

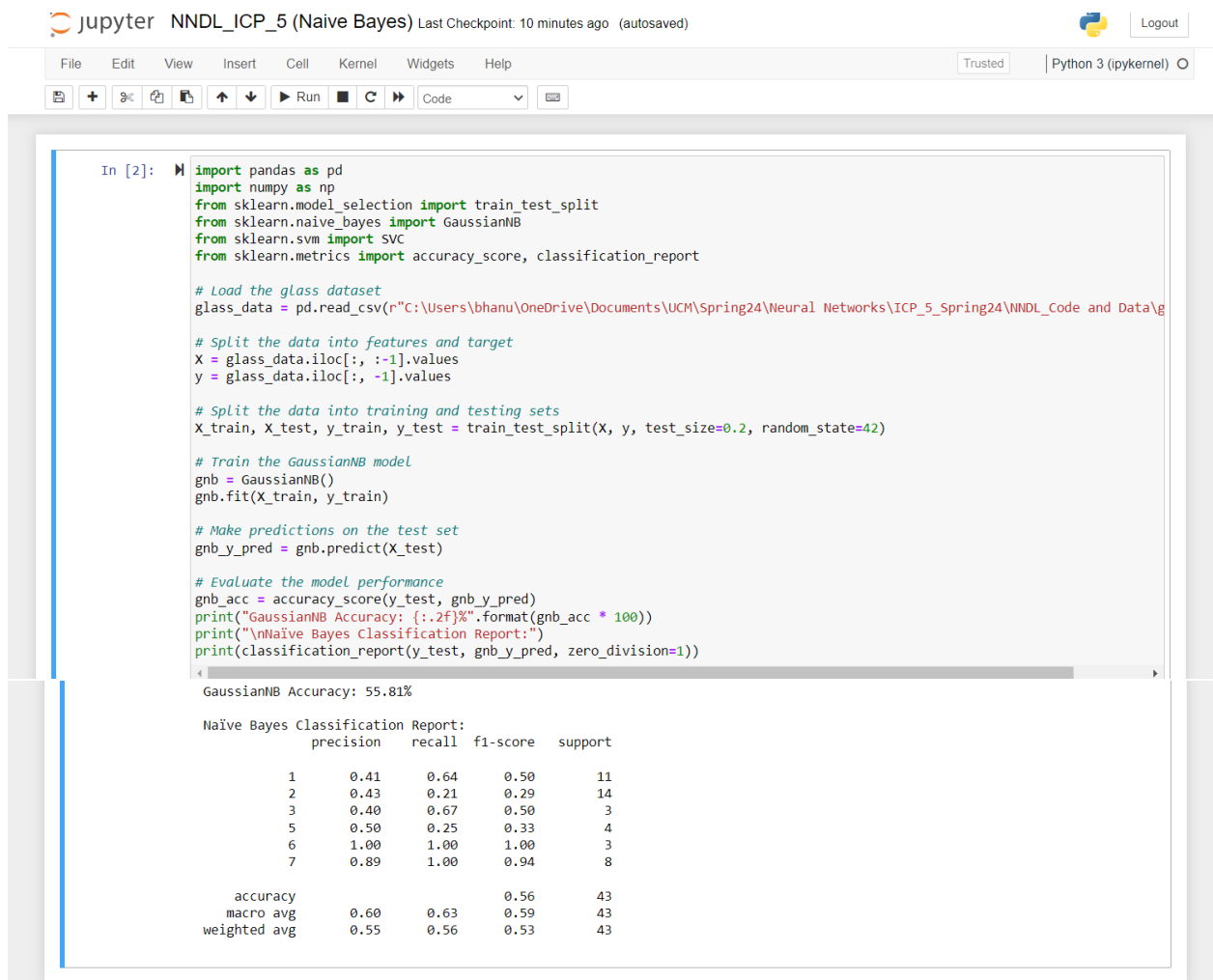
# Spring 2024: CS5720 Neural Networks & Deep Learning - ICP-5

## Bhanu Chandrika Lakkimsetti (700747439)

GitHub Link: [https://github.com/bhanuchandrika99/NNDL\\_ICP\\_5](https://github.com/bhanuchandrika99/NNDL_ICP_5)

### 1. Naïve Bayes

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 [2]: 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(r"C:\Users\bhanu\OneDrive\Documents\UCM\Spring24\Neural Networks\ICP_5_Spring24\NNDL_Code and Data\g

