In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn import svm
from sklearn import preprocessing, model_selection, neighbors, discriminant_analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
```

In [2]:
```
file = ("glass.data")
df = pd.read_csv(file,delim_whitespace=False, header=None)
columns = ["Id", "Ri", "Na", "Mg", "Al", "Si", "K", "Ca", "Ba", "Fe", "Type"]
df.columns = columns
df
```

Out[2]:

	Id	Ri	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
1	2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...
209	210	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	211	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	212	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	213	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	214	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 11 columns

In [3]:
```
x = df.iloc[:,1:10]
y = df.iloc[:,-1:]
x , y
```

Out[3]:
```
(          Ri     Na    Mg    Al     Si     K     Ca    Ba   Fe
 0    1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.00  0.0
 1    1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0
 2    1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0
 3    1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0
 4    1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0
 ..       ...    ...   ...   ...    ...   ...   ...   ...  ...
 209  1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0
 210  1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0
 211  1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0
 212  1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0
 213  1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0

 [214 rows x 9 columns],
      Type
 0       1
 1       1
 2       1
 3       1
 4       1
 ..    ...
 209     7
 210     7
 211     7
 212     7
 213     7

 [214 rows x 1 columns])
```

In [4]:
```
x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.3,random_state=3)
```

In [5]:
```
x_train,y_train
```

Out[5]:
```
(          Ri     Na    Mg    Al     Si     K     Ca    Ba    Fe
 30   1.51768  12.65  3.56  1.30  73.08  0.61   8.69  0.00  0.14
 74   1.51596  13.02  3.56  1.54  73.11  0.72   7.90  0.00  0.00
 102  1.51820  12.62  2.76  0.83  73.81  0.35   9.42  0.00  0.20
 88   1.51618  13.01  3.50  1.48  72.89  0.60   8.12  0.00  0.00
 86   1.51569  13.24  3.49  1.47  73.25  0.38   8.03  0.00  0.00
 ..       ...    ...   ...   ...    ...   ...    ...   ...   ...
 200  1.51508  15.15  0.00  2.25  73.50  0.00   8.34  0.63  0.00
 184  1.51115  17.38  0.00  0.34  75.41  0.00   6.65  0.00  0.00
 131  1.52614  13.70  0.00  1.36  71.24  0.19  13.44  0.00  0.10
 152  1.51779  13.64  3.65  0.65  73.00  0.06   8.93  0.00  0.00
 106  1.53125  10.73  0.00  2.10  69.81  0.58  13.30  3.15  0.28

 [149 rows x 9 columns],
      Type
 30      1
 74      2
 102     2
 88      2
 86      2
 ..    ...
 200     7
 184     6
 131     2
 152     3
 106     2

 [149 rows x 1 columns])
```

In [6]:
```
x_train.shape, y_train.shape
```

Out[6]:
```
((149, 9), (149, 1))
```

In [7]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
```

In [8]:
```
x_train,y_train
```

Out[8]:
```
(          Ri     Na    Mg    Al     Si     K     Ca    Ba    Fe
 30   1.51768  12.65  3.56  1.30  73.08  0.61   8.69  0.00  0.14
 74   1.51596  13.02  3.56  1.54  73.11  0.72   7.90  0.00  0.00
 102  1.51820  12.62  2.76  0.83  73.81  0.35   9.42  0.00  0.20
 88   1.51618  13.01  3.50  1.48  72.89  0.60   8.12  0.00  0.00
 86   1.51569  13.24  3.49  1.47  73.25  0.38   8.03  0.00  0.00
 ..       ...    ...   ...   ...    ...   ...    ...   ...   ...
 200  1.51508  15.15  0.00  2.25  73.50  0.00   8.34  0.63  0.00
 184  1.51115  17.38  0.00  0.34  75.41  0.00   6.65  0.00  0.00
 131  1.52614  13.70  0.00  1.36  71.24  0.19  13.44  0.00  0.10
 152  1.51779  13.64  3.65  0.65  73.00  0.06   8.93  0.00  0.00
 106  1.53125  10.73  0.00  2.10  69.81  0.58  13.30  3.15  0.28

 [149 rows x 9 columns],
     Type
 30     1
 74     2
 102    2
 88     2
 86     2
 ..   ...
 200    7
 184    6
 131    2
 152    3
 106    2

 [149 rows x 1 columns])
```

