Assignment 2

- Donwload the credit dataset 'Credit.csv' from https://github.com/vahidpartovinia/ycbs255/ (https://github.com/vahidpartovinia/ycbs255/)

Setup:

In [12]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

path = "Desktop/"
filename = "Credit.csv"
data = pd.read_csv(path+filename)

data.head()
```

Out[12]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married
0	1	14.891	3606	283	2	34	11	Male	No	Yes
1	2	106.025	6645	483	3	82	15	Female	Yes	Yes
2	3	104.593	7075	514	4	71	11	Male	No	No
3	4	148.924	9504	681	3	36	11	Female	No	No
4	5	55.882	4897	357	2	68	16	Male	No	Yes

Create the attribute matrix 'X' as a subset of the data that includes only the qualitative attributes 'Income', 'Limit', 'Rating', 'Cards', 'Age', and 'Education'

In [13]:

```
X = data[['Income', 'Limit', 'Rating','Cards','Age','Education']]
X.head()
```

Out[13]:

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

Create a discrete response variable, say 'y' by transforming 'Balance' to a binary output, which equals 1 if 'Balance > 1500' and equals 0 otherwise.

In [21]:

```
#To add y column to the original table
data['y'] = np.where(data['Balance']>1500, 1, 0)
data.head()
```

Out[21]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married
0	1	14.891	3606	283	2	34	11	Male	No	Yes
1	2	106.025	6645	483	3	82	15	Female	Yes	Yes
2	3	104.593	7075	514	4	71	11	Male	No	No
3	4	148.924	9504	681	3	36	11	Female	No	No
4	5	55.882	4897	357	2	68	16	Male	No	Yes

Fit 1- logistic regression, 2- linear discriminant, and 3- quadratic discriminant on the binary Balance as the output variable. Use 'Income', 'Limit', 'Rating', 'Cards', 'Age', and 'Education' as the input variables.

In [39]:

```
#Logistic regression:
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
predicators = data[['Income', 'Limit', 'Rating', 'Cards', 'Age','Education']]
prediction = data["y"]

logreg = LogisticRegression()
logreg.fit(predicators,prediction)
```

Out[39]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_interce
pt=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jo
bs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0
.0001,
          verbose=0, warm_start=False)
```

In [55]:

```
#linear discriminant
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

lda = LinearDiscriminantAnalysis()
lda.fit(predicators,prediction)
```

Out[55]:

```
LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage
=None,
              solver='svd', store_covariance=False, tol=0.0001)
```

In [56]:

```
#quadratic discriminant
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
qda = QuadraticDiscriminantAnalysis()
qda.fit(predicators,prediction)
```

Out[56]:

```
QuadraticDiscriminantAnalysis(priors=None, reg_param=0.0,
               store_covariance=False, store_covariances=None, tol=0
.0001)
```

Find the probability of (Balance > 1500), for x1 and x2 below, using these three methods. Compare these probabilities and comment.

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

In [59]:

```
#Linear regression
linear_reg_pred = np.array([[63,8100,600,4,30,14],[186,13414,950,2,41,13]])
print(linear_reg_pred)
print(logreg.predict_proba(linear_reg_pred))

print(logreg.predict(linear_reg_pred))
```

```
[[   63  8100   600     4    30    14]
 [  186 13414   950     2    41    13]]
[[ 0.93231523  0.06768477]
 [ 0.17298983  0.82701017]]
[0 1]
```

The probability that Balance > 1500 for x1 is 0.06768477, and for x2 is 0.06768477

In [60]:

```
#Linear discriminant
print(lda.predict_proba(linear_reg_pred))
print(lda.predict(linear_reg_pred))
```

```
[[ 0.94144572  0.05855428]
 [ 0.00721199  0.99278801]]
[0 1]
```

The probability that Balance > 1500 for x1 is 0.05855428, and for x2 is 0.99278801

In [62]:

```
#quadratic discriminant
print(qda.predict_proba(linear_reg_pred))
print(qda.predict(linear_reg_pred))
```

```
[[  9.99999999e-01   1.24419207e-09]
 [  7.83057752e-04   9.99216942e-01]]
[0 1]
```

The probability that Balance > 1500 for x1 is 1.24419207e-09, and for x2 is 9.99216942e-01