Assignment no 6

Aim:

1. Concepts used in Naïve Bayes classifier
2. Naive Bayes Example
3. Confusion Matrix Evaluation Metric

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv("iris.csv");
df
```

Out[2]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

150 rows × 5 columns

In [3]:
```
df.head()
```

Out[3]:

	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

In [4]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, cor
from sklearn.preprocessing import LabelEncoder
```

In [5]:
```
# Initialize LabelEncoder
le = LabelEncoder()

# Convert 'variety' column (categorical) to numerical values
df['variety'] = le.fit_transform(df['variety'])

# Display the dataset with encoded labels
print(df.head())
```

```
   sepal.length  sepal.width  petal.length  petal.width  variety
0           5.1          3.5           1.4          0.2        0
1           4.9          3.0           1.4          0.2        0
2           4.7          3.2           1.3          0.2        0
3           4.6          3.1           1.5          0.2        0
4           5.0          3.6           1.4          0.2        0
```

In [6]:
```
df
```

Out[6]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [7]:
```
# Check for null values in the dataset
print(df.isnull().sum())
```

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

In [8]:
```
# Independent variables (X) - All columns except the target ('variety')
X = df.drop('variety', axis=1)

# Dependent variable (y) - The 'variety' column
y = df['variety']
```

In [9]:
```
X
```

Out[9]:

	sepal.length	sepal.width	petal.length	petal.width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [10]:
```
y
```

Out[10]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: variety, Length: 150, dtype: int32
```

In [11]:
```
# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Print the shape of the splits to verify
print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}")
```

```
Training data shape: (120, 4)
Testing data shape: (30, 4)
```

In [12]:
```
X_train
```

Out[12]:

	sepal.length	sepal.width	petal.length	petal.width
22	4.6	3.6	1.0	0.2
15	5.7	4.4	1.5	0.4
65	6.7	3.1	4.4	1.4
11	4.8	3.4	1.6	0.2
42	4.4	3.2	1.3	0.2
...	...	...	...	...
71	6.1	2.8	4.0	1.3
106	4.9	2.5	4.5	1.7
14	5.8	4.0	1.2	0.2
92	5.8	2.6	4.0	1.2
102	7.1	3.0	5.9	2.1

120 rows × 4 columns

In [13]:
```
X_test
```

Out[13]:

	sepal.length	sepal.width	petal.length	petal.width
73	6.1	2.8	4.7	1.2
18	5.7	3.8	1.7	0.3
118	7.7	2.6	6.9	2.3
78	6.0	2.9	4.5	1.5
76	6.8	2.8	4.8	1.4
31	5.4	3.4	1.5	0.4
64	5.6	2.9	3.6	1.3
141	6.9	3.1	5.1	2.3
68	6.2	2.2	4.5	1.5
82	5.8	2.7	3.9	1.2
110	6.5	3.2	5.1	2.0
12	4.8	3.0	1.4	0.1
36	5.5	3.5	1.3	0.2
9	4.9	3.1	1.5	0.1
19	5.1	3.8	1.5	0.3
56	6.3	3.3	4.7	1.6
104	6.5	3.0	5.8	2.2
69	5.6	2.5	3.9	1.1
55	5.7	2.8	4.5	1.3
132	6.4	2.8	5.6	2.2
29	4.7	3.2	1.6	0.2
127	6.1	3.0	4.9	1.8
26	5.0	3.4	1.6	0.4
128	6.4	2.8	5.6	2.1
131	7.9	3.8	6.4	2.0
145	6.7	3.0	5.2	2.3
108	6.7	2.5	5.8	1.8
143	6.8	3.2	5.9	2.3
45	4.8	3.0	1.4	0.3
30	4.8	3.1	1.6	0.2

In [14]:
```
y_train
```

Out[14]:
```
22     0
15     0
65     1
11     0
42     0
      ..
71     1
106    2
14     0
92     1
102    2
Name: variety, Length: 120, dtype: int32
```

In [15]:
```
y_test
```

Out[15]:
```
73     1
18     0
118    2
78     1
76     1
31     0
64     1
141    2
68     1
82     1
110    2
12     0
36     0
9      0
19     0
56     1
104    2
69     1
55     1
132    2
29     0
127    2
26     0
128    2
131    2
145    2
108    2
143    2
45     0
30     0
Name: variety, dtype: int32
```

In [16]:
```
# Initialize the StandardScaler
scaler = StandardScaler()

# Fit on training data and transform both training and testing data
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [17]:
```
X_train
```

