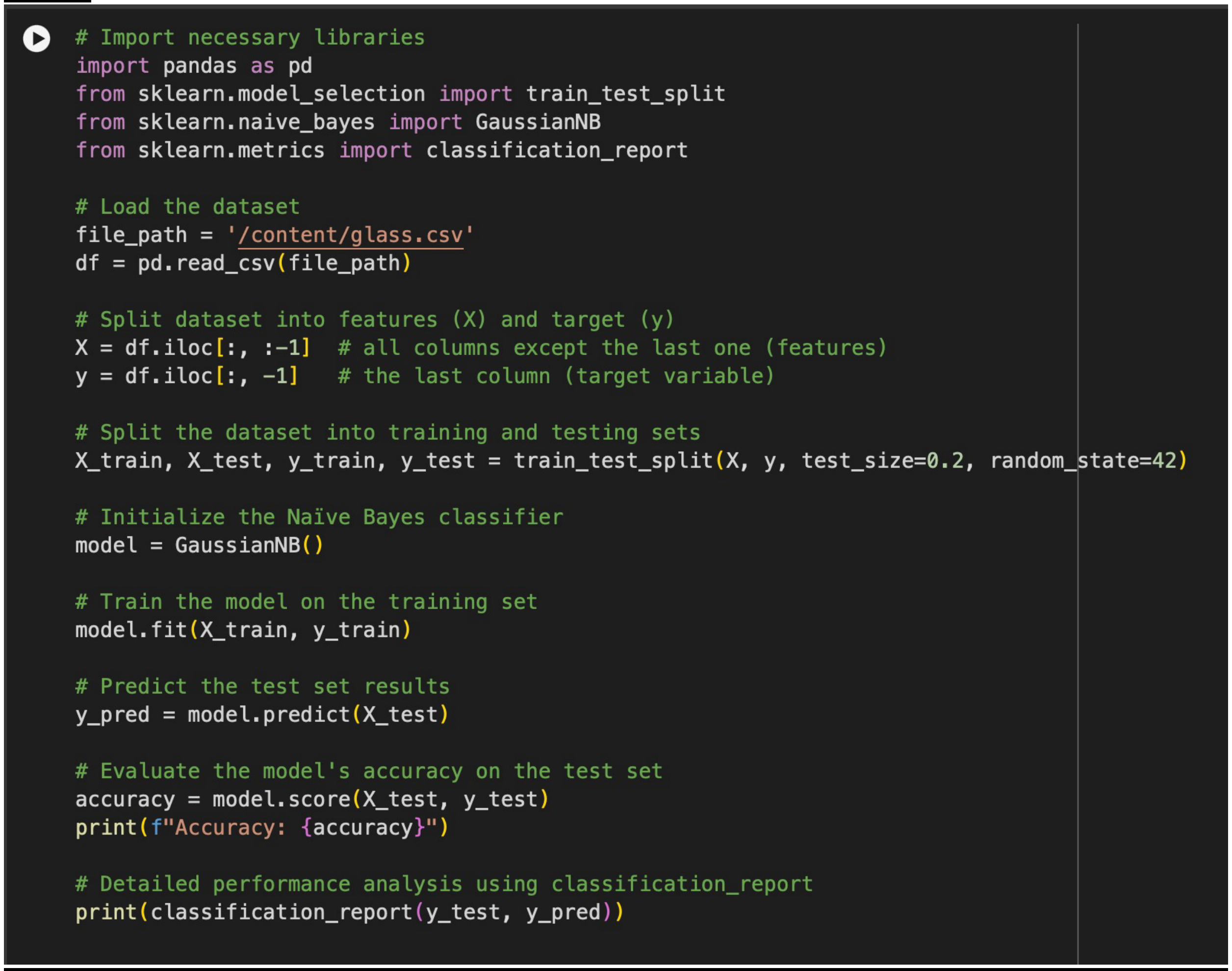
ICP -5

GitHub Link: https://github.com/SurajGamini18/Neural-Networks-Deep-Learning-Assignments