In [9]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

LINEAR

In [10]:
```
clf = LinearDiscriminantAnalysis().fit(x_train,y_train)
```

```
/Users/georgepsaltakis/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [11]:
```
y_pred=clf.predict(x_test)
```

In [12]:
```
y_pred
```

Out[12]:
```
array([1, 1, 7, 6, 2, 7, 7, 2, 1, 7, 2, 2, 2, 1, 5, 6, 2, 7, 2, 2, 7, 7,
       5, 1, 3, 2, 7, 2, 1, 6, 1, 2, 5, 1, 1, 1, 2, 2, 6, 1, 7, 1, 1, 1,
       7, 1, 1, 1, 7, 2, 1, 2, 1, 1, 2, 7, 1, 1, 1, 1, 2, 2, 1, 6, 1])
```

In [13]:
```
clf.score(x_test, y_test)
```

Out[13]:
```
0.7076923076923077
```

In [14]:
```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```

In [15]:
```
conf_m = confusion_matrix(y_test, y_pred)
```

In [16]:
```
conf_m
```

Out[16]:
```
array([[18,  3,  0,  0,  0,  0],
       [ 5, 12,  0,  1,  2,  0],
       [ 3,  1,  1,  0,  0,  0],
       [ 0,  1,  0,  2,  1,  1],
       [ 0,  0,  0,  0,  2,  0],
       [ 0,  1,  0,  0,  0, 11]])
```

In [17]:
```
report = classification_report(y_test, y_pred)
```

In [18]:
```
print('report:', report, sep='\n')
```

```
report:
              precision    recall  f1-score   support

           1       0.69      0.86      0.77        21
           2       0.67      0.60      0.63        20
           3       1.00      0.20      0.33         5
           5       0.67      0.40      0.50         5
           6       0.40      1.00      0.57         2
           7       0.92      0.92      0.92        12

    accuracy                           0.71        65
   macro avg       0.72      0.66      0.62        65
weighted avg       0.74      0.71      0.69        65
```

QUADRATIC

In [19]:
```
x_train,y_train
```

Out[19]:
```
(          Ri     Na    Mg    Al     Si     K     Ca    Ba    Fe
 30   1.51768  12.65  3.56  1.30  73.08  0.61   8.69  0.00  0.14
 74   1.51596  13.02  3.56  1.54  73.11  0.72   7.90  0.00  0.00
 102  1.51820  12.62  2.76  0.83  73.81  0.35   9.42  0.00  0.20
 88   1.51618  13.01  3.50  1.48  72.89  0.60   8.12  0.00  0.00
 86   1.51569  13.24  3.49  1.47  73.25  0.38   8.03  0.00  0.00
 ..       ...    ...   ...   ...    ...   ...    ...   ...   ...
 200  1.51508  15.15  0.00  2.25  73.50  0.00   8.34  0.63  0.00
 184  1.51115  17.38  0.00  0.34  75.41  0.00   6.65  0.00  0.00
 131  1.52614  13.70  0.00  1.36  71.24  0.19  13.44  0.00  0.10
 152  1.51779  13.64  3.65  0.65  73.00  0.06   8.93  0.00  0.00
 106  1.53125  10.73  0.00  2.10  69.81  0.58  13.30  3.15  0.28

 [149 rows x 9 columns],
      Type
 30      1
 74      2
 102     2
 88      2
 86      2
 ..    ...
 200     7
 184     6
 131     2
 152     3
 106     2

 [149 rows x 1 columns])
```

In [20]:
```
clf = QuadraticDiscriminantAnalysis().fit(x_train,y_train)
```

```
/Users/georgepsaltakis/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/Users/georgepsaltakis/opt/anaconda3/lib/python3.9/site-packages/sklearn/discriminant_analysis.py:878: UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")
```

In [21]:
```
y_pred=clf.predict(x_test)
```

In [22]:
```
clf.score(x_test, y_test)
```

Out[22]:
```
0.5846153846153846
```

In [23]:
```
conf_m = confusion_matrix(y_test, y_pred)
```

In [24]:
```
conf_m
```

Out[24]:
```
array([[20,  1,  0,  0,  0,  0],
       [12,  6,  1,  0,  1,  0],
       [ 3,  1,  1,  0,  0,  0],
       [ 0,  4,  0,  0,  0,  1],
       [ 0,  1,  0,  0,  0,  1],
       [ 1,  0,  0,  0,  0, 11]])
```

In [25]:
```
report = classification_report(y_test, y_pred)
```

```
/Users/georgepsaltakis/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defin
ed and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/georgepsaltakis/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defin
ed and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/georgepsaltakis/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defin
ed and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [26]:
```
print('report:', report, sep='\n')
```

```
report:
              precision    recall  f1-score   support

           1       0.56      0.95      0.70        21
           2       0.46      0.30      0.36        20
           3       0.50      0.20      0.29         5
           5       0.00      0.00      0.00         5
           6       0.00      0.00      0.00         2
           7       0.85      0.92      0.88        12

    accuracy                           0.58        65
   macro avg       0.39      0.39      0.37        65
weighted avg       0.52      0.58      0.52        65
```

In [ ]: