

# Neural Networks and Deep Learning

## ICP5

Student Name: Sravani Lankala  
Student Id: 700746285

GitHub Link: [https://github.com/sravanilankala/NNDL\\_ICP5\\_Fall2023](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)

The screenshot displays a Jupyter Notebook titled "700746285\_ICP5.ipynb". The interface includes a file explorer on the left, a code editor in the center, and a runtime output area on the right.

**File Explorer:** The file explorer shows the following files and folders:

- sample\_data
- Preprocessing-EDA.py
- glass.csv
- knn.py
- svm.py
- test.csv
- test\_preprocessed.csv
- train.csv

**Code Editor:** The code editor contains the following Python code:

```
# 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)

# importing required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import LinearSVC
```

**Runtime Output:** The runtime output shows the execution of the code, including the loading of the "glass.csv" file and the resulting DataFrame structure.

**Naïve Bayes**

[2] # reading "Glass.csv" file

```
df = pd.read_csv("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

[3] # separating x\_data and y\_data

```
y_data = df['Type']
x_data = df.drop('Type', axis=1)
```

700746285\_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

+ Code + Text

# 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

sample\_data

Preprocessing-EDA.py

glass.csv

knn.py

svm.py

test.csv

test\_preprocessed.csv

train.csv

train\_preprocessed.csv

+ Code + Text

# 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

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 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 2 2 1 3 2 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

sample\_data

Preprocessing-EDA.py

glass.csv

knn.py

svm.py

test.csv

test\_preprocessed.csv

train.csv

train\_preprocessed.csv

+ Code + Text

[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

- 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)`

```
[27] # 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)

# training Linear SVM Model
svm_model = LinearSVC(random_state=6)
svm_model.fit(x_train, y_train)

LinearSVC
LinearSVC(random_state=6)

[28] # predicting the x_test data using Linear SVM Model
y_pred = svm_model.predict(x_test)
print(y_pred)

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

```
# Linear SVM Model score
print(svm_model.score(x_test, y_test))

0.5384615384615384

# classification report of Linear SVM Model
print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

     1       0.50         0.45         0.47         20
     2       0.56         0.66         0.60         29
     3       0.00         0.00         0.00          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          0.54
 macro avg         0.32         0.35         0.34         65
 weighted avg      0.50         0.54         0.52         65
```

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.

