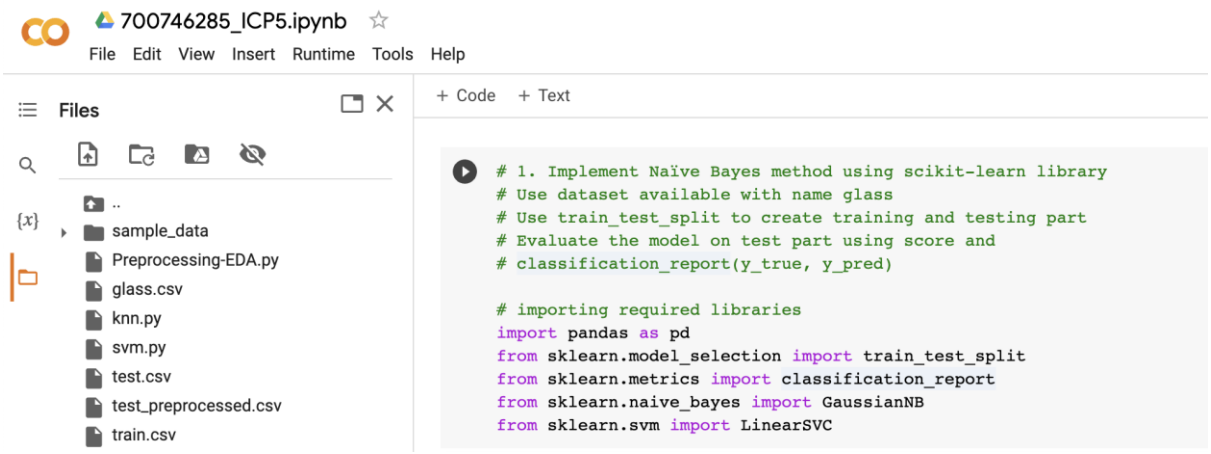
Neural Networks and Deep Learning
ICP5

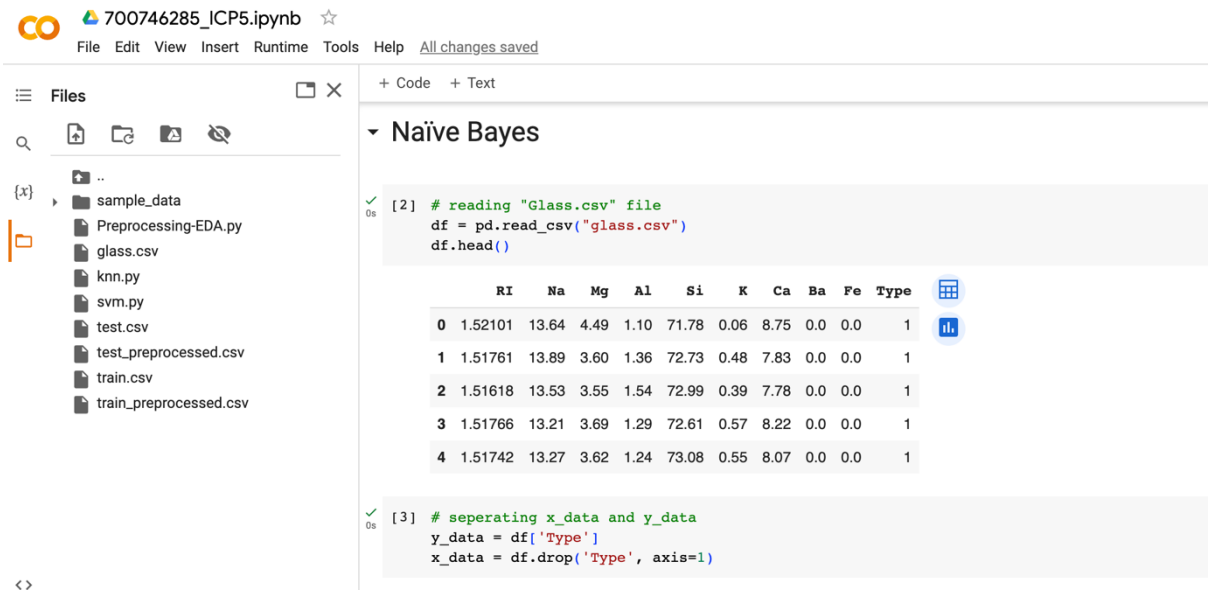
Student Name: Sravani Lankala
Student Id: 700746285

