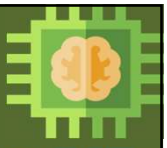
Elective Course
Course Code: CS4103
Autumn 2025-26



Lecture #47

Artificial Intelligence for Data Science

Week-13:

MACHINE LEARNING (Part XV)

Exploring Naïve Bayes (NB) Classifier using Python (Scikit-learn Library)

Course Instructor:

Dr. Monidipa Das
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata, India 741246

Naïve Bayes (NB) Classifier: Scikit-learn Library

BernoulliNB	Naive Bayes classifier for multivariate Bernoulli models.
CategoricalNB	Naive Bayes classifier for categorical features.
ComplementNB	The Complement Naive Bayes classifier
GaussianNB	Gaussian Naive Bayes (GaussianNB).
MultinomialNB	Naive Bayes classifier for multinomial models.

```
class sklearn.naive_bayes.BernoulliNB(*, alpha=1.0, force_alpha=True, binarize=0.0, fit_prior=True, class_prior=None)
```

```
class sklearn.naive_bayes.GaussianNB(*, priors=None, var_smoothing=1e-09)
```

```
class sklearn.naive_bayes.CategoricalNB(*, alpha=1.0, force_alpha=True, fit_prior=True, class_prior=None, min_categories=None)
```
[source]

```
class sklearn.naive_bayes.ComplementNB(*, alpha=1.0, force_alpha=True, fit_prior=True, class_prior=None, norm=False)
```
[source]

```
class sklearn.naive_bayes.MultinomialNB(*, alpha=1.0, force_alpha=True, fit_prior=True, class_prior=None)
```
[source]

Suitable Dataset: Examples

BernoulliNB

Email	Feature: "free"	Feature: "money"	Feature: "call"	Feature: "win"	Target (Spam: 1, Not-spam: 0)
e1	1	1	0	0	1
e2	0	0	0	1	0
e3	1	0	1	0	1

CategoricalNB

Sample	Feature: weather	Feature: day	Feature: time	Target (Traffic Jam?)
1	Clear	Workday	Lunch	No
2	Clear	Workday	Evening	Yes
6	Rainy	Workday	Morning	Yes

MultinomialNB

Email	Feature: "free"	Feature: "money"	Feature: "call"	Feature: "win"	Target (Spam: 1, Not-spam: 0)
e1	2	4	0	0	1
e2	0	0	0	3	0
e3	1	0	3	0	1

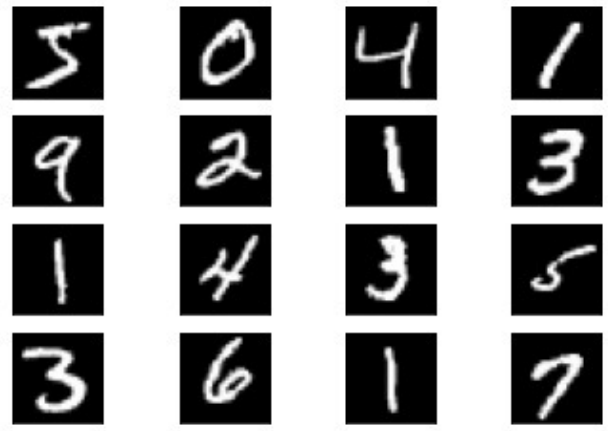
GaussianNB

Sample	Feature: sepal length	Feature: sepal width	Feature: petal length	Feature: petal width	Target
1	4.9	3	1.4	0.2	setosa
2	7	3.2	4.7	1.4	versicolor
3	6.4	3.2	6.1	2.5	virginica

NB for Handwritten-digit Classification: Dataset Loading

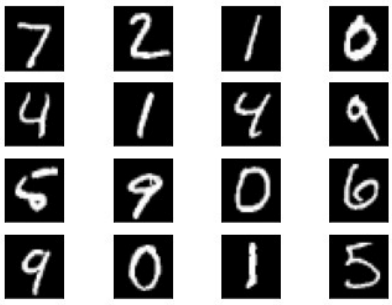

```
import matplotlib.pyplot as plt
import pickle
import numpy as np

with open("F:/CS4103/code/mnist.pkl","rb") as fh:
    train_set, validation_set, test_set = pickle.load(fh,encoding='latin1')

train_imgs,train_labels = train_set[0], train_set[1]
valid_imgs, valid_labels = validation_set[0], validation_set[1]
test_imgs, test_labels = test_set[0], test_set[1]


image_size = 28
no_of_different_labels = 10
image_pixels = image_size * image_size
```

