

Spring 2024: CS5720

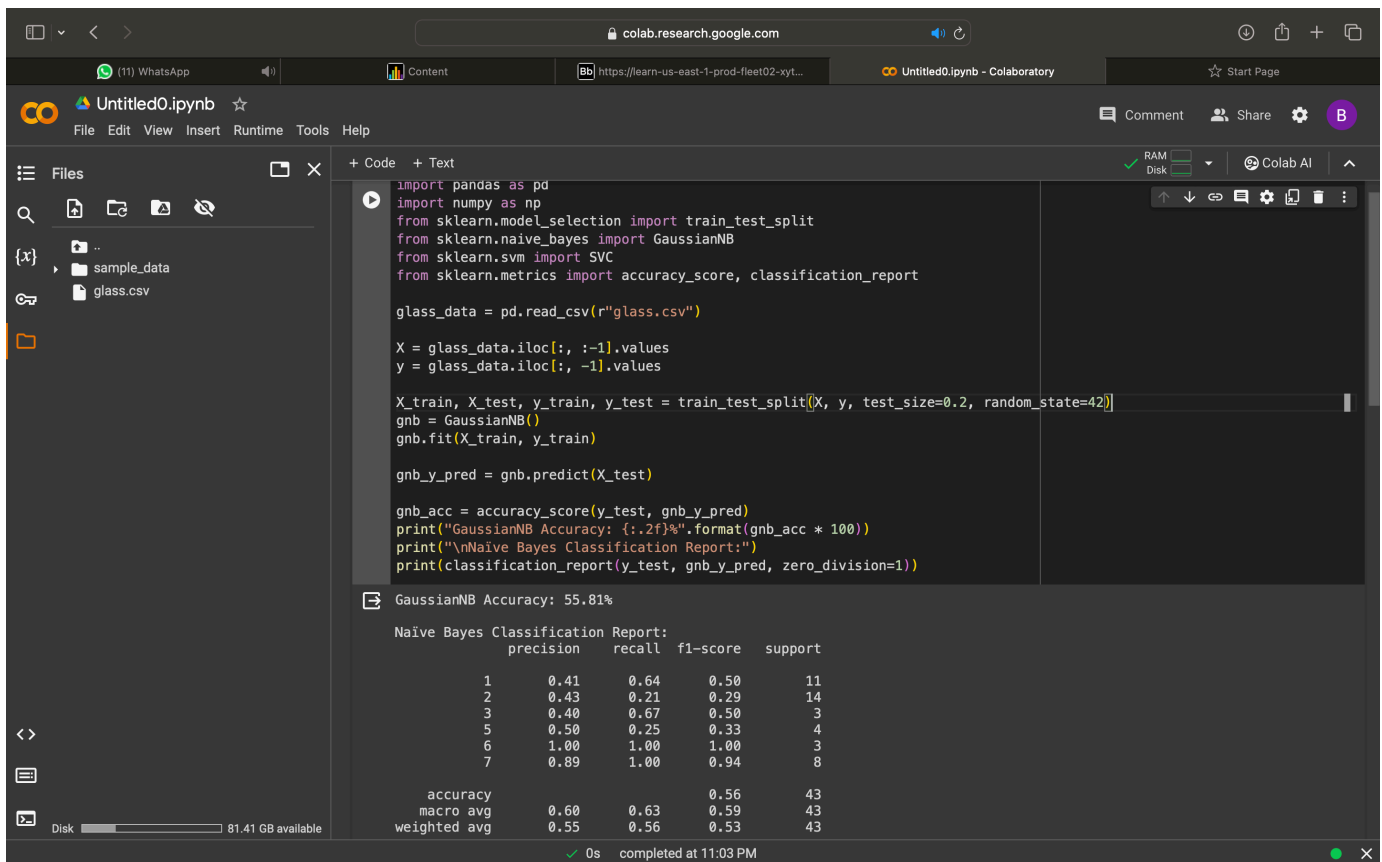
Neural Networks and Deep Learning - ICP-5

Bhavana Billa (700756590)

Github Link: <https://github.com/BillaBhavana7/neuralN>

1. Naive Bayes

Implement Naive 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, ypred)



```
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

glass_data = pd.read_csv(r"glass.csv")

X = glass_data.iloc[:, :-1].values
y = glass_data.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

gnb = GaussianNB()
gnb.fit(X_train, y_train)

gnb_y_pred = gnb.predict(X_test)

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))
```