# 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("\nNaive Bayes Classification Report:")
print(classification_report(y_test, gnb_y_pred, zero_division=1))
```

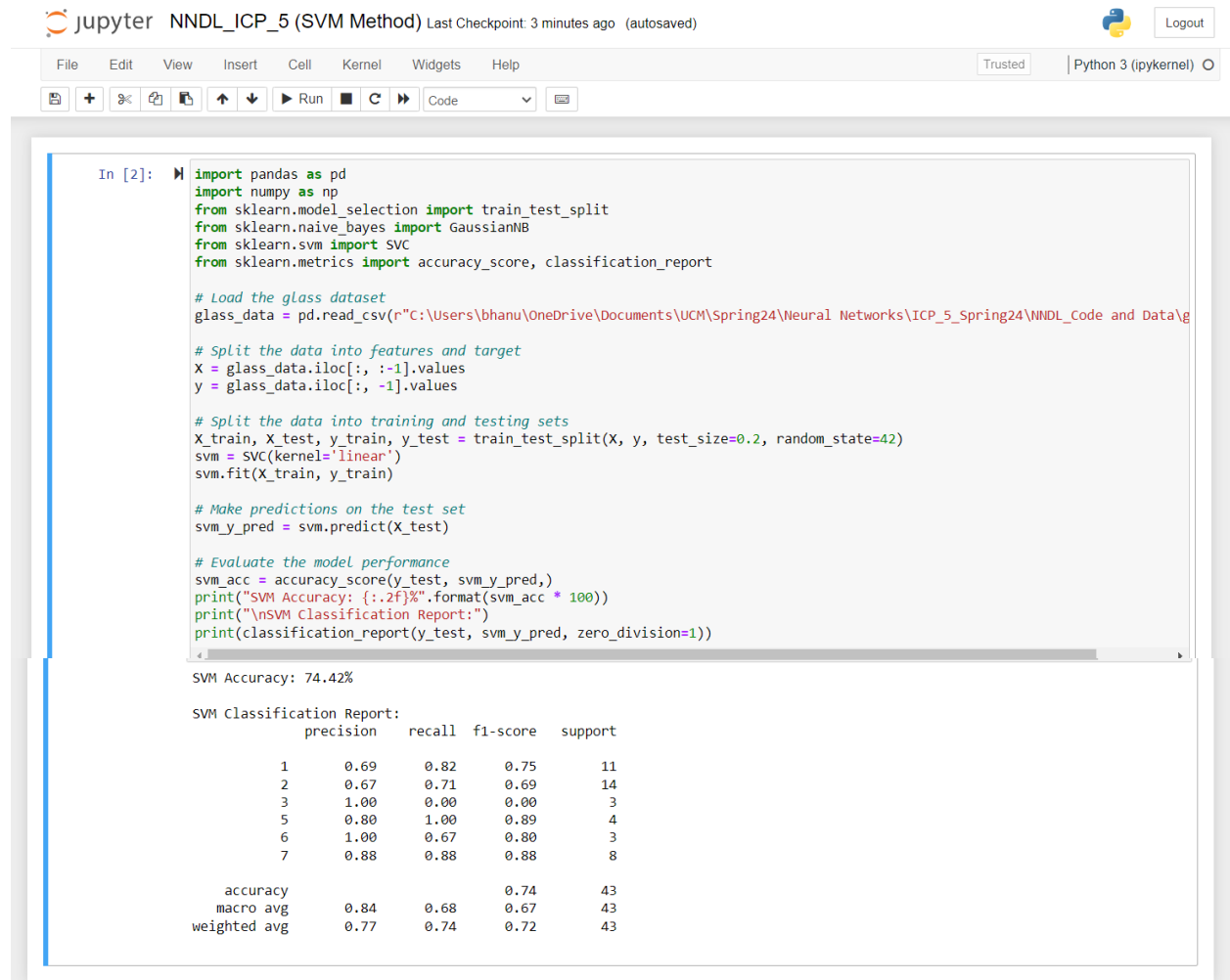
GaussianNB Accuracy: 55.81%

Naive 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. SVM Method

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)`



The image shows a Jupyter Notebook titled "NNDL\_ICP\_5 (SVM Method)". The code in the notebook implements a linear SVM model using the scikit-learn library. It loads the 'glass' dataset, splits it into training and testing sets, trains an SVM model, and evaluates its performance using accuracy and a classification report.

```
In [2]: 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(r"C:\Users\bhanu\OneDrive\Documents\UCM\Spring24\Neural Networks\ICP_5_Spring24\NNDL_Code and Data\g

# 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

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

Based on the data, the GaussianNB accuracy is 55.81% and the SVM technique yielded an accuracy of 74.42%. This indicates that the SVM Algorithm performs well in this scenario with the same test size and random state.

The following are some supporting factors for these:

Based on the Bayes theorem, GNB is a straightforward probabilistic classifier that is quick to learn and simple to understand. It assumes that the characteristics are conditionally independent given the class, i.e., it makes strong independence assumptions between the features. When words or n-grams represent the characteristics in a text classification problem, this method is most frequently employed. GNB is a suitable solution when you have a large number of features, as it scales well to high-dimensional data.

SVM, on the other hand, is a more advanced technique that determines the ideal decision border between classes by utilizing the idea of margins. SVM is frequently employed for image classification or classification tasks with a limited number of data because it can handle intricate non-linear correlations between features. SVM may be a preferable option when working with high-dimensional data because it is also more resistant to overfitting than GNB when the number of features is high.

For classification tasks, GNB is generally a reasonable first choice, particularly when the problem is well understood, the data is huge, and computational resources are restricted. SVM is a more potent technique that works well for more difficult classification issues, particularly where there is a requirement for greater robustness against overfitting, the problem is poorly understood, or the data is small.

In conclusion, the model can only be trained with a very small amount of data—just 215 rows—and several parameters, which complicate the prediction. Because of the minimal amount of data and the previously stated factors, the SVM algorithm is therefore the most appropriate for this set of data.