NB for Handwritten-digit Classification: Data Visualization



```
fig, axes = plt.subplots(4, 4)

# Flatten the axes array for easy
iteration
axes = axes.flatten()

#Plot a few training sample images
for i, ax in enumerate(axes):
    ax.imshow(train_imgs[i].reshape(
        image_size, image_size),
        cmap=plt.cm.gray)
    ax.set_xticks(())
    ax.set_yticks(())
plt.show()
```


NB for Handwritten-digit Classification: Building Classifier

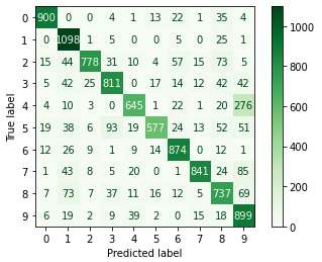


```
from sklearn.naive_bayes import GaussianNB

#created a object for the classifier
NB=GaussianNB(priors=np.ones(10)*.1, var_smoothing=0.1)

#train using train data
NB.fit(train_imgs,train_labels)

fig, axes = plt.subplots(4, 4)
#flatten the axes array for easy iteration
axes = axes.flatten()
#plotting a few test images
for i, ax in enumerate(axes):
    ax.imshow(test_imgs[i].reshape(image_size, image_size), cmap=plt.cm.gray)
    ax.set_xticks(())
    ax.set_yticks(())

#predict target data for test feature data and save into pred variable
predicted=NB.predict(test_imgs)
```

NB for Handwritten-digit Classification: Evaluation

```
from sklearn.metrics import
confusion_matrix,ConfusionMatrixDisplay,classificat
ion_report,accuracy_score
import numpy as np

print("Classification report:\n",
classification_report(test_labels, predicted))

cm_pred = confusion_matrix(test_labels, predicted)
plt.figure()
disp =
ConfusionMatrixDisplay(confusion_matrix=cm_pred,
display_labels=np.unique(test_labels))
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()

print("Accuracy={}".format(
accuracy_score(test_labels, predicted)))
```

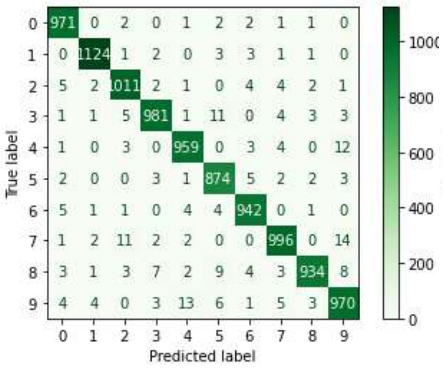
```
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.92      0.92       980
           1       0.79      0.97      0.87      1135
           2       0.93      0.75      0.83      1032
           3       0.81      0.80      0.81      1010
           4       0.86      0.66      0.74       982
           5       0.90      0.65      0.75       892
           6       0.85      0.91      0.88       958
           7       0.93      0.82      0.87      1028
           8       0.71      0.76      0.73       974
           9       0.63      0.89      0.74      1009

    accuracy                           0.82     10000
   macro avg       0.83      0.81      0.81     10000
weighted avg       0.83      0.82      0.82     10000
```



Accuracy=0.816

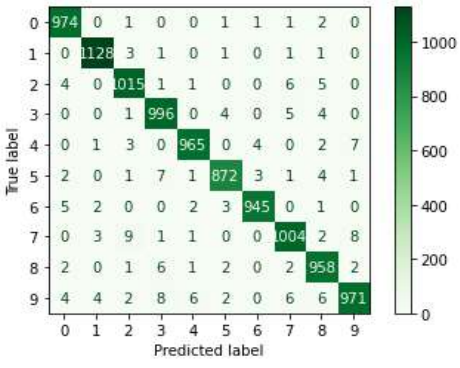
MNIST: Comparative Study

Neural Network (MLP)



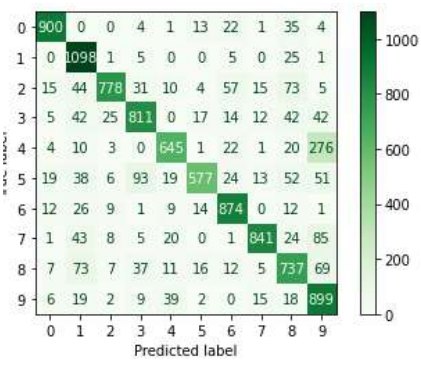
Accuracy=0.976200

Support Vector Machine



Accuracy=0.9828

Naïve Bayes



Accuracy=0.816

NB for Iris Classification: Dataset Loading


```
import pandas as pd

#Load the dataset
dataset =
pd.read_csv('F:/CS4103/code/iris.csv')

#View dataset
print("First 6 rows of the dataset:")
print(dataset.head(6))

#Dataset column names
print("\nName of the columns in the dataset:")
print(list(dataset.columns))

input_features = list(dataset.columns[:-1])
class_col=dataset.columns[-1]
print("Name of the input feature columns:",input_features)
print("Name of the class/label column:",class_col)
```

```
First 6 rows of the dataset:
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
5           5.4          3.9           1.7          0.4  Setosa
```

```
Name of the columns in the dataset:
['sepal.length', 'sepal.width',
'petal.length', 'petal.width', 'variety']
Name of the input feature columns:
['sepal.length', 'sepal.width',
'petal.length', 'petal.width']
Name of the class/label column: variety
```


NB for Iris Classification: Exploring Dataset Details


```
# Number of rows or instances and number of columns features
print("\nTotal number of data points/examples/instances:", dataset.shape[0])
print("Total number of input features:", dataset.shape[1]-1)

#Dataset description
print("\nDescription of the dataset:")
print(dataset.describe())

#Print number of classes
c=dataset[class_col].nunique()
print("\nNumber of classes(discrete labels):",c)

#Number of instances per class
print("\nSample count per class:")
print(dataset[class_col].value_counts())

#Check missing values in variables
dataset.isnull().sum()
```

```
Total number of data points/examples/instances: 150
Total number of input features: 4
```

```
Description of the dataset:
       sepal.length  sepal.width  petal.length  petal.width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

```
Number of classes (discrete labels): 3

Sample count per class:
Setosa        50
Versicolor    50
Virginica     50
Name: variety, dtype: int64
```

```
sepal.length    0
sepal.width     0
petal.length    0
petal.width     0
variety         0
dtype: int64
```

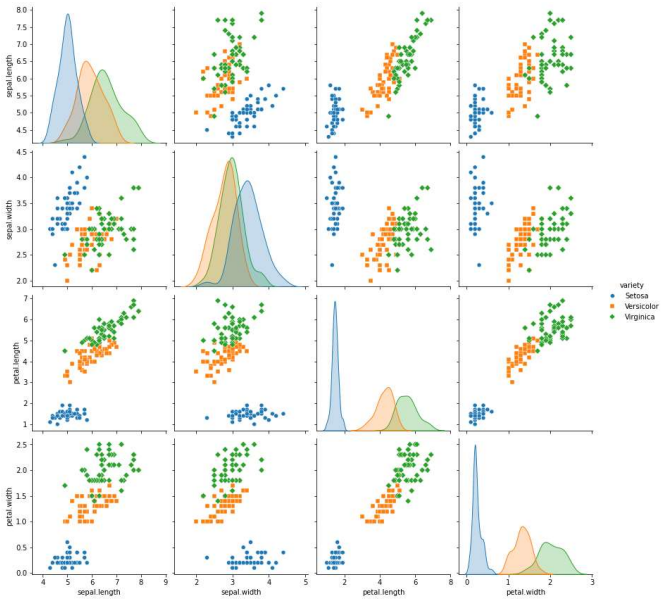
NB for Iris Classification: Data Split


```
X = dataset[input_features].values
y = dataset[class_col].values

#label encoder from sklearn for label encoding
from sklearn.preprocessing import LabelEncoder
y=LabelEncoder().fit_transform(y)

#split dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

#check the shape of X_train and X_test
print("Traning set shape:",X_train.shape,"\nTest set shape:",X_test.shape)
```

```
Traning set shape: (120, 4)
Test set shape: (30, 4)
```


NB for Iris Classification: Data Visualization


```
#visualize the distribution
of features and their
relationship with others

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure()
sns.pairplot(dataset, hue =
class_col, size=3, markers=["o",
"s", "D"])
plt.show()
```



NB for Iris Classification: Building Classifier and Evaluation

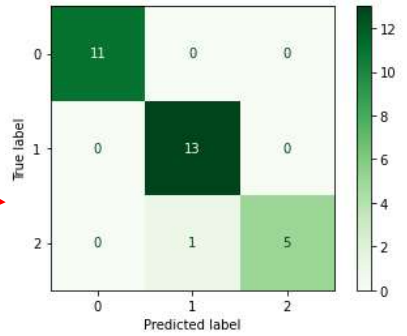

