Neural Networks Assignment 5

ID: 700739769

Name: Anjani Priya Marthati

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]:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Load the glass dataset
glass_data = pd.read_csv("C:\\Neural networks\\NNDL_Code and Data (2)\\NNDL_Code and Data\\glass.csv")

# Split the data into features and target
X = glass_data.iloc[:, :-1].values
y = glass_data.iloc[:, -1].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the GaussianNB model
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Make predictions on the test set
gnb_y_pred = gnb.predict(X_test)

# Evaluate the model performance
gnb_acc = accuracy_score(y_test, gnb_y_pred)
print("GaussianNB Accuracy: {:.2f}%".format(gnb_acc * 100))
print("\nNaïve Bayes Classification Report:")
print(classification_report(y_test, gnb_y_pred, zero_division=1))
```

```
GaussianNB Accuracy: 55.81%

Naïve Bayes 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)
   Which algorithm you got better accuracy? Can you justify why?

In [7]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Load the glass dataset
glass_data = pd.read_csv("C:\\Neural networks\\NNDL_Code and Data (2)\\NNDL_Code and Data\\glass.csv")

# Split the data into features and target
X = glass_data.iloc[:, :-1].values
y = glass_data.iloc[:, -1].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)

# Make predictions on the test set
svm_y_pred = svm.predict(X_test)

# Evaluate the model performance
svm_acc = accuracy_score(y_test, svm_y_pred,)
print("SVM Accuracy: {:.2f}%".format(svm_acc * 100))
print("\nSVM Classification Report:")
print(classification_report(y_test, svm_y_pred, zero_division=1))
```

```
SVM Accuracy: 74.42%

SVM 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
```

Based upon the Data SVM method provided an accuracy of 74.42% and GaussianNB Accuracy is 55.81% . With same test size and random state, Thus this states that SVM Algorithm performs well in this case.

Supporting Factors for these are as follows:

GNB is a simple probabilistic classifier based on Bayes' theorem that is fast to train and easy to interpret. It makes strong independence assumptions between the features, meaning that it assumes that the features are conditionally independent given the class. It is most commonly used for text classification problems where the features are words or n-grams. GNB is a good choice when you have a large number of features, as it scales well to high-dimensional data.

SVM, on the other hand, is a more sophisticated method that uses the concept of margins to find the best decision boundary between classes. SVM can handle complex non-linear relationships between features and is often used for image classification or classification problems with a small number of samples. SVM is also more robust to overfitting than GNB when the number of features is large, so it might be a better choice when dealing with high-dimensional data.

In general, GNB is a good first choice for classification problems, especially when the data is large, the problem is well understood, and computational resources are limited. SVM is a more powerful method that is useful for more complex classification problems, especially when the data is small, the problem is not well understood, or there is a need for more robustness to overfitting.

Conclusion: So here the data we have is only 215 rows which are very limited to train the model and there are multiple parameters which makes the prediction complex. So, SVM Algorithm is best suited for this data since the data is very small and also based upon the reasons mentioned above.

Github Link: https://github.com/Anju369/Assignment-5

Video Link:
https://vimeo.com/manage/videos/912471829/9e6bc26bb0?studio_recording=true&record_session_id=9e7df35f-eb63-45c5-be26-fa134ce5b3a9