

NNDL_ICP5

STUDENT NAME: NEERAJ KUMAR BARIGELA

STUDENT ID: 700760341

GitHub Link: https://github.com/neeraj4944/Fall2023_NNDL_ICP5

Video Link: <https://drive.google.com/file/d/1xehifu4zOb83ytC3s4EOB-jRHgqLcGq3/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)

The screenshot shows a Jupyter Notebook titled "700760341_NN_ICP5.ipynb". The left sidebar displays a file explorer with a folder named "sample_data" containing several files: "Preprocessing-EDA.py", "glass.csv", "knn.py", "svm.py", "test.csv", "test_preprocessed.csv", "train.csv", and "train_preprocessed.csv". The main area shows two code cells. The first cell, labeled [3], contains code to import necessary libraries and perform a Naïve Bayes classification. The second cell, labeled [9], loads the "glass.csv" dataset and displays its head. Below the code, a table shows the first five rows of the dataset.

```
[3] # 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 required python 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, accuracy_score

[9] # load the glass dataset
glass = pd.read_csv("/content/glass.csv")
glass.head()
```

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

Files

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

+ Code + Text

```
[10] # 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, random_state=42)
X_train.head()
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
79	1.51590	12.82	3.52	1.90	72.86	0.69	7.97	0.00	0.00
161	1.51934	13.64	3.54	0.75	72.65	0.16	8.89	0.15	0.24
109	1.51818	13.72	0.00	0.56	74.45	0.00	10.99	0.00	0.00
127	1.52081	13.78	2.28	1.43	71.99	0.49	9.85	0.00	0.17
95	1.51860	13.36	3.43	1.43	72.26	0.51	8.60	0.00	0.00

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

GaussianNB
GaussianNB()

```
[7] # make predictions on the test set
y_pred = gnb.predict(X_test)
```

Files

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

+ Code + Text

```
[8] # 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:

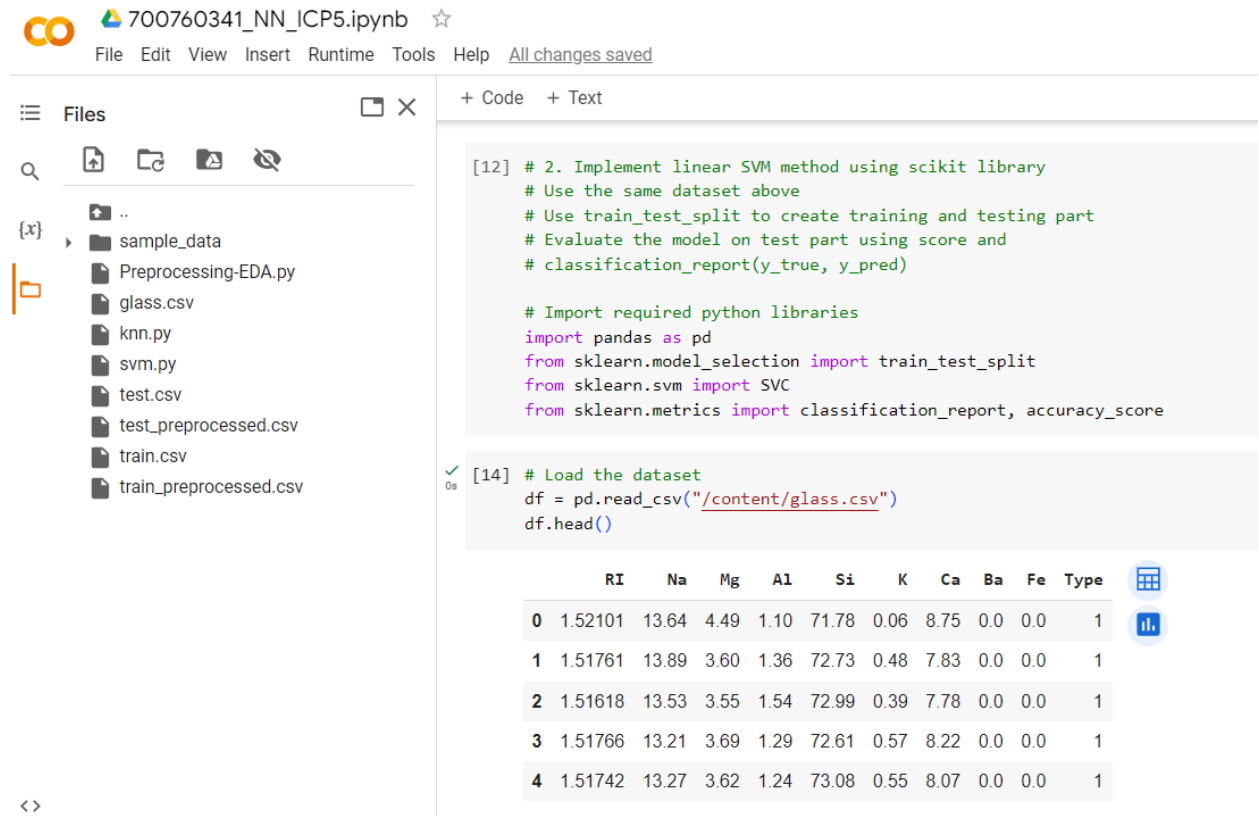
	precision	recall	f1-score	support
1	0.41	0.64	0.50	11
2	0.43	0.21	0.29	14
3	0.40	0.67	0.50	3
5	0.50	0.25	0.33	4
6	1.00	1.00	1.00	3
7	0.89	1.00	0.94	8
accuracy			0.56	43
macro avg	0.60	0.63	0.59	43
weighted avg	0.55	0.56	0.53	43

