HW 3

October 17, 2022

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```python
col_names = ["status", "duration", "credit_history", "purpose", "amount",␣
↪"savings", "employment_duration", "installment_rate", "personal_status_sex",
             "other_debtors", "present_residence", "property", "age",␣
↪"other_installment_plans", "housing", "number_credits", "job",␣
↪"people_liable", "telephone", "foreign_worker", "credit_risk"]
df = pd.read_table("SouthGermanCredit.asc", delimiter = " ", names = col_names,␣
↪header = 0)
```

[3]:
```python
df.head()
```

[3]:
```
   status  duration  credit_history  purpose  amount  savings  \
0       1        18               4        2    1049        1
1       1         9               4        0    2799        1
2       2        12               2        9     841        2
3       1        12               4        0    2122        1
4       1        12               4        0    2171        1

   employment_duration  installment_rate  personal_status_sex  other_debtors  \
0                    2                 4                    2              1
1                    3                 2                    3              1
2                    4                 2                    2              1
3                    3                 3                    3              1
4                    3                 4                    3              1

   …  property  age  other_installment_plans  housing  number_credits  job  \
0  …         2   21                        3        1               1    3
1  …         1   36                        3        1               2    3
2  …         1   23                        3        1               1    2
3  …         1   39                        3        1               2    2
4  …         2   38                        1        2               2    2

   people_liable  telephone  foreign_worker  credit_risk
0              2          1               2            1
```

```
1          1        1            2          1
2          2        1            2          1
3          1        1            1          1
4          2        1            1          1

[5 rows x 21 columns]
```

[4]:
```python
from sklearn.model_selection import train_test_split
```

[5]:
```python
X = df.iloc[:,:20]
y = df.iloc[:,20:]
```

[6]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
```

[7]:
```python
import statsmodels.api as sm
```

[8]:
```python
log_reg = sm.Logit(y_train, X_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.478361
         Iterations 6
```

[9]:
```python
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            credit_risk   No. Observations:                  700
Model:                          Logit   Df Residuals:                      680
Method:                           MLE   Df Model:                           19
Date:                Mon, 17 Oct 2022   Pseudo R-squ.:                  0.2138
Time:                        19:17:54   Log-Likelihood:                -334.85
converged:                       True   LL-Null:                       -425.90
Covariance Type:            nonrobust   LLR p-value:                 1.200e-28
================================================================================
===========
                              coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
status                      0.5566      0.084      6.640      0.000       0.392
0.721
duration                   -0.0283      0.010     -2.749      0.006      -0.049
-0.008
credit_history              0.3980      0.104      3.836      0.000       0.195
0.601
purpose                     0.0320      0.036      0.887      0.375      -0.039
0.103
```

```
amount                     -0.0001   4.77e-05     -2.208      0.027      -0.000
-1.18e-05
savings                     0.2659      0.071      3.742      0.000       0.127
0.405
employment_duration         0.1039      0.084      1.232      0.218      -0.061
0.269
installment_rate           -0.3275      0.100     -3.291      0.001      -0.523
-0.132
personal_status_sex         0.2207      0.134      1.650      0.099      -0.041
0.483
other_debtors               0.2460      0.206      1.194      0.233      -0.158
0.650
present_residence          -0.0014      0.092     -0.015      0.988      -0.182
0.179
property                   -0.1402      0.109     -1.288      0.198      -0.354
0.073
age                         0.0115      0.010      1.170      0.242      -0.008
0.031
other_installment_plans     0.1033      0.136      0.759      0.448      -0.164
0.370
housing                     0.1680      0.202      0.830      0.407      -0.229
0.565
number_credits             -0.1324      0.191     -0.695      0.487      -0.506
0.241
job                         0.1070      0.163      0.656      0.512      -0.213
0.427
people_liable               0.0568      0.270      0.210      0.833      -0.473
0.586
telephone                   0.3839      0.224      1.717      0.086      -0.054
0.822
foreign_worker             -1.3645      0.469     -2.907      0.004      -2.284
-0.445
================================================================================
===========
```

