Asigment 2

Student: Andres Urrego Angel

Course: Computational Applied Statistics

Deployement process

1. FIrst of all the dataset is loaded in a dataframe, then I created a new dataframe 'X' to allocate the predictors 'Income', 'Limit', 'Rating', 'Cards', 'Age', and 'Education'. Finally I print a sample data along with a description of the predictors to know deeper the trend.

Important notes

- I have defined a method binarize data to transform the values of a column ('balance')
- I add the new colum target 'y' to the dataframe X

In [1]:
```
import pandas as pd
pd.options.mode.chained_assignment = None

# Method to binarize a column
def binarize_data(field):
    values = []
    for number in field:
        if (number > 1500):
             value = 1
        else:
            value = 0
        values.append(value)
    return values


#Read the file
Data = pd.read_csv('Credit.csv')
df = Data[['Income', 'Limit', 'Rating', 'Cards', 'Age', 'Education','Balance']
] # Create the dataframe X with the predictors required

# Binarize the column Balance and join it to the dataframe X
balance_to_binarize = Data['Balance']
balance_y = (binarize_data(balance_to_binarize))
df['y'] =  binarize_data(Data['Balance'])

# Few validations for a sample data and descriptions
print(df.head(n = 5))
df.describe()
```

```
    Income  Limit  Rating  Cards  Age  Education  Balance  y
0   14.891   3606     283      2   34         11      333  0
1  106.025   6645     483      3   82         15      903  0
2  104.593   7075     514      4   71         11      580  0
3  148.924   9504     681      3   36         11      964  0
4   55.882   4897     357      2   68         16      331  0
```

Out[1]:

	Income	Limit	Rating	Cards	Age	Education	Balance	
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400
mean	45.218885	4735.600000	354.940000	2.957500	55.667500	13.450000	520.015000	0.0
std	35.244273	2308.198848	154.724143	1.371275	17.249807	3.125207	459.758877	0.1
min	10.354000	855.000000	93.000000	1.000000	23.000000	5.000000	0.000000	0.0
25%	21.007250	3088.000000	247.250000	2.000000	41.750000	11.000000	68.750000	0.0
50%	33.115500	4622.500000	344.000000	3.000000	56.000000	14.000000	459.500000	0.0
75%	57.470750	5872.750000	437.250000	4.000000	70.000000	16.000000	863.000000	0.0
max	186.634000	13913.000000	982.000000	9.000000	98.000000	20.000000	1999.000000	1.0

Deploy prediction models

I deploy a linear regression model for predictors picked out in the dataframe X and the target is the new binarized column 'Balance'. For this purpose I use the library sklearn.

Linear Regression

In [2]:
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X= df[['Income', 'Limit', 'Rating', 'Cards', 'Age', 'Education']], y =
df['y'])
```

Out[2]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Linear Discriminant

In [3]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
X = df[['Income', 'Limit', 'Rating', 'Cards', 'Age', 'Education']]
y = df['y']
lda.fit(X,y)
```

Out[3]:
```
LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
                           solver='svd', store_covariance=False, tol=0.0001)
```

Cuadratic Discriminant

In [4]:
```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
qda = QuadraticDiscriminantAnalysis()
X = df[['Income', 'Limit', 'Rating', 'Cards', 'Age', 'Education']]
y = df['y']
qda.fit(X,y)
```

Out[4]:
```
QuadraticDiscriminantAnalysis(priors=None, reg_param=0.0,
                              store_covariance=False, store_covariances=None, tol=0.0001)
```

Perform predictions

Based on the assignment, once I have setup the three models above is time to deploy a predictions for different predictors values considering the below X's:

- x1= 'Income' = 63, 'Limit' = 8100, 'Rating' = 600, 'Cards' = 4, 'Age' = 30, 'Education' =14
- x2= 'Income' = 186, 'Limit' = 13414, 'Rating' = 950, 'Cards' = 2, 'Age' = 41, 'Education' =13

LR : Linear Regression

LD : Linear Discriminant

CD : Cuadratic Discriminant

In [15]:
```
import numpy as np

predictors1 = np.array([63,810,600,4,30,14])
predictors2 = np.array([186,13414,950,2,41,13])

#Linear Regression
print('-'*10+'Linear Regression'+'-'*10)
print('LR - based on X1 the balance prediction is: {}'.format(lr.predict(predi
ctors1.reshape(1,6))))
print('LR - based on X2 the balance prediction is: {}'.format(lr.predict(predi
ctors2.reshape(1,6))))

#Linear Discriminant
print('-'*10+'Linear Discriminant'+'-'*10)
print('LD - based on X1 the balance prediction is: {}'.format(lda.predict_prob
a(predictors1.reshape(-1,6))))
print('LD - based on X2 the balance prediction is: {}'.format(lda.predict_prob
a(predictors2.reshape(-1,6))))

#Quadratic Discriminant
print('-'*10+'Quadratic Discriminant'+'-'*10)
print('CD - based on X1 the balance prediction is: {}'.format(qda.predict_prob
a(predictors1.reshape(-1,6))))
print('CD - based on X2 the balance prediction is: {}'.format(qda.predict_prob
a(predictors2.reshape(-1,6))))
```

```
----------Linear Regression----------
LR - based on X1 the balance prediction is: [ 0.14925515]
LR - based on X2 the balance prediction is: [ 0.25937594]
----------Linear Discriminant----------
LD - based on X1 the balance prediction is: [[ 0.72741979  0.27258021]]
LD - based on X2 the balance prediction is: [[ 0.00721199  0.99278801]]
----------Quadratic Discriminant----------
CD - based on X1 the balance prediction is: [[  1.51455739e-154   1.00000000e+0
00]]
CD - based on X2 the balance prediction is: [[  7.83057752e-04   9.99216942e-01
]]
```

Conclusion

After generate the outcome for each model based in the two X's inputs is clearly remarkable that the deployment was performed from the most simpliest model to the most complex in terms of troughouput.

The linear regression generates a prediction even un redeable in terms of accuracy therefore based in this model likely the target **Y** (Balance) would be zero.

In the other hand the linear discriminant is closer to an accurate prediction and the **X2** seems to be closer to came up 1 over the **X1**.

Finally the last prediction based in quadratic discriminant shows up how surely based in a more complex but effective calculation the **X2** has more chance to become 1 for the target **Y** (balance).