**Neural Networks and Deep Learning – ICP 5**

**GitHub Link:** [**https://github.com/rumanathaskeen22/Neural-Networks-and-Deep-Learning-ICP-5**](https://github.com/rumanathaskeen22/Neural-Networks-and-Deep-Learning-ICP-5)

**Video Link:** Executed in Class

**Name: Lnu Rumana Thaskeen**

**Student ID: 700742859**

**CRN: 23216**

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 this program, I am first Importing the required packages to create arrays which are pandas and numpy. Then I am loading the data set by reading the giving glass.csv file by using the

Functions like the Pandas read\_csv() method which enable you to work with files effectively.

Then I am displaying the structure of data from given file, using the df.head() method. The head () method returns a specified number of rows, string from the top.

X2=df.drop(["Type"], axis=1): This line drops the "Type" column from the dataframe 'df' and assigns the new dataframe to the variable 'X2'. The 'axis' parameter is set to 1, indicating that a column is being dropped, rather than a row.

y=df["Type"]: This line extracts the "Type" column from the original dataframe 'df' and assigns it to a new variable 'y'. Then I have used train\_test\_split to create training and testing part.

I have then imported three modules from the scikit-learn library. GaussianNB: This module implements the Gaussian Naive Bayes algorithm for classification. metrics: This module provides a range of evaluation metrics for classification problems, including precision, recall, f1-score, and support.

accuracy\_score: This function calculates the accuracy of the classification model.

gnb = GaussianNB(): This line creates an instance of the Gaussian Naive Bayes algorithm.

y\_pred = gnb.fit(X\_train, y\_train).predict(X\_test): This line trains the model on the training data (X\_train and y\_train) and uses the predict method to generate predictions for the test data (X\_test). The predictions are stored in the variable 'y\_pred'.

acc\_nb=accuracy\_score(y\_test,y\_pred): This line calculates the accuracy of the model using the accuracy\_score function, which was imported earlier. The accuracy is stored in the variable 'acc\_nb'.

Upon Calculating the % and rounding it off to two digits, we get the Naïve Bayes accuracy as 53.49 and generated a classification report for the model as shown below.

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

1. **Implement linear SVM method using scikit library Use the same dataset above Use train\_test\_split to create training and testing part.**

In this program, I have used the same dataset used above and the steps I have taken to import libraries and read the dataset are mentioned above in question 1. I have then imported two modules from the scikit-learn library. SVC: This module implements the Support Vector Classification algorithm.

LinearSVC: This module implements the Linear Support Vector Classification algorithm, which is a variation of the SVC algorithm that uses a linear decision boundary instead of a non-linear boundary.

svc.fit(X\_train, y\_train): This line trains the model on the training data (X\_train and y\_train).

y\_pred = svc.predict(X\_test): This line uses the predict method to generate predictions for the test data (X\_test). The predictions are stored in the variable 'y\_pred'. I have then calculated the accuracy of the model using the accuracy\_score function. Upon calculating % and rounding off to 2 digits the SVM accuracy we get is 62.61 and the generated classification report is as shown below:

Text

Description automatically generated with medium confidence

Text

Description automatically generated

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

Linear SVM has a better accuracy score than Gaussian Naive Bayes with 62.61% compared to 53.49%. The reason for this could be because Linear SVM is better suited for complex and non-linearly separable datasets. The SVM algorithm tries to find the maximum margin between the data points and the decision boundary, which makes it robust to outliers and non-linearities. Additionally, SVM allows for more flexibility in terms of choosing different kernel functions to tackle different types of data distributions. Gaussian Naive Bayes, on the other hand, assumes independence between the features and is not as robust as SVM for complex datasets.

The dataset size and structure, if the data has many features and is easy to interpret, Naive Bayes can be more effective.

The data distribution, Naive Bayes assumes that the features are independent, if the features in the data are not independent, the performance of Naive Bayes can decrease.

The accuracy of the model can also depend on the choice of hyperparameters, if the hyperparameters of the Naïve Bayes algorithm were not optimized, this could result in a lower accuracy score.

It is important to keep in mind that accuracy is not the only evaluation metric that should be considered, other metrics such as precision, recall, and F1-score should also be considered to get a comprehensive evaluation of the model's performance.