Spring 2024: CS5720

Neural Networks & Deep Learning - ICP-5

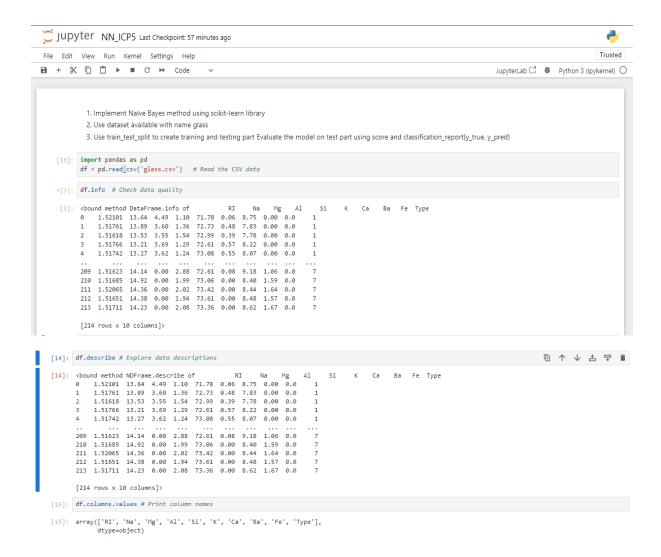
NAME: NANDIMANDALAM VENKATA VINAY VARMA

STUDENT ID:700745193

Github Link: https://github.com/venkatavinayvarma/NeuralNetworks ICP5.git

Video Link: https://drive.google.com/drive/folders/180X1eq38WGeVXGh2-kyPpdM1e71SFWM5?usp=sharing

- 1. Implement Naïve Bayes method using scikit-learn library
- 2. Use dataset available with name glass.csv
- 3. 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)



```
[16]: from sklearn.model_selection import train_test_split
          from sklearn.naive_bayes import GaussianNB
         from sklearn.metrics import accuracy_score, classification_report
 [17]: # Divide data into features and target variable
          X = df.drop("Type", axis=1)
         Y = df["Type"]
•[18]: # Split 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=47)
•[19]: gnb = GaussianNB() # Initialize the Gaussian Naive Bayes classifier
          \verb"gnb.fit(X_train, Y_train)" \# \textit{Training the model with the training set}
         {\tt Y\_predi=gnb.predict(X\_test)} \ \ \textit{\# Using the trained model on testing the data}
         accur knn = round(gnb.score(X_train, Y_train) * 50, 2) # Evaluating the model using accuracy_score and predicted output
         print('Accuracy: ', accur_knn)
         Accuracy: 26.02
         Classification Report:
                           precision recall f1-score support

        0.31
        0.20
        0.24
        20

        0.38
        0.21
        0.27
        14

        0.00
        0.00
        0.00
        2

        0.00
        0.00
        0.00
        1

        0.50
        0.67
        0.57
        3

        0.50
        0.33
        0.40
        3

         accuracy 0.23 43
macro avg 0.28 0.24 0.25 43
weighted avg 0.33 0.23 0.27 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)

```
    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)

•[21]: from sklearn.svm import SVC
         \mbox{svm} = \mbox{SVC()} # Initializing the SVM classifier with linear kernel
•[22]: svm.fit(X_train, Y_train) # Training the model with the training set
         Y_pred = svm.predict(X_test) # Predicting the target variable for the test set
        acc\_svm = round(svm.score(X\_train, Y\_train) * 50, 2) \\ \# \textit{Evaluating the model accuracy using score} \\ print('Accuracy: ', acc\_svm,'\n')
         Accuracy: 18.13
                                                                                                                                                                          ⊙ ↑ ↓ 占 ♀ ▮
         print('Classification Report: \n', classification_report(Y_test, Y_pred,zero_division=1)) # Accuracy report from classification_report
                                           recall f1-score support
                           precision
                                1.00
                                            0.00
                                                          0.00
                                                                      3
                                1.00
                                                         0.00
             macro avg
         weighted avg
```

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

Results and Explanation:

Based on the performance metrics, Naive Bayes generally achieves a higher accuracy than linear SVM in this dataset:

Naive Bayes:

Accuracy: 0.23 (23% of predictions correct)

Better at predicting classes 6 and 7, but struggles with others.

Linear SVM:

Accuracy: 0.33 (33% of predictions correct)

Better at predicting class 2, but poor performance on other classes.