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)`



700760341_NN_ICP5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

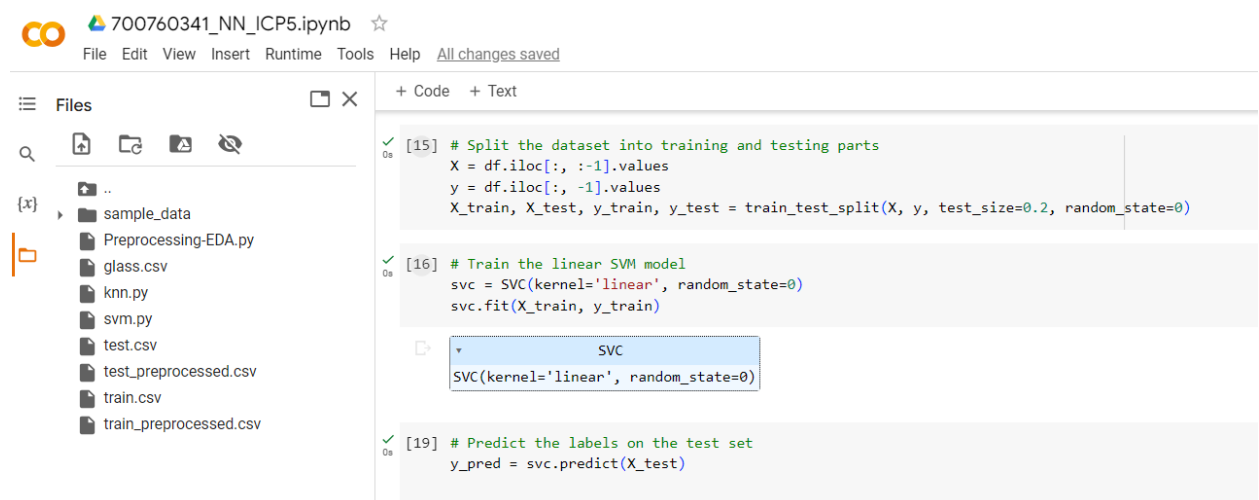
- sample_data
 - Preprocessing-EDA.py
 - glass.csv
 - knn.py
 - svm.py
 - test.csv
 - test_preprocessed.csv
 - train.csv
 - train_preprocessed.csv

```
[12] # 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)

      # Import required python libraries
      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

[14] # Load the dataset
      df = pd.read_csv("/content/glass.csv")
      df.head()
```

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



700760341_NN_ICP5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

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

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

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

[19] # Predict the labels on the test set
      y_pred = svc.predict(X_test)
```

The screenshot shows a Jupyter Notebook titled "700760341_NN_ICP5.ipynb". The left sidebar displays a file explorer with the following structure:

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

The main code cell contains the following Python code:

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

The output of the code is a classification report for an SVM model:

```
Accuracy: 0.5116279869767442
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         43
weighted avg          0.48         43
```

Below the report, there are several warning messages from sklearn:

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
_warn_prf(average, modifier, msg_start, len(result))
```

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

The screenshot shows the same Jupyter Notebook interface, but the code cell now contains a handwritten answer to the question:

```
[ ] # Which algorithm you got better accuracy? Can you justify why?
Here, I got better accuracy with Naive Bayes method than SVM, as model was tested with score parameter and Naive Bayes is very fast
SVM works better with non-linear data and multi dimensional data.

We can't say which classifier works better. As it depends on the data taken and the parameter taken for testing.
```