GitHub Link: https://github.com/sravanilankala/NNDL_ICP5_Fall2023

Video Link: https://drive.google.com/file/d/1UW-dNexKj--Qdk9EclOxFCbrYg0K1YbQ/view?usp=sharing

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)





700746285_ICP5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment

Files

+ Code + Text

```
..
sample_data
Preprocessing-EDA.py
glass.csv
knn.py
svm.py
test.csv
test_preprocessed.csv
train.csv
train_preprocessed.csv
```

```
# x_data
x_data.head()
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0

```
[5] # splitting the data into train and test sets
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=7)
```

```
[6] # train data shape
print(x_train.shape, y_train.shape)
```

```
(149, 9) (149,)
```

700746285_ICP5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

+ Code + Text

..
sample_data
Preprocessing-EDA.py
glass.csv
knn.py
svm.py
test.csv
test_preprocessed.csv
train.csv
train_preprocessed.csv

```
# test data shape
print(x_test.shape, y_test.shape)
```

```
(65, 9) (65,)
```

```
[8] # training Naive Bayes Model
nb_model = GaussianNB()
nb_model.fit(x_train, y_train)
```

```
GaussianNB()
```

```
[9] # predicting the x_test data using Naive Bayes Model
y_pred = nb_model.predict(x_test)
print(y_pred)
```

```
[3 3 3 3 6 3 2 3 3 3 3 2 3 3 3 1 1 2 3 6 3 2 3 7 3 7 7 1 1 3 7 2 3 5 2 7 3
 3 3 3 3 7 5 3 3 7 1 2 3 3 3 3 3 3 2 2 1 3 2 3 3 3 3 7 3]
```