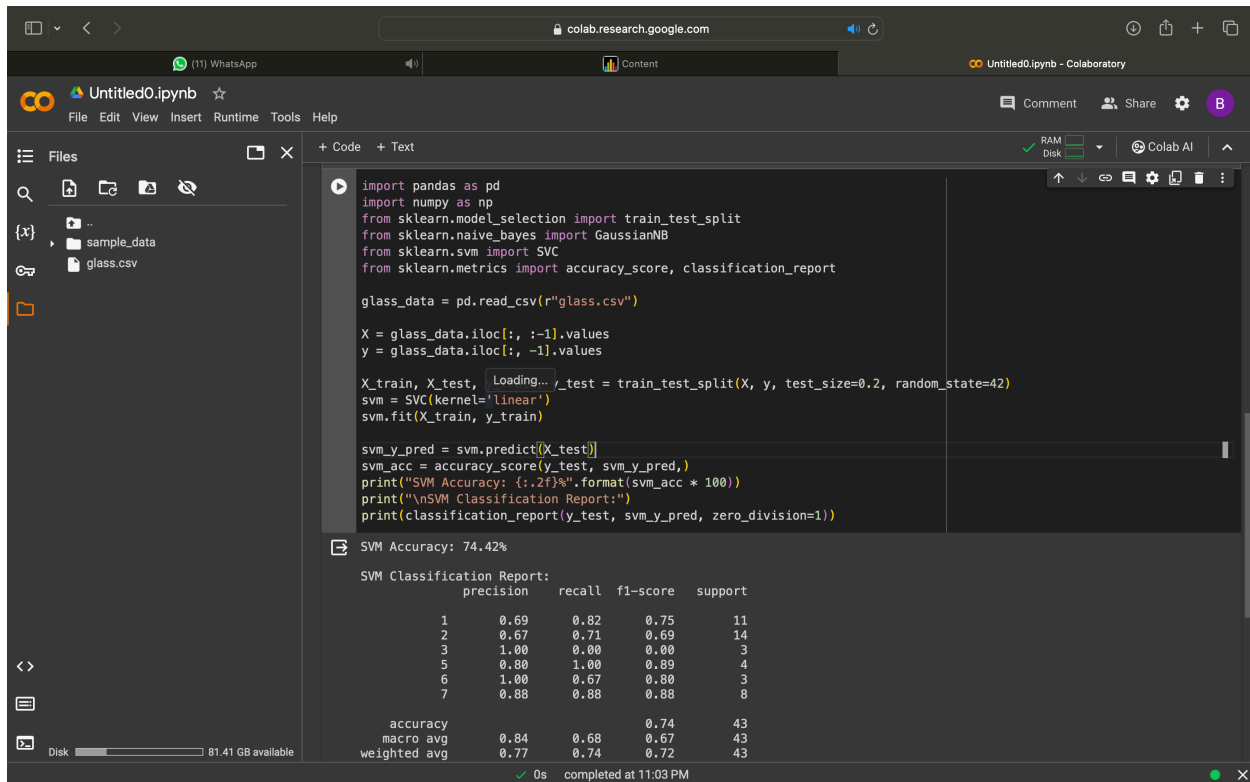
GaussianNB Accuracy: 55.81%

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

0s completed at 11:03 PM

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 screenshot shows a Google Colab notebook with the following code and output:

```
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

glass_data = pd.read_csv(r"glass.csv")

X = glass_data.iloc[:, :-1].values
y = glass_data.iloc[:, -1].values

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)

svm_y_pred = svm.predict(X_test)
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	0.88	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

0s completed at 11:03 PM

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

The findings indicate that the SVM approach produced an accuracy of 74.42% and the GaussianNB accuracy is 55.81%. This suggests that, given the same test size and random state, the SVM Algorithm performs admirably in this situation.

The following are some supporting factors for these:

GNB is a basic probabilistic classifier that is easy to learn and comprehend, and it is based on the Bayes theorem. It applies strong independence assumptions between the features, i.e., it assumes that the characteristics are conditionally independent given the class. This approach is most commonly used when the characteristics in a text classification problem are represented by words or n-grams. Given that GNB handles high-dimensional data well, it's a good option if you have a lot of features.

SVM, on the other hand, is a more sophisticated method that makes use of the concept of margins to identify the optimal decision border between classes. Because SVM can handle complex non-linear relationships between features, it is often used for image classification or classification problems with little data. SVM is also more resistant to overfitting than GNB when the number of features is high, so it might be a better choice when working with high-dimensional data.

To sum up, the model can only be trained using a relatively small set of data (215 rows) and multiple parameters, which makes prediction more difficult. The SVM algorithm is thus the most suitable for this set of data due to the little amount of data and the previously mentioned characteristics.