Out[17]:
```
array([[-1.47393679,  1.20365799, -1.56253475, -1.31260282],
       [-0.13307079,  2.99237573, -1.27600637, -1.04563275],
       [ 1.08589829,  0.08570939,  0.38585821,  0.28921757],
       [-1.23014297,  0.75647855, -1.2187007 , -1.31260282],
       [-1.7177306 ,  0.30929911, -1.39061772, -1.31260282],
       [ 0.59831066, -1.25582892,  0.72969227,  0.95664273],
       [ 0.72020757,  0.30929911,  0.44316389,  0.4227026 ],
       [-0.74255534,  0.98006827, -1.27600637, -1.31260282],
       [-0.98634915,  1.20365799, -1.33331205, -1.31260282],
       [-0.74255534,  2.32160658, -1.27600637, -1.44608785],
       [-0.01117388, -0.80864948,  0.78699794,  0.95664273],
       [ 0.23261993,  0.75647855,  0.44316389,  0.55618763],
       [ 1.08589829,  0.08570939,  0.55777524,  0.4227026 ],
       [-0.49876152,  1.87442714, -1.39061772, -1.04563275],
       [-0.49876152,  1.4272477 , -1.27600637, -1.31260282],
       [-0.37686461, -1.47941864, -0.01528151, -0.24472256],
       [ 0.59831066, -0.58505976,  0.78699794,  0.4227026 ],
       [ 0.72020757,  0.08570939,  1.01622064,  0.8231577 ],
       [ 0.96400139, -0.13788033,  0.38585821,  0.28921757],
```

In [18]:
```
X_test
```

Out[18]:
```
array([[ 0.35451684, -0.58505976,  0.55777524,  0.02224751],
       [-0.13307079,  1.65083742, -1.16139502, -1.17911778],
       [ 2.30486738, -1.0322392 ,  1.8185001 ,  1.49058286],
       [ 0.23261993, -0.36147005,  0.44316389,  0.4227026 ],
       [ 1.2077952 , -0.58505976,  0.61508092,  0.28921757],
       [-0.49876152,  0.75647855, -1.27600637, -1.04563275],
       [-0.2549677 , -0.36147005, -0.07258719,  0.15573254],
       [ 1.32969211,  0.08570939,  0.78699794,  1.49058286],
       [ 0.47641375, -1.92659808,  0.44316389,  0.4227026 ],
       [-0.01117388, -0.80864948,  0.09932984,  0.02224751],
       [ 0.84210448,  0.30929911,  0.78699794,  1.09012776],
       [-1.23014297, -0.13788033, -1.33331205, -1.44608785],
       [-0.37686461,  0.98006827, -1.39061772, -1.31260282],
       [-1.10824606,  0.08570939, -1.27600637, -1.44608785],
       [-0.86445224,  1.65083742, -1.27600637, -1.17911778],
       [ 0.59831066,  0.53288883,  0.55777524,  0.55618763],
       [ 0.84210448, -0.13788033,  1.18813767,  1.35709783],
       [-0.2549677 , -1.25582892,  0.09932984, -0.11123753],
       [-0.13307079, -0.58505976,  0.44316389,  0.15573254],
       [ 0.72020757, -0.58505976,  1.07352632,  1.35709783],
       [-1.35203988,  0.30929911, -1.2187007 , -1.31260282],
       [ 0.35451684, -0.13788033,  0.67238659,  0.8231577 ],
       [-0.98634915,  0.75647855, -1.2187007 , -1.04563275],
       [ 0.72020757, -0.58505976,  1.07352632,  1.22361279],
       [ 2.5486612 ,  1.65083742,  1.53197172,  1.09012776],
       [ 1.08589829, -0.13788033,  0.84430362,  1.49058286],
       [ 1.08589829, -1.25582892,  1.18813767,  0.8231577 ],
       [ 1.2077952 ,  0.30929911,  1.24544335,  1.49058286],
       [-1.23014297, -0.13788033, -1.33331205, -1.17911778],
       [-1.23014297,  0.08570939, -1.2187007 , -1.31260282]])
```

In [19]:
```
# Initialize the Naive Bayes model
gaussian = GaussianNB()

# Train the model on the training data
gaussian.fit(X_train, y_train)
```

Out[19]:
```
GaussianNB
GaussianNB()
```
(https://scikit-learn.org/1.4/modules/generated/sklearn.naive_bayes.GaussianNB.html)

In [20]:
```
# Predict on the training data
y_train_pred = gaussian.predict(X_train)

# Predict on the testing data
y_test_pred = gaussian.predict(X_test)
```

In [21]:
```
y_train_pred
```

Out[21]:
```
array([0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1, 1, 2, 1, 2, 1, 2,
       1, 0, 2, 1, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 1, 2, 0, 1, 2, 0, 2, 2,
       1, 1, 2, 1, 0, 1, 2, 0, 0, 1, 2, 0, 2, 0, 0, 2, 1, 2, 1, 2, 2, 1,
       0, 0, 1, 2, 0, 0, 0, 1, 2, 0, 2, 2, 0, 1, 1, 2, 1, 2, 0, 2, 1, 2,
       1, 1, 1, 0, 1, 1, 0, 1, 2, 2, 0, 1, 2, 2, 0, 2, 0, 2, 2, 2, 1, 2,
       1, 1, 2, 2, 0, 1, 1, 0, 1, 2])
```

In [22]:
```
y_test_pred
```

Out[22]:
```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0])
```

In [23]:
```
train_accuracy = accuracy_score(y_train, y_train_pred)
train_accuracy
```

Out[23]:
```
0.95
```

In [24]:
```
train_precision = precision_score(y_train, y_train_pred, average='micro')
train_precision
```

