Desarrollo Laboratorio 16

In [1]:
```
import numpy as np
import pandas as pd
import os
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysi
from sklearn.metrics import confusion_matrix, classification_report, precision_score
from sklearn.model_selection import train_test_split

import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
os.chdir("D:\Social Data Consulting\Python for Data Science\data")
```

1. Solo considera las columnas 'Ri','Na','Mg','Al','Si','K','Ca','Ba','Fe','Y' y reserva el testeo en 30%

In [3]:
```
datos='glass.data'
df_glass=pd.read_csv(datos,delimiter=",",names=['Nro','Ri','Na','Mg','Al','Si','K','Ca','Ba','Fe',
df_glass.info()
del df_glass['Nro']
df_glass.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Nro     214 non-null    int64  
 1   Ri      214 non-null    float64
 2   Na      214 non-null    float64
 3   Mg      214 non-null    float64
 4   Al      214 non-null    float64
 5   Si      214 non-null    float64
 6   K       214 non-null    float64
 7   Ca      214 non-null    float64
 8   Ba      214 non-null    float64
 9   Fe      214 non-null    float64
 10  Y       214 non-null    int64  
dtypes: float64(9), int64(2)
memory usage: 18.5 KB
```

Out[3]:

	Ri	Na	Mg	Al	Si	K	Ca	Ba	Fe	Y
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

In [4]:
```
X=df_glass.iloc[:,:9]
y=df_glass.iloc[:,9]
```

In [5]:
```
X_train,X_test,y_train,y_test=train_test_split(X,
                                               y,
                                               test_size=0.3,
                                               stratify=y)
```

2. Encontrar los priori

In [6]:

```
lda = LinearDiscriminantAnalysis()
model = lda.fit(X_train,# los valores de los predictores de entrenamiento
                y_train)#los valores de y de entrenamiento

print("Las probabilidades a priori son:\n")
print("Probabilidad a Priori grupo I:",model.priors_[0].round(3))
print("Probabilidad a Priori grupo II:",model.priors_[1].round(3))
print("Probabilidad a Priori grupo III:",model.priors_[2].round(3))
print("Probabilidad a Priori grupo IV:",model.priors_[3].round(3))
print("Probabilidad a Priori grupo V:",model.priors_[4].round(3))
print("Probabilidad a Priori grupo VI:",model.priors_[5].round(3))
```

```
Las probabilidades a priori son:

Probabilidad a Priori grupo I: 0.329
Probabilidad a Priori grupo II: 0.356
Probabilidad a Priori grupo III: 0.081
Probabilidad a Priori grupo IV: 0.06
Probabilidad a Priori grupo V: 0.04
Probabilidad a Priori grupo VI: 0.134
```

3. Encontrar los coeficientes

In [7]:

```
print("Los coeficientes estimados para la Función Discriminante es:")
print(model.coef_)
```

```
Los coeficientes estimados para la Función Discriminante es:
[[ 6.58288716e+01 -1.59308461e+00  7.93942854e-01 -2.63653199e+00
  -1.02547488e+00 -6.60713420e-01 -5.13618157e-01 -1.98250907e+00
   3.45646769e-01]
 [-8.44653357e+01 -5.33180058e+00 -3.39070433e+00 -5.11239396e+00
  -4.90645133e+00 -4.33809613e+00 -3.89097163e+00 -5.56832660e+00
   2.17799256e+00]
 [-1.40571379e+03 -4.52112034e+00 -1.42784354e+00 -7.12939833e+00
  -6.17196031e+00 -3.87391999e+00 -9.32040820e-01 -3.07821332e+00
  -3.91569739e+00]
 [-2.20004143e+01 -2.68100074e-01 -2.48952574e+00  2.36509913e+00
  -4.91859588e-01  2.38922252e+00  6.01612005e-01 -3.12842808e+00
  -4.49877717e+00]
 [ 2.41417408e+02  7.06328456e+00 -1.77805236e+00  4.17171779e+00
   4.47369364e+00  1.84920848e+00  1.15294200e+00  1.35262387e+00
  -1.60152793e+00]
 [ 8.43455642e+02  1.87466607e+01  9.55061489e+00  2.19691764e+01
   1.80969144e+01  1.38091419e+01  1.15120558e+01  2.24621462e+01
  -1.76418832e+00]]
```

4. Reporte de clasificacion con la data de testeo

In [8]:

```
pred=model.predict(X_test)
print(np.unique(pred,
                return_counts=True))
```

```
(array([1, 2, 3, 5, 6, 7], dtype=int64), array([21, 27,  3,  5,  1,  8], dtype=int64))
```

In [9]:

```
print(classification_report(y_test, pred, digits=2))
```

```
              precision    recall  f1-score   support

           1       0.76      0.76      0.76        21
           2       0.67      0.78      0.72        23
           3       0.33      0.20      0.25         5
           5       0.40      0.50      0.44         4
           6       0.00      0.00      0.00         3
           7       0.88      0.78      0.82         9

    accuracy                           0.68        65
   macro avg       0.51      0.50      0.50        65
weighted avg       0.65      0.68      0.66        65
```

5. Generar la matriz de confusion

In [10]:

```
print(confusion_matrix(pred, y_test))
```

```
[[16  2  1  0  1  1]
 [ 3 18  3  1  2  0]
 [ 2  0  1  0  0  0]
 [ 0  2  0  2  0  1]
 [ 0  1  0  0  0  0]
 [ 0  0  0  1  0  7]]
```

In [11]:

```
confusion_matrix = pd.crosstab(pred, y_test)
confusion_matrix
```

Out[11]:

Y	1	2	3	5	6	7
row_0						
1	16	2	1	0	1	1
2	3	18	3	1	2	0
3	2	0	1	0	0	0
5	0	2	0	2	0	1
6	0	1	0	0	0	0
7	0	0	0	1	0	7