Neural Networks Deep Learning – Icp5

Name: Deekshitha, Gaddameedhi **ID: 700755765**

GitHub link: https://github.com/deekshitha430/icp5_neural

Video Link:
https://drive.google.com/file/d/1oXitiiA8ENQXzvZjWeiQ61PW_s9sq43Y/view?usp=sharing

1. Implement Naïve Bayes method using scikit-learn library
   Use dataset available with name glass

[13]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report


data = pd.read_csv('glass.csv')

X = data.drop('Type', axis=1)
y = data['Type']
```

[14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

Use train_test_split to create training and testing part
Evaluate the model on test part using score
classification_report(y_true, y_pred)

Neural Networks Deep Learning – Icp5

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
nb_classifier = GaussianNB()
```

```
nb_classifier.fit(X_train, y_train)

y_pred = nb_classifier.predict(X_test)

acc = nb_classifier.score(X_test, y_test)
print("Accuracy:", acc)

print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.5581395348837209
              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)

Neural Networks Deep Learning – Icp5

```
]: from sklearn.svm import SVC

glass_data = pd.read_csv('glass.csv')
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)

accuracy = svm_classifier.score(X_test, y_test)
print("Accuracy:", accuracy)

print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7441860465116279
              precision    recall  f1-score   support

           1       0.69      0.82      0.75        11
           2       0.67      0.71      0.69        14
           3       0.00      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       0.67      0.68      0.67        43
weighted avg       0.70      0.74      0.72        43

C:\Users\gadda\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: Un
```

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

The glass dataset has a complex decision boundary separating different types of glass based on their chemical composition and properties. SVM's ability to capture non-linear relationships using kernel functions allows it to model this complex decision boundary more effectively compared to Naïve Bayes, which assumes linear independence between features.
It has many features describing the properties of different types of glass, SVM's capability to handle high-dimensional data without overfitting becomes advantageous.
Glass data set has smaller dataset which allows SVM to perform in a generalized way. This simple data enables SVM to capture more intricate decision boundaries, leading to improved classification accuracy.