Video Link:
https://drive.google.com/file/d/1KCXUxN9jdGee0h46VdZMz44PfHpcT4th/view?usp=sharing

Venkata Suraj, Gamini
700744962

Ques=on1:
Code:

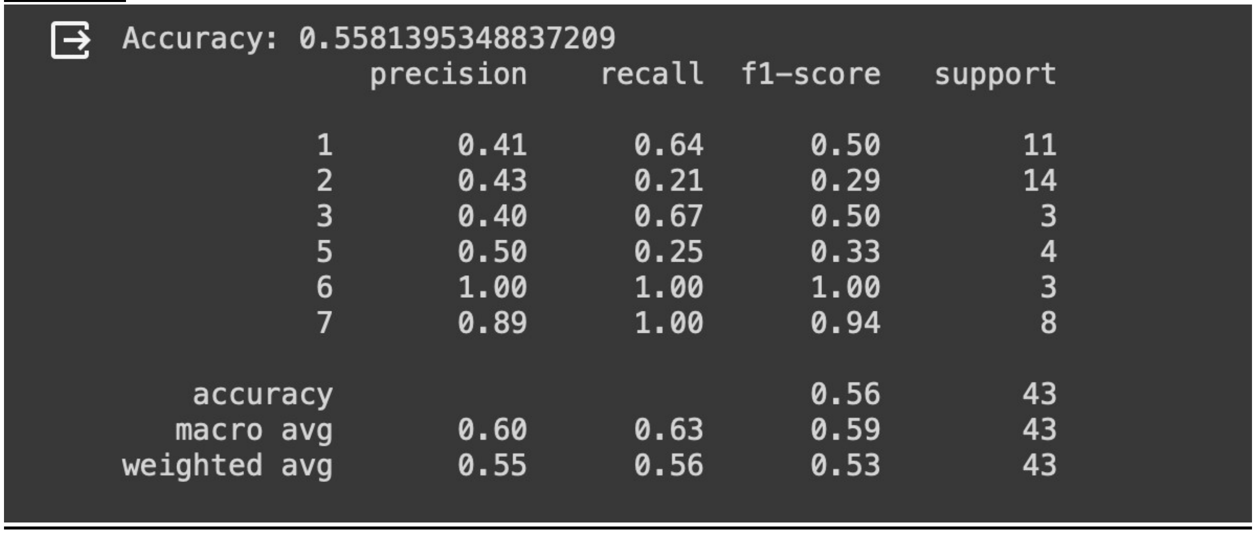


```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

# Load the dataset
file_path = '/content/glass.csv'
df = pd.read_csv(file_path)

# Split dataset into features (X) and target (y)
X = df.iloc[:, :-1]  # all columns except the last one (features)
y = df.iloc[:, -1]   # the last column (target variable)

# Split the dataset 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)

# Initialize the Naïve Bayes classifier
model = GaussianNB()

# Train the model on the training set
model.fit(X_train, y_train)

# Predict the test set results
y_pred = model.predict(X_test)

# Evaluate the model's accuracy on the test set
accuracy = model.score(X_test, y_test)
print(f"Accuracy: {accuracy}")

# Detailed performance analysis using classification_report
print(classification_report(y_test, y_pred))
```

Explana=on:

1. **Imports and Data Loading:** Imports libraries, loads the dataset from a CSV file into a DataFrame.
2. **Data Prepara=on:** Splits the dataset into features (`X`) and target (`y`), then divides it into training and tesDng sets.
3. **Model Training:** IniDalizes a `GaussianNB` model, trains it with the training data.
4. **Evalua=on:** Predicts on the test set, calculates accuracy, and generates a classificaDon report for detailed performance metrics.