Features Selected Using Combination of Statistical Significance and Impact. (All significant coefs at 0.1 alpha level were selected)

[10]:
```python
X_train_reduced = X_train[["status", "duration", "credit_history", "amount",␣
↪"savings", "installment_rate", "foreign_worker", "personal_status_sex",␣
↪"telephone"]]
X_test_reduced = X_test[["status", "duration", "credit_history", "amount",␣
↪"savings", "installment_rate", "foreign_worker", "personal_status_sex",␣
↪"telephone"]]
```

[11]:
```python
log_reg_reduced = sm.Logit(y_train, X_train_reduced).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.486058
```

```
         Iterations 6
```

[12]:
```python
print(log_reg_reduced.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            credit_risk   No. Observations:                  700
Model:                          Logit   Df Residuals:                      691
Method:                           MLE   Df Model:                            8
Date:                Mon, 17 Oct 2022   Pseudo R-squ.:                  0.2011
Time:                        19:17:55   Log-Likelihood:                -340.24
converged:                       True   LL-Null:                       -425.90
Covariance Type:            nonrobust   LLR p-value:                 6.846e-33
================================================================================
=======
                         coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
status                 0.5662      0.082      6.922      0.000       0.406
0.726
duration              -0.0280      0.010     -2.793      0.005      -0.048
-0.008
credit_history         0.4078      0.091      4.485      0.000       0.230
0.586
amount                -0.0001    4.7e-05     -2.313      0.021      -0.000
-1.66e-05
savings                0.2565      0.069      3.711      0.000       0.121
0.392
installment_rate      -0.2833      0.096     -2.947      0.003      -0.472
-0.095
foreign_worker        -0.7495      0.270     -2.780      0.005      -1.278
-0.221
personal_status_sex    0.2612      0.129      2.024      0.043       0.008
0.514
telephone              0.4695      0.203      2.312      0.021       0.072
0.867
================================================================================
=======
```

[13]:
```python
y1 = log_reg.predict(X_train)
y2 = log_reg.predict(X_test)
y3 = log_reg_reduced.predict(X_train_reduced)
y4 = log_reg_reduced.predict(X_test_reduced)
```

