NEURAL NETWORKS

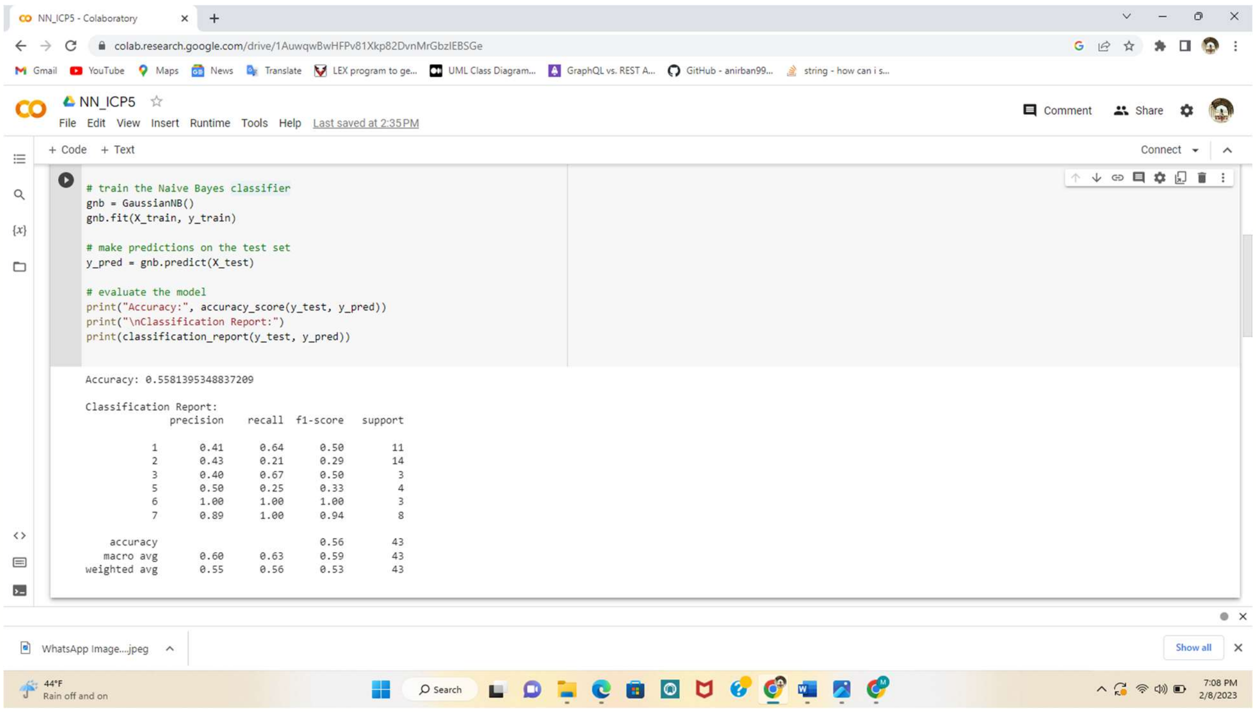
NAME : PRAVEENA GOLI
ID : 700743010

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)

```
 import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score

# load the glass dataset
glass = pd.read_csv("/content/glass.csv")

# split the data into training and testing sets
X = glass.drop("Type", axis=1)
y = glass["Type"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
andom_state=42)

# train the Naive Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# make predictions on the test set
y_pred = gnb.predict(X_test)

# evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Here, in this first question, I have implemented naïve bayes method using scikit learn library using the glass dataset provided. Used train_test_split to create training and testing part. By using Naïve bayes classifier , and by using 0.2 as test_size. Then I evaluated the part On test part using score. I got 55% accuracy.



```
# train the Naive Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# make predictions on the test set
y_pred = gnb.predict(X_test)

# evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

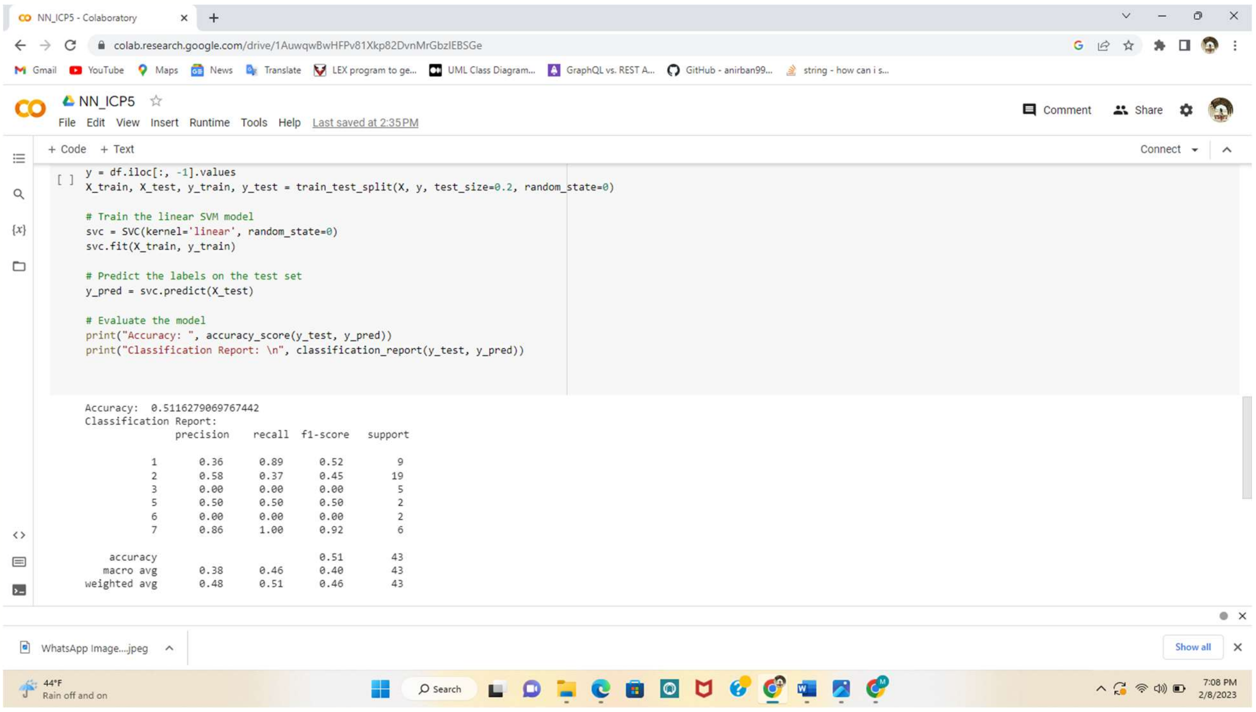
```
Accuracy: 0.5581395348837209

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. 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)

```
  # 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)
# Which algorithm you got better accuracy? Can you justify why?

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score

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

# Split the dataset into training and testing parts
X = df.iloc[:, :-1].values
```

```python
y = df.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
andom_state=0)

# Train the linear SVM model
svc = SVC(kernel='linear', random_state=0)
svc.fit(X_train, y_train)

# Predict the labels on the test set
y_pred = svc.predict(X_test)

# Evaluate the model
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Classification Report: \n", classification_report(y_test, y_pred))
```



```python
y = df.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Train the linear SVM model
svc = SVC(kernel='linear', random_state=0)
svc.fit(X_train, y_train)

# Predict the labels on the test set
y_pred = svc.predict(X_test)

# Evaluate the model
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Classification Report: \n", classification_report(y_test, y_pred))
```

```
Accuracy:  0.5116279069767442
Classification Report:
               precision    recall  f1-score   support

           1       0.36      0.89      0.52         9
           2       0.58      0.37      0.45        19
           3       0.00      0.00      0.00         5
           5       0.50      0.50      0.50         2
           6       0.00      0.00      0.00         2
           7       0.86      1.00      0.92         6

    accuracy                           0.51        43
   macro avg       0.38      0.46      0.40        43
weighted avg       0.48      0.51      0.46        43
```

Here, in this Second question, I have implemented SVM method using scikit learn library using the glass dataset provided. Used train_test_split to create training and testing part.
By using Naïve bayes classifier , and by using 0.2 as test_size. Then I evaluated the part On test part using score. I got 51% accuracy.

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

Here I got better accuracy with NB than SVM, as we tested the model with score parameter and NB is very fast and SVM works better with non linear data and multi dimensional.

But we can't say which classifier works better than which classifier comparatively. As it depends on the data we took and the parameter we taken for the testing part, But in here used same data and same parameter still NB got higher accuracy than SVM. But may be due to the linearity of the data.

REPO LINK : https://github.com/Goli18/NN_ICP5.git

VIDEO LINK: https://files.fm/f/x29t8drtj