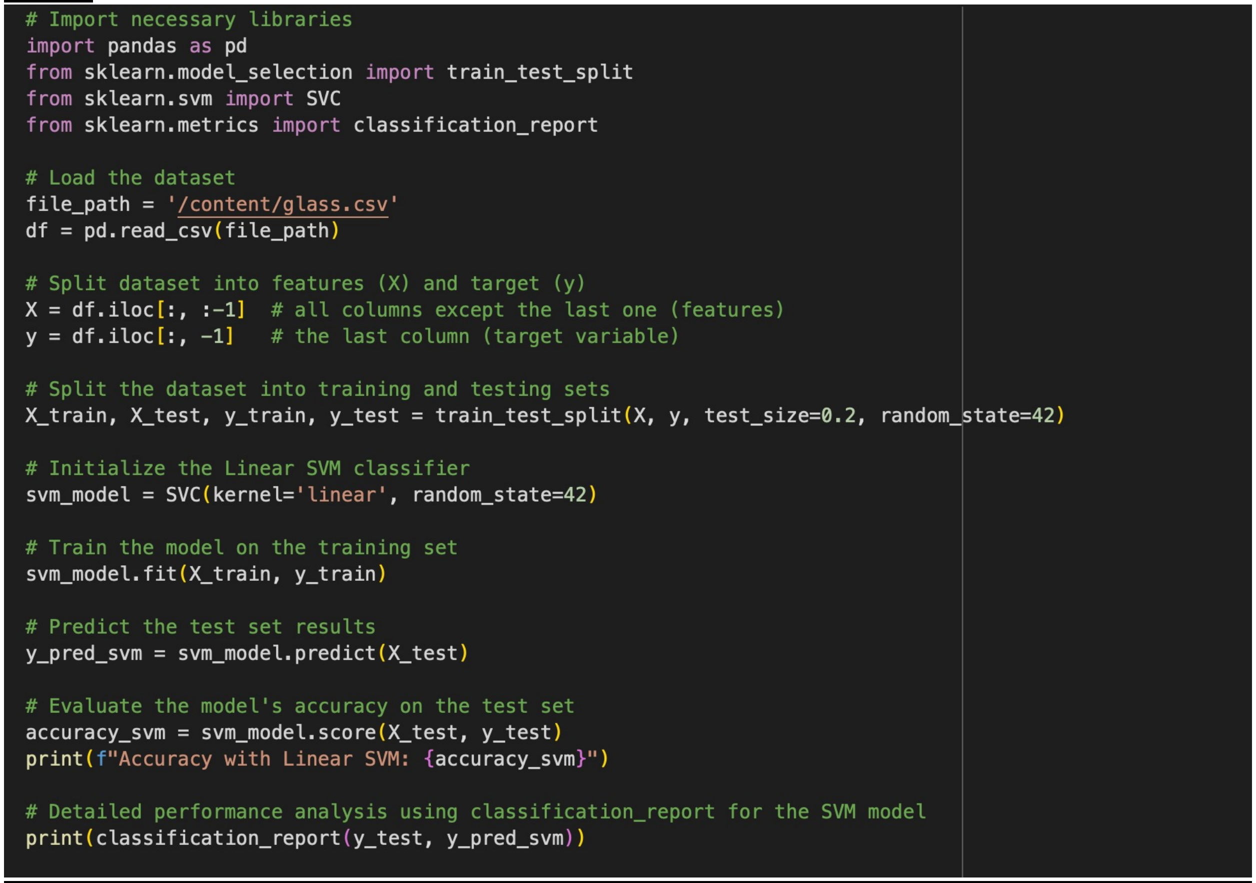
Output:



```
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
```

Ques=on2:

