*.

Two variables are created to use for changing test size(*testSizeStr)* and random state.The test size(*testSizeStr*) is incremented from 10% to 90% while random state variable is incremented from 0 to 50.Through this incrementing variables process, lda or qda classifies data and the classification accuracy is saved into another variable *dataArray.*Collected data is plotted against test size.The classification algorithm,that summarizes the procedure,can be viewed in Figure 2.

Results:

After LDA and QDA classifications done on the chosen dataset,the accuracy data is plotted to show graphically and two tables are created to show the accuracy data numerically.

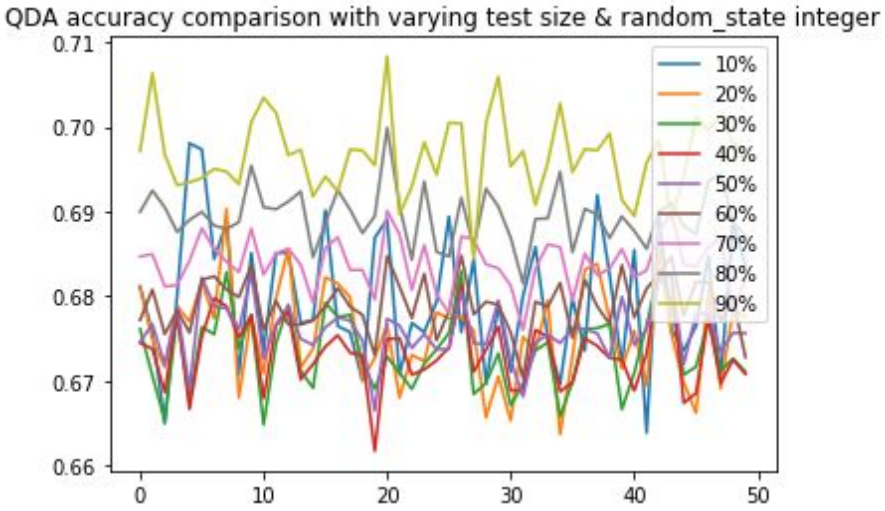

Figure 3: QDA accuracy plot with varying test size and random_state integer

In Figure 3,accuracy output plot of QDA can be seen when the test size is increased from 10% to 90% and the random state integer from 0 to 50.The highest accuracy is obtained when the test size is 90%,when the random state integer is 20, throughout varying random state from 0 to 50 (yellow trace) and the second highest accuracy rate is obtained when the test size is 80%,when the random state integer is at 20 as well.

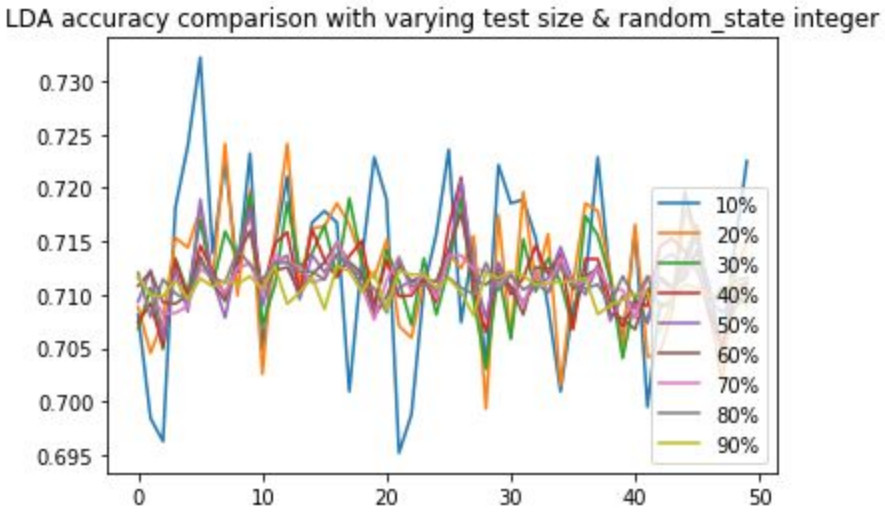

Figure 4: LDA accuracy plot with varying test size and random_state integer

Figure 4 shows the accuracy output plot of the LDA classifier.The LDA gives the highest accuracy rate when the test size is 10% and the random state integer is at 9.