[14]:
```python
from sklearn import metrics
```

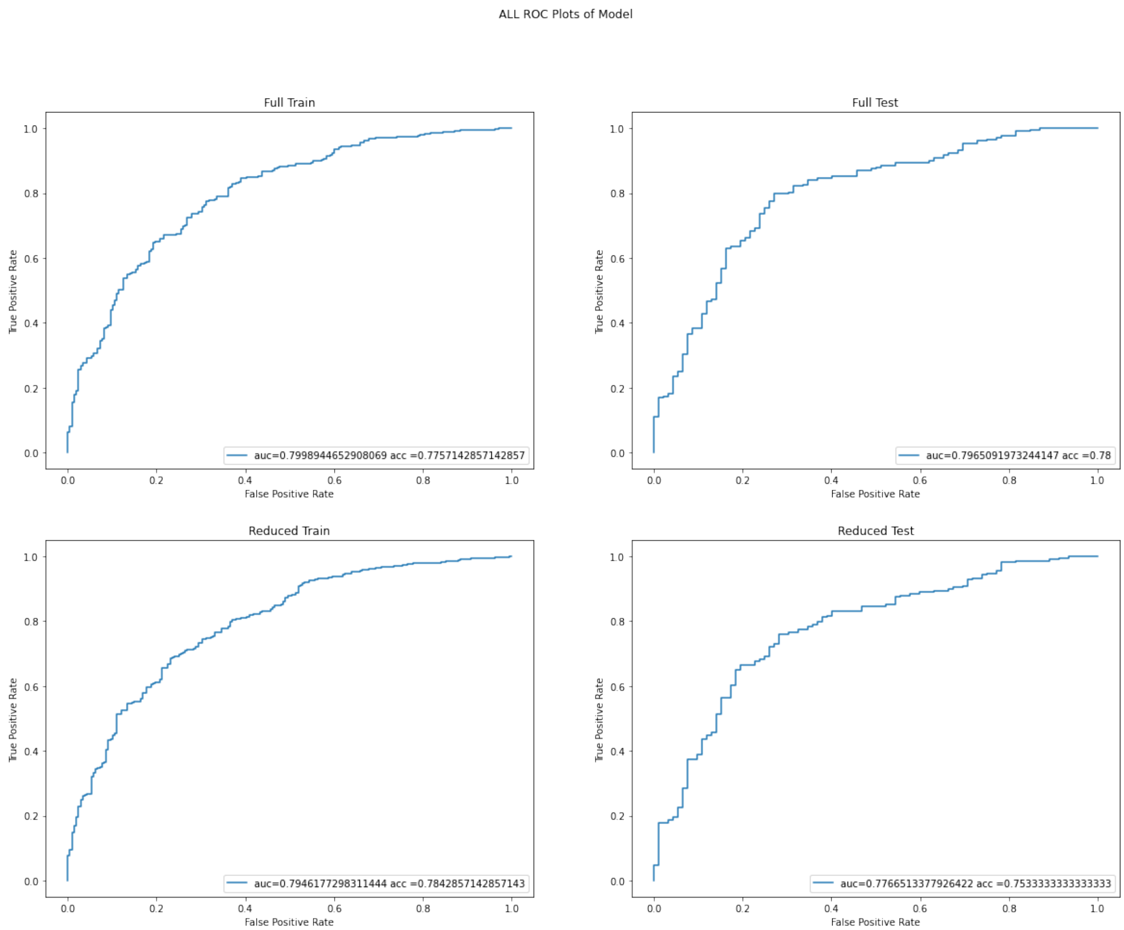
[15]:
```
fpr1, tpr1, _ = metrics.roc_curve(y_train,  y1)
fpr2, tpr2, _ = metrics.roc_curve(y_test,  y2)
fpr3, tpr3, _ = metrics.roc_curve(y_train,  y3)
fpr4, tpr4, _ = metrics.roc_curve(y_test,  y4)
auc1 = metrics.auc(fpr1, tpr1)
auc2 = metrics.auc(fpr2, tpr2)
auc3 = metrics.auc(fpr3, tpr3)
auc4 = metrics.auc(fpr4, tpr4)
```

[16]:
```
from sklearn.metrics import accuracy_score
```

[17]:
```
a1 = 0
a2 = 0
a3 = 0
a4 = 0
for t in np.linspace(0,1,101):
    y_pred1 = np.where(y1>t, 1, 0)
    if accuracy_score(y_train, y_pred1) > a1:
        a1 = accuracy_score(y_train, y_pred1)
for t in np.linspace(0,1,101):
    y_pred2 = np.where(y2>t, 1, 0)
    if accuracy_score(y_test, y_pred2) > a2:
        a2 = accuracy_score(y_test, y_pred2)
for t in np.linspace(0,1,101):
    y_pred3 = np.where(y3>t, 1, 0)
    if accuracy_score(y_train, y_pred3) > a3:
        a3 = accuracy_score(y_train, y_pred3)
for t in np.linspace(0,1,101):
    y_pred4 = np.where(y4>t, 1, 0)
    if accuracy_score(y_test, y_pred4) > a4:
        a4 = accuracy_score(y_test, y_pred4)
```

