In [31]:

```
#Loading the librariess
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, accuracy_sc

#Loading the dataset
data = pd.read_csv("C:\\Users\\G.Madhu mitha\\OneDrive\\Documents\\.WINTER SEM 2022-23\\
data
```

Out[31]:

	Id_number	refractive_index	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 [16]:

```
#Splitting the data into train and test set
X = data.drop('Type', axis=1)
y = data['Type']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
X_train
```

Out[16]:

	Id_number	refractive_index	Na	Mg	Al	Si	K	Ca	Ba	Fe
137	138	1.51711	12.89	3.62	1.57	72.96	0.61	8.11	0.00	0.00
65	66	1.52099	13.69	3.59	1.12	71.96	0.09	9.40	0.00	0.00
108	109	1.52222	14.43	0.00	1.00	72.67	0.10	11.52	0.00	0.08
181	182	1.51888	14.99	0.78	1.74	72.50	0.00	9.95	0.00	0.00
31	32	1.51747	12.84	3.50	1.14	73.27	0.56	8.55	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...
106	107	1.53125	10.73	0.00	2.10	69.81	0.58	13.30	3.15	0.28
14	15	1.51763	12.61	3.59	1.31	73.29	0.58	8.50	0.00	0.00
92	93	1.51588	13.12	3.41	1.58	73.26	0.07	8.39	0.00	0.19
179	180	1.51852	14.09	2.19	1.66	72.67	0.00	9.32	0.00	0.00
102	103	1.51820	12.62	2.76	0.83	73.81	0.35	9.42	0.00	0.20

149 rows × 10 columns

In [17]:

```
X_test
```

Out[17]:

	Id_number	refractive_index	Na	Mg	Al	Si	K	Ca	Ba	Fe
9	10	1.51755	13.00	3.60	1.36	72.99	0.57	8.40	0.00	0.11
197	198	1.51727	14.70	0.00	2.34	73.28	0.00	8.95	0.66	0.00
66	67	1.52152	13.05	3.65	0.87	72.22	0.19	9.85	0.00	0.17
191	192	1.51602	14.85	0.00	2.38	73.28	0.00	8.76	0.64	0.09
117	118	1.51708	13.72	3.68	1.81	72.06	0.64	7.88	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...
5	6	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0.00	0.26
135	136	1.51789	13.19	3.90	1.30	72.33	0.55	8.44	0.00	0.28
56	57	1.51215	12.99	3.47	1.12	72.98	0.62	8.35	0.00	0.31
199	200	1.51609	15.01	0.00	2.51	73.05	0.05	8.83	0.53	0.00
173	174	1.52043	13.38	0.00	1.40	72.25	0.33	12.50	0.00	0.00

65 rows × 10 columns

In [18]:

```
y_test
```

Out[18]:

```
9      1
197    7
66     1
191    7
117    2
      ..
5      1
135    2
56     1
199    7
173    5
Name: Type, Length: 65, dtype: int64
```

In [19]:

```
y_train
```

Out[19]:

```
137    2
65     1
108    2
181    6
31     1
      ..
106    2
14     1
92     2
179    6
102    2
Name: Type, Length: 149, dtype: int64
```

In [20]:

```
# Normalize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [22]:

```
mlp = MLPClassifier(hidden_layer_sizes=(10, 5), activation='relu', solver='adam', random
mlp
```

Out[22]:

```
MLPClassifier(hidden_layer_sizes=(10, 5), random_state=42)
```

In [23]:

```
#Fit the mlp model on training data
mlp.fit(X_train, y_train)
```

```
C:\Users\G.Madhu mitha\anaconda3\lib\site-packages\sklearn\neural_network
\_multilayer_perceptron.py:582: ConvergenceWarning: Stochastic Optimizer:
Maximum iterations (200) reached and the optimization hasn't converged ye
t.
  warnings.warn(
```

Out[23]:

```
MLPClassifier(hidden_layer_sizes=(10, 5), random_state=42)
```

In [28]:

```
#Predicting class label for test data
y_pred = mlp.predict(X_test)
y_pred
```

Out[28]:

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

In [36]:

```
#Evaluate the performance measures
accuracy =accuracy_score(y_test,y_pred)
print(accuracy)
precision = precision_score(y_test,y_pred,average="macro")
print(precision)
recall = recall_score(y_test, y_pred, average='macro')
print(recall)
#Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
0.8769230769230769
0.9325860948667967
0.7339816933638444
[[18  1  0  0  0  0]
 [ 1 22  0  0  0  0]
 [ 0  2  2  0  0  0]
 [ 0  2  0  4  0  0]
 [ 0  0  0  0  1  2]
 [ 0  0  0  0  0 10]]
```

In [26]:

```

```