Test Size/Random State	0	1	2	3	4	42
10%	67.57	66.49	68.00	69.80	69.73	68.94
20%	67.48	67.16	67.88	67.70	68.17	68.56
30%	67.06	66.53	67.78	66.77	67.63	68.30
40%	67.37	66.86	67.81	66.66	67.52	68.25
50%	67.67	67.19	67.87	66.90	68.22	68.16
90%	71.16	70.99	70.98	71.13	70.95	69.84

Table 1 : Sample QDA accuracy table

Table 1 shows some QDA accuracies when the test size and random state integer increase.It can be said that there is no direct relationship between the random state integer and the classifier's accuracy.But,we can say that the classification accuracy increases with test size increase.The highest accuracy is obtained from QDA when the test size is 90%.

Test Size/Random State	0	1	2	3	4	9
10%	70.81	69.84	69.62	71.81	72.39	72.32
20%	70.88	70.45	70.75	71.53	71.44	71.98
30%	70.68	71.06	70.48	71.33	71.01	71.93
40%	70.74	70.92	70.51	71.31	70.96	71.81
50%	70.93	71.22	70.58	71.23	70.84	71.79
90%	71.16	70.99	70.98	71.13	70.95	71,17

Table 2 : Sample LDA accuracy table

Similarly,Table 2 shows similar results but with less accuracy difference in classification results when the test size and random state variable have changed.Before comparing the overall mean accuracies of both classifiers,it would be wrong to make a conclusion but from Table 2,one observation can be made : LDA is a better classifier to use for the chosen dataset if higher accuracy is needed as classification result.

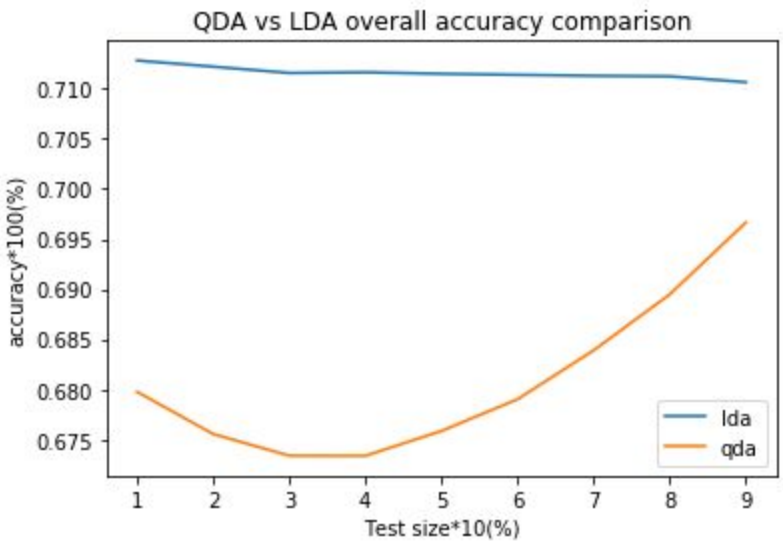

Figure 5 : Overall accuracy comparison of QDA and LDA

From the accuracy tables (Table 1 and Table 2) an observation about which classifier is better for the chosen dataset was made.In Figure 5,the mean overall accuracy of both classifiers is shown when the test size is changed from 10% to 90% with 10% increments.LDA gives almost a *linear* accuracy result whereas QDA gives a non-linear one.Toward 90% test size,the QDA gives its highest accuracy;however,it is still not near what LDA gives as almost always more than 71% accuracy.The highest accuracy that can be obtained from QDA is about 69.5%.

Conclusion:

Accuracy result of a classifier depends on the choice of parameters it is set and the dataset chosen to be classified.A suicide rate dataset is used for this classification project with LDA and QDA and as a result,LDA gives the highest accuracy, ~ 71% whereas QDA is not able to give anything higher than 69.5%.One possible reason why LDA gives higher accuracy is that the dataset used for classification might have *too linear* data to be classified.

References:

[1] Sklearn - QDA :
https://scikit-learn.org/0.15/modules/generated/sklearn.qda.QDA.html

[2] Sklearn - LDA:
https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

[3] Suicide rate Data on Kaggle.com : https://www.kaggle.com/szamil/who-suicide-statistics

Appendix A : Python Code for the project

```
# Burak Koryan | Feb 9 2019 | burak@koryan.ca
# Description : LDA classification of suicide rate data
# The dataset used for this project was found on kaggle.com and
# as a machine learning tutorial,a udemy.com course on machine learning
# using Python has been studied.
#############################################################################

# Importing the libraries
import numpy as np
import pandas as pd
import statistics
import random
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report,
precision_score

dataset = pd.read_csv('master.csv')
X = dataset.iloc[:, [1,6]].values
Y = dataset.iloc[:, 2].values

testSizeStr = [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
dataArray = np.zeros((10,50))
dataIndex = np.zeros(50)

for i in range(1,10):
    for j in range(0,50):
        random.seed(i)
        rndint = random.randint(1,50)
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =
testSizeStr[i], random_state = j)
        dataIndex[j] = j;

        sc = StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)

        lda = LDA(n_components = i)
        X_train = lda.fit_transform(X_train,y_train)
```

```
        X_test = lda.transform(X_test)

        classifier = LogisticRegression(random_state = j)
        classifier.fit(X_train,y_train)

        y_pred = classifier.predict(X_test)

        cm = confusion_matrix(y_test,y_pred)
        dataArray[i,j] = accuracy_score(y_test,y_pred)
    print('\n')

labelString = ["0%","10%","20%","30%","40%","50%","60%","70%","80%","90%"]
for k in range(1,10):
     plt.plot(dataIndex,dataArray[k,:],label = labelString[k])
plt.title('LDA accuracy comparison with varying test size & random_state integer')
plt.legend(loc='lower right')
plt.show()

d = np.zeros(10)
for m in range(1,10):
        d[m]= statistics.mean(dataArray[m,:])

num = [1,2,3,4,5,6,7,8,9]
plt.plot(num,d[1:],label='lda')
plt.plot(num,t[:9],label='qda')
plt.title('QDA vs LDA overall accuracy comparison')
plt.xlabel('Test size*10(%)')
plt.ylabel('accuracy*100(%)')
plt.legend()
plt.show()

#############################################################################
# Burak Koryan | Feb 9 2019 | burak@koryan.ca
# Description : QDA classification of suicide rate data
# The dataset used for this project was found on kaggle.com and
# as a machine learning tutorial,a udemy.com course on machine learning
# using Python has been studied.
#############################################################################

import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
```

```
from sklearn.metrics import confusion_matrix, classification_report,
precision_score

dataset = pd.read_csv('master.csv')
X = dataset.iloc[:, [1,6]].values
Y = dataset.iloc[:, 2].values

testSizeStr = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
dataArray = np.zeros((10,50))
dataIndex = np.zeros(50)

for i in range(0,9):
    for j in range(0,50):
        random.seed(i)
        rndint = random.randint(1,50)
        dataIndex[j] = j;
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =
testSizeStr[i], random_state = j)

        # Feature Scaling
        sc = StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)

        clf = QDA(reg_param = testSizeStr[i])
        clf_fit = clf.fit(X_train, y_train)

        pred2=clf_fit.predict(X_test)
        print(np.round(accuracy_score(y_test,pred2)*100))
        dataArray[i,j] = accuracy_score(y_test,pred2)
    print('\n')

labelString = ["10%","20%","30%","40%","50%","60%","70%","80%","90%"]
for k in range(0,9):
        plt.plot(dataIndex,dataArray[k,:],label = labelString[k])
plt.title('QDA accuracy comparison with varying test size & random_state integer')
plt.legend(loc='upper right')
plt.show()

import statistics
t = np.zeros(10)

for m in range(0,9):
    t[m]= statistics.mean(dataArray[m,:])
```