Out[24]:
```
0.95
```

In [25]:
```
train_recall = recall_score(y_train, y_train_pred, average='micro')
train_recall
```

Out[25]:
```
0.95
```

In [26]:
```
test_accuracy = accuracy_score(y_test, y_test_pred)
test_accuracy
```

Out[26]:
```
1.0
```

In [27]:
```
test_precision = precision_score(y_test, y_test_pred, average='micro')
test_precision
```

Out[27]:
```
1.0
```

In [28]:
```
test_recall = recall_score(y_test, y_test_pred, average='micro')
test_recall
```

Out[28]:
```
1.0
```

In [29]:
```
cm = confusion_matrix(y_test, y_test_pred)
cm
```

Out[29]:
```
array([[10,  0,  0],
       [ 0,  9,  0],
       [ 0,  0, 11]], dtype=int64)
```

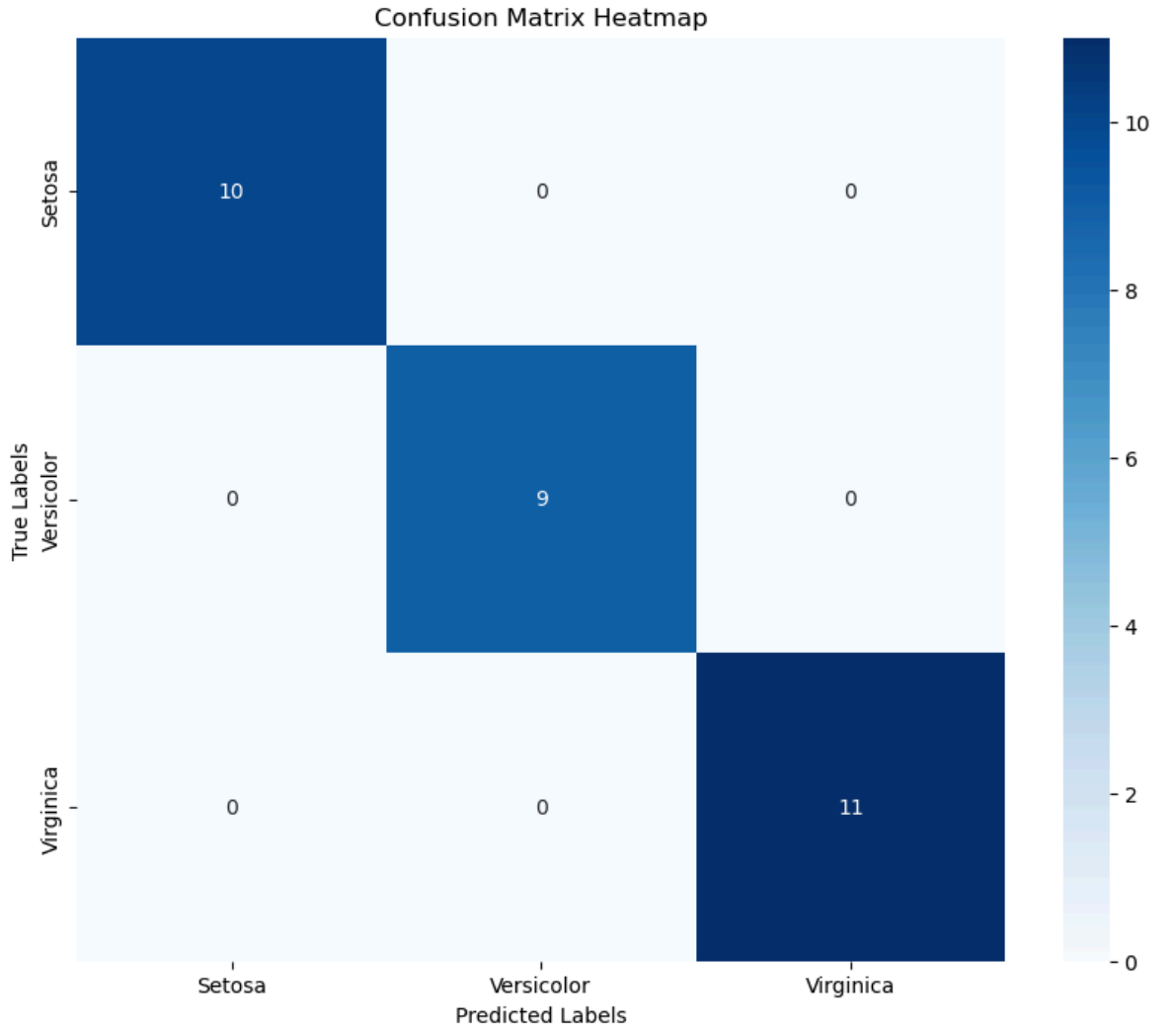
In [30]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Confusion matrix for testing data
cm = confusion_matrix(y_test, y_test_pred)

# Create a heatmap to visualize the confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=le.classes_, yt

# Add labels and title
plt.title("Confusion Matrix Heatmap")
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')

# Display the heatmap
plt.show()
```



In [ ]:
```
Name:Kadhane Pratiksha
Rollno:13213
Batch:B1
```