[18]:
```
fig, axs = plt.subplots(2, 2, figsize=(20, 15))
fig.suptitle('ALL ROC Plots of Model')
axs[0,0].plot(fpr1,tpr1,label="auc="+str(auc1)+" acc ="+str(a1))
axs[0,0].set_title('Full Train')

axs[0,1].plot(fpr2,tpr2,label="auc="+str(auc2)+" acc ="+str(a2))
axs[0,1].set_title('Full Test')

axs[1,0].plot(fpr3,tpr3,label="auc="+str(auc3)+" acc ="+str(a3))
axs[1,0].set_title('Reduced Train')

axs[1,1].plot(fpr4,tpr4,label="auc="+str(auc4)+" acc ="+str(a4))
axs[1,1].set_title('Reduced Test')

for ax in axs.flat:
```

```
    ax.set(xlabel='False Positive Rate', ylabel='True Positive Rate')
    ax.legend(loc=4)
plt.show()
```



The model perfroms okay on predicting the credit risk. Both models has an optimal accuracy slightly below 80% on the train dataset and seems to not overfit as ROC Curves between the train and test sets are very similar. The Full Model Performs the best and doesn't overfit so it isn't too complex/redundant.

[19]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
```

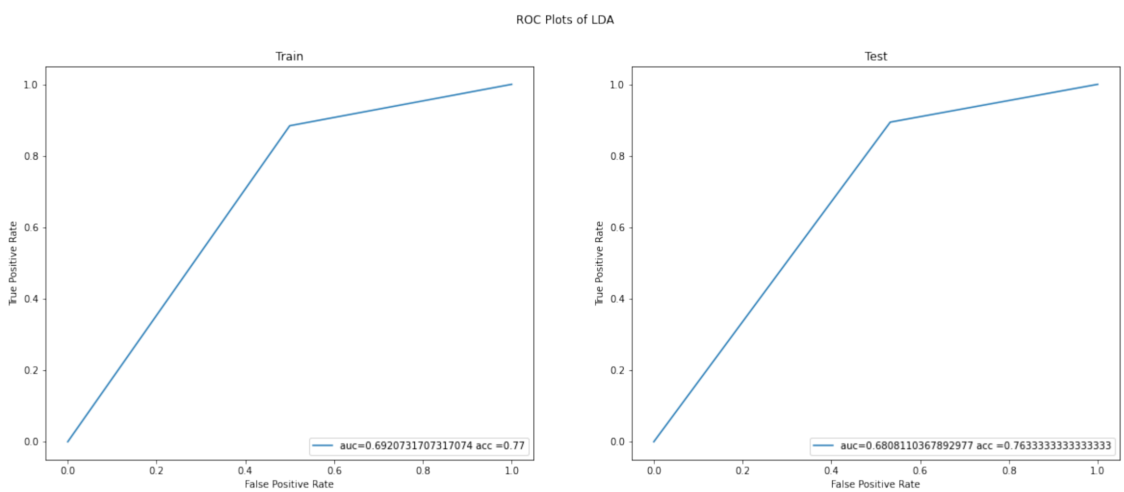
[20]:
```
clfL=LDA()
clfL.fit(X_train,y_train.values.ravel())
```

[20]:
```
LinearDiscriminantAnalysis()
```

[21]:
```
y_1 = clfL.predict(X_train)
y_2 = clfL.predict(X_test)
```

[22]:
```python
fpr_1, tpr_1, _ = metrics.roc_curve(y_train,  y_1)
fpr_2, tpr_2, _ = metrics.roc_curve(y_test,  y_2)
a_1 = accuracy_score(y_train, y_1)
a_2 = accuracy_score(y_test, y_2)
auc_1 = metrics.auc(fpr_1, tpr_1)
auc_2 = metrics.auc(fpr_2, tpr_2)
```

[23]:
```python
fig, axs = plt.subplots(1, 2, figsize=(20, 7.5))
fig.suptitle('ROC Plots of LDA')
axs[0].plot(fpr_1,tpr_1,label="auc="+str(auc_1)+" acc ="+str(a_1))
axs[0].set_title('Train')