```
from sklearn.naive_bayes import GaussianNB

#created a object for the classifier
NB=GaussianNB()
#train usinge train data
NB.fit(X_train,y_train)

#predict target data for test feature data and save into pred variable
predicted=NB.predict(X_test)

from sklearn.metrics import
confusion_matrix,ConfusionMatrixDisplay,classification_report,accuracy_score
import numpy as np

print("Accuracy={}".format(accuracy_score(y_test, predicted)))


predicted_train=NB.predict(X_train)
print("Accuracy={}".format(accuracy_score(y_train, predicted_train)))
```

```
Test Accuracy=0.9667
Train Accuracy=0.9500
```


NB for Iris Classification: Evaluation


```
print("Classification report:\n",
classification_report(y_test, predicted))

cm_pred =
confusion_matrix(y_test, predicted)

plt.figure()
disp = ConfusionMatrixDisplay(
confusion_matrix=cm_pred,
display_labels=np.unique(y_test))
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()
```

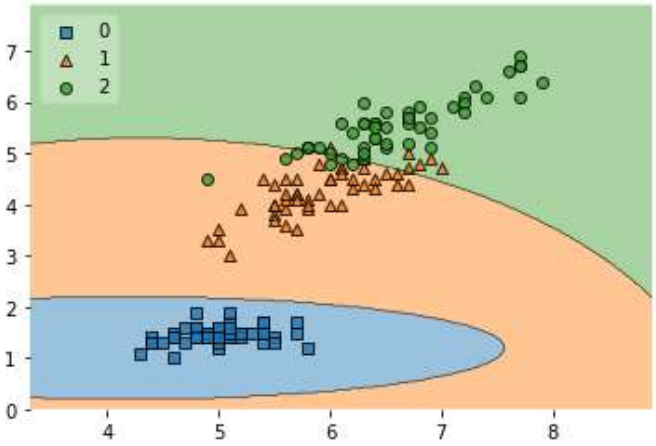
```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.93      1.00      0.96        13
           2       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30
```



NB for Iris Classification: Decision Boundary

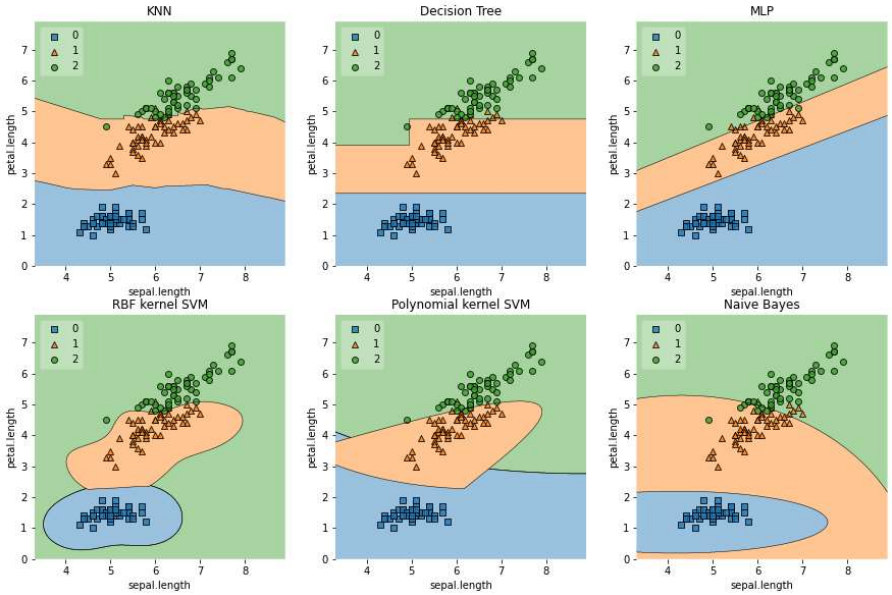
15

```
#created a object for the classifier
NB=GaussianNB()

#train usinge train data
NB.fit(X_train[:,[0,2]],y_train)

from mlxtend.plotting import
plot_decision_regions

fig = plot_decision_regions(
X=X[:,[0,2]], y=y, clf=NB, legend=2)

plt.show()
```



Iris: Comparative Study

16



Model	Test Accuracy
KNN (3)	96.67%
KNN (9)	100%
DT (4)	100%
MLP (10,5)	100%
SVM ('rbf')	100%
NB (default)	96.67%

Practice Problem

17

- Naïve Bayes Classifier

Prediction using *m-estimate*

Solution discussed during the lecture session

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

18

Questions?