Code:



```python
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report

# Load the dataset
file_path = '/content/glass.csv'
df = pd.read_csv(file_path)

# Split dataset into features (X) and target (y)
X = df.iloc[:, :-1]  # all columns except the last one (features)
y = df.iloc[:, -1]   # the last column (target variable)

# Split the dataset 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)

# Initialize the Linear SVM classifier
svm_model = SVC(kernel='linear', random_state=42)

# Train the model on the training set
svm_model.fit(X_train, y_train)

# Predict the test set results
y_pred_svm = svm_model.predict(X_test)

# Evaluate the model's accuracy on the test set
accuracy_svm = svm_model.score(X_test, y_test)
print(f"Accuracy with Linear SVM: {accuracy_svm}")

# Detailed performance analysis using classification_report for the SVM model
print(classification_report(y_test, y_pred_svm))
```

Explana=on:

1. **Library Imports**: Brings in required libraries for data handling (`pandas`), dataset spliPng (`train_test_split`), modeling (`SVC` for SVM), and evaluaDon (`classificaDon_report`).

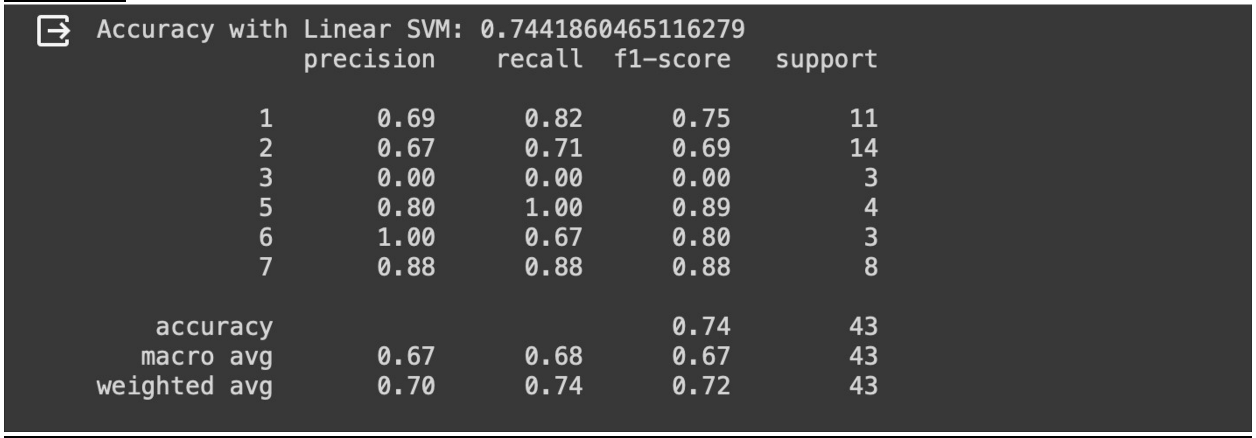
2. **Dataset Loading**: Reads a CSV file into a DataFrame for analysis.

3. **Data SpliPng**: Divides data into input features and target variable, then further splits these into training and tesDng sets for model validaDon.

4. **SVM Training**: Trains a Linear SVM model on the training data.

5. **Model Evalua=on:** Predicts the test data, assesses accuracy, and provides detailed metrics like precision, recall, and F1-score for a comprehensive understanding of model performance.

Output:



```
Accuracy with Linear SVM: 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
```

Comparison:

Comparing the results of the Linear SVM model with the Naïve Bayes model:

The Linear **SVM model shows a significantly higher accuracy (74.42%)** compared to the Naïve Bayes model (approximately 55.81%).

The improvement in accuracy can be a]ributed to several factors:
Linear SVM is parDcularly effecDve for high-dimensional data (like the glass dataset) because it works by finding the hyperplane that best separates the data into classes, which can be more effecDve for complex decision boundaries than the assumpDon of feature independence used by Naïve Bayes.

SVMs are also more flexible in handling non-linear data using kernel tricks, although we used a linear kernel here, it sDll benefits from the robust opDmizaDon and margin maximizaDon principles inherent to SVMs.

Naïve Bayes assumes that all features are independent of each other, which might not be the case in real-world datasets like glass classificaDon, where chemical components might have relaDonships affecDng the glass type.