```
[10] # Naive Bayes Model score
print(nb_model.score(x_test, y_test))
```

```
0.24615384615384617
```

700746285_ICP5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

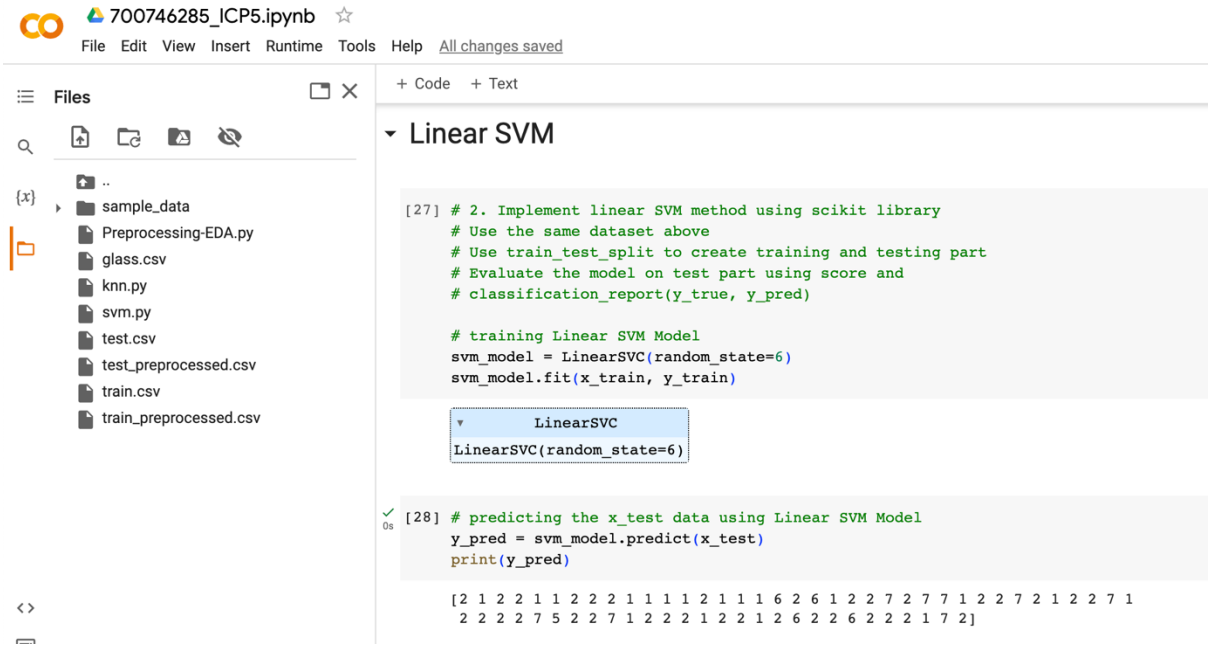
+ Code + Text

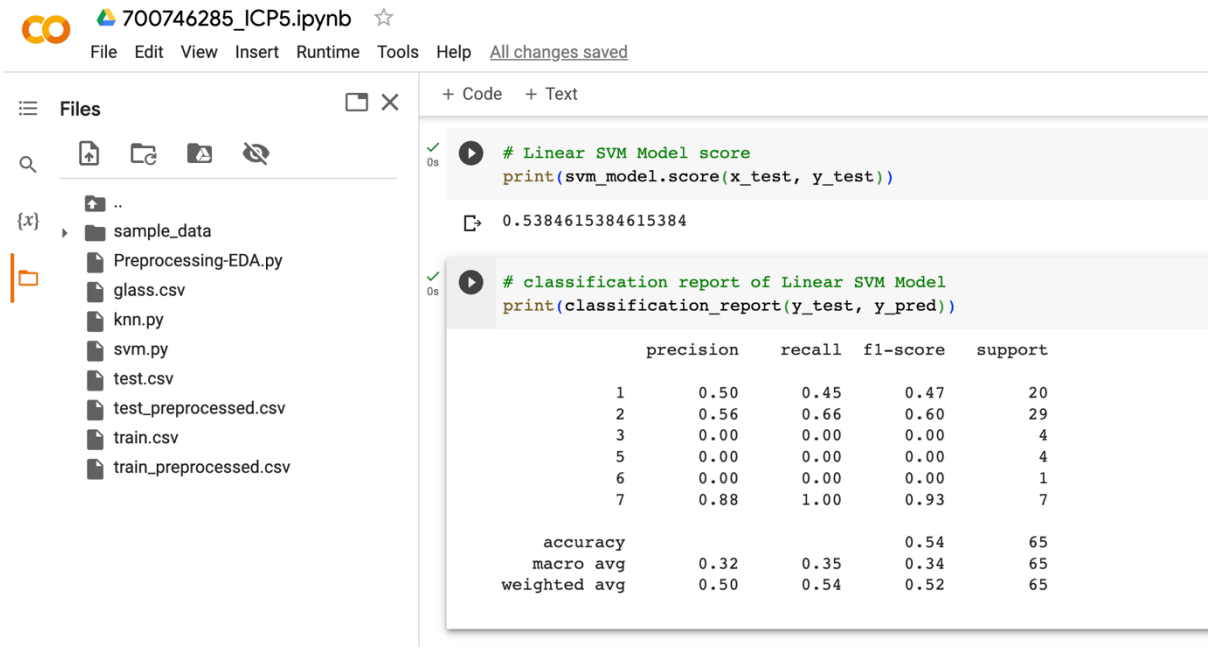
..
sample_data
Preprocessing-EDA.py
glass.csv
knn.py
svm.py
test.csv
test_preprocessed.csv
train.csv
train_preprocessed.csv

```
[11] # classification report of Naive Bayes Model
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
1	0.33	0.10	0.15	20
2	0.60	0.21	0.31	29
3	0.03	0.25	0.05	4
5	0.00	0.00	0.00	4
6	0.00	0.00	0.00	1
7	0.88	1.00	0.93	7
accuracy			0.25	65
macro avg	0.31	0.26	0.24	65
weighted avg	0.47	0.25	0.29	65

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?

Linear SVM has better accuracy than Naive Bayes Model because SVM can perform well in classifying multi-dimensional data and since Naive Bayes is based upon the frequency of occurrence it was not able to classify data.