axs[1].plot(fpr_2,tpr_2,label="auc="+str(auc_2)+" acc ="+str(a_2))
axs[1].set_title('Test')

for ax in axs.flat:
    ax.set(xlabel='False Positive Rate', ylabel='True Positive Rate')
    ax.legend(loc=4)
plt.show()
```



[24]:
```python
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
```

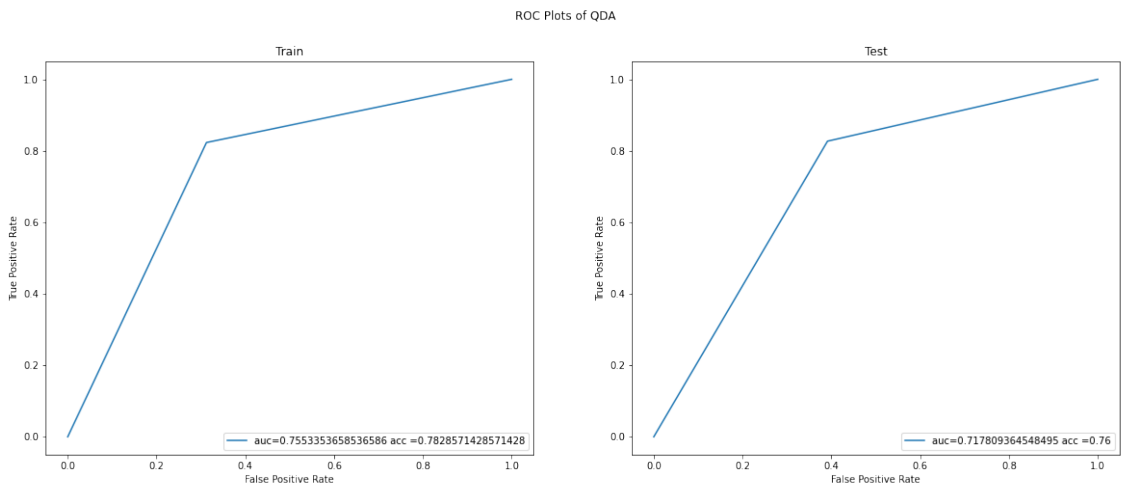
[25]:
```python
clfQ=QDA()
clfQ.fit(X_train,y_train.values.ravel())
```

[25]: QuadraticDiscriminantAnalysis()

[26]:
```python
y_3 = clfQ.predict(X_train)
y_4 = clfQ.predict(X_test)
```

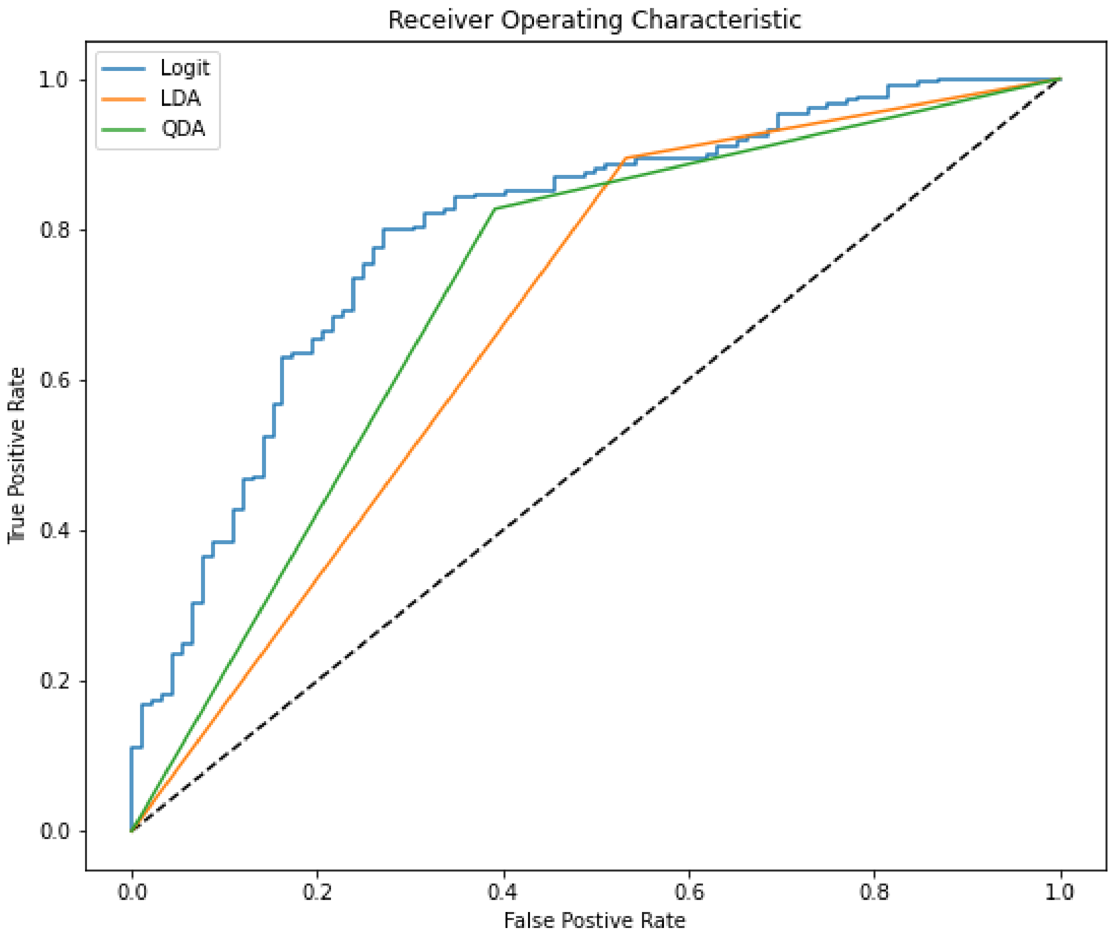
[27]:
```
fpr_3, tpr_3, _ = metrics.roc_curve(y_train,  y_3)
fpr_4, tpr_4, _ = metrics.roc_curve(y_test,  y_4)
a_3 = accuracy_score(y_train, y_3)
a_4 = accuracy_score(y_test, y_4)
auc_3 = metrics.auc(fpr_3, tpr_3)
auc_4 = metrics.auc(fpr_4, tpr_4)
```

[28]:
```
fig, axs = plt.subplots(1, 2, figsize=(20, 7.5))
fig.suptitle('ROC Plots of QDA')
axs[0].plot(fpr_3,tpr_3,label="auc="+str(auc_3)+" acc ="+str(a_3))
axs[0].set_title('Train')

axs[1].plot(fpr_4,tpr_4,label="auc="+str(auc_4)+" acc ="+str(a_4))
axs[1].set_title('Test')

for ax in axs.flat:
    ax.set(xlabel='False Positive Rate', ylabel='True Positive Rate')
    ax.legend(loc=4)
plt.show()
```



[43]:
```
plt.figure(figsize=(9, 7.5))
plt.plot([0,1],[0,1], 'k--')
plt.plot(fpr2, tpr2, label= "Logit")
plt.plot(fpr_2, tpr_2, label= "LDA")
plt.plot(fpr_4, tpr_4, label= "QDA")
plt.legend()
plt.xlabel("False Postive Rate")
plt.ylabel("True Positive Rate")
plt.title('Receiver Operating Characteristic')
```

[43]: Text(0.5, 1.0, 'Receiver Operating Characteristic')



Comparing the ROC curves of the 3 Models, it is clear that the Logistic Regression model is the most superior as it has a higher gain (Better in the highest Deciles). For LDA has TPR of <.4 for a FPR of .2 and QDA has a TPR <.5 at the same FPR. Logistic Regression has the Highest TPR at the .2 FPR level.