1. Implement Naïve Bayes method using scikit-learn library Use dataset available with name glass Use train_test_split to create training and testing part Evaluate the model on test part using score and classification_report(y_true, y_pred)

In [9]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
```

In [10]:
```
data = pd.read_csv("glass.csv")
data.head()
```

Out[10]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
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 [11]:
```
x = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
```

In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

In [13]:
```
algo = GaussianNB()
algo.fit(x_train, y_train)
```

Out[13]:
```
GaussianNB()
```

In [14]:
```
y_pred = algo.predict(x_test)
```

In [15]:
```
accuracy = accuracy_score(y_test, y_pred)
```

In [16]:
```
print("Accuracy of Naive Bayes: {:.2f}%".format(accuracy * 100))
print("\nClassification Report:")
print(classification_report(y_test, y_pred, zero_division=1))
```

```
Accuracy of Naive Bayes: 55.81%

Classification Report:
              precision    recall  f1-score   support

           1       0.41      0.64      0.50        11
           2       0.43      0.21      0.29        14
           3       0.40      0.67      0.50         3
           5       0.50      0.25      0.33         4
           6       1.00      1.00      1.00         3
           7       0.89      1.00      0.94         8

    accuracy                           0.56        43
   macro avg       0.60      0.63      0.59        43
weighted avg       0.55      0.56      0.53        43
```

2. Implement linear SVM method using scikit library Use the same dataset above Use train_test_split to create training and testing part Evaluate the model on test part using score and classification_report(y_true, y_pred)

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
```

In [2]:
```
data = pd.read_csv("glass.csv")
data.head()
```

Out[2]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
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 [3]:
```
x = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
```

In [4]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

In [5]:
```
algo = SVC(kernel='linear')
algo.fit(x_train, y_train)
```

Out[5]:
```
SVC(kernel='linear')
```

In [6]:
```
y_pred = algo.predict(x_test)
```

In [7]:
```
accuracy = accuracy_score(y_test, y_pred)
```

In [8]:
```
print("Accuracy of SVM: {:.2f}%".format(accuracy * 100))
print("\nClassification Report:")
print(classification_report(y_test, y_pred, zero_division=1))
```

```
Accuracy of SVM: 74.42%

Classification Report:
              precision    recall  f1-score   support

           1       0.69      0.82      0.75        11
           2       0.67      0.71      0.69        14
           3       1.00      0.00      0.00         3
           5       0.80      1.00      0.89         4
           6       1.00      0.67      0.80         3
           7       0.88      0.88      0.88         8

    accuracy                           0.74        43
   macro avg       0.84      0.68      0.67        43
weighted avg       0.77      0.74      0.72        43
```

3. Which algorithm you got better accuracy? Can you justify why?

Ans. The superiority of Linear SVM over Gaussian Naive Bayes in terms of accuracy might be attributed to Linear SVM's ability to handle complex and non-linearly separable datasets more effectively. This advantage arises from Linear SVM's pursuit of maximizing the margin between data points and the decision boundary, thereby enhancing its robustness to outliers and non-linearities present in the data. Moreover, Linear SVM offers a wider range of options by allowing the use of various kernel functions, which can be tailored to different types of data distributions, further enhancing its adaptability. Conversely, Gaussian Naive Bayes, which assumes feature independence, may not be as resilient as Linear SVM when confronted with complex datasets that exhibit intricate relationships among features.