practical-6

May 9, 2025

[4]:
```
import numpy as np
```

[5]:
```
import pandas as pd
```

[6]:
```
from sklearn.model_selection import train_test_split
```

[7]:
```
from sklearn.naive_bayes import GaussianNB
```

[8]:
```
import matplotlib.pyplot as plt
```

[9]:
```
import seaborn as sns
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[9], line 1
----> 1 import seaborn as sns

ModuleNotFoundError: No module named 'seaborn'
```

[10]:
```
!pip install seaborn
```

```
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
seaborn) (2.2.5)
Requirement already satisfied: pandas>=1.2 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
seaborn) (2.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
seaborn) (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in
c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from
python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2


[notice] A new release of pip is available: 24.2 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[11]:

```python
import seaborn as sns
```

[12]:

```python
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,
classification_report,accuracy_score, precision_score, recall_score, f1_score
```

```
  Cell In[12], line 1
    from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,
                                                                        ^
SyntaxError: trailing comma not allowed without surrounding parentheses
```

```
[13]: from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
       ↪classification_report, accuracy_score, precision_score, recall_score,␣
       ↪f1_score
```

```
[14]: from sklearn.metrics import (
          confusion_matrix,
          ConfusionMatrixDisplay,
          classification_report,
          accuracy_score,
          precision_score,
          recall_score,
          f1_score
      )
```

```
[15]: from sklearn.preprocessing import LabelEncoder
```

```
[16]: import warnings
```

```
[17]: warnings.filterwarnings("ignore")
```

```
[19]: df=pd.read_csv("iris.csv")
```

```
[20]: df
```

```
[20]:       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
      0      1            5.1           3.5            1.4           0.2
      1      2            4.9           3.0            1.4           0.2
      2      3            4.7           3.2            1.3           0.2
      3      4            4.6           3.1            1.5           0.2
      4      5            5.0           3.6            1.4           0.2
      ..   …            …           …            …           …
      145  146            6.7           3.0            5.2           2.3
      146  147            6.3           2.5            5.0           1.9
      147  148            6.5           3.0            5.2           2.0
      148  149            6.2           3.4            5.4           2.3
      149  150            5.9           3.0            5.1           1.8

                  Species
      0       Iris-setosa
      1       Iris-setosa
      2       Iris-setosa
      3       Iris-setosa
      4       Iris-setosa
      ..              …
      145  Iris-virginica
      146  Iris-virginica
      147  Iris-virginica
```

```
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

[21]:
```
df.head(5)
```

[21]:
```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

[22]:
```
df.tail()
```

[22]:
```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

[23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

[26]:
```
print(df.shape)
```

```
(150, 6)
```

[27]:

```
df['Species'].unique()
```

[27]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

[28]:

```
df.isnull().sum()
```

[28]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

[29]:

```
x = df.iloc[:,1:4]
y = df.iloc[:,4:5]
```

[30]:

```
encode = LabelEncoder()
y_encoded = encode.fit_transform(y)
```

[31]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y_encoded,test_size = 0.3,
random_state = 0)
```

[32]:

```
naive_bayes = GaussianNB()
naive_bayes.fit(x_train,y_train)
pred = naive_bayes.predict(x_test)
```

[33]:

```
pred
```

[33]:

```
array([14,  9,  3, 17,  1, 19,  1, 10, 11,  9, 15, 10, 11, 11, 11,  1,  9,
        9,  1,  1, 13,  9,  1,  2, 11,  0,  1,  9,  6,  1, 14,  9,  1, 14,
       20,  9,  3, 15,  9,  9, 17,  1, 14,  1,  3])
```

[34]:

```
y_test
```

[34]:

```
array([20,  6,  1, 14,  1, 21,  2, 11, 10,  9, 10, 11,  8, 11, 10,  0, 11,
        8,  2,  3, 16, 11,  1,  1, 14,  1,  3,  9,  6,  3, 14, 11,  1, 14,
       18, 10,  2, 12, 11,  8, 17,  1, 19,  5,  1])
```

[38]:

```
matrix = confusion_matrix(y_test,pred,labels = naive_bayes.classes_)
print(matrix)
tp, fn, fp, tn = confusion_matrix(y_test,pred,labels=[1,0]).reshape(-1)
```

```
[[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 4 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 2 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 2 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 4 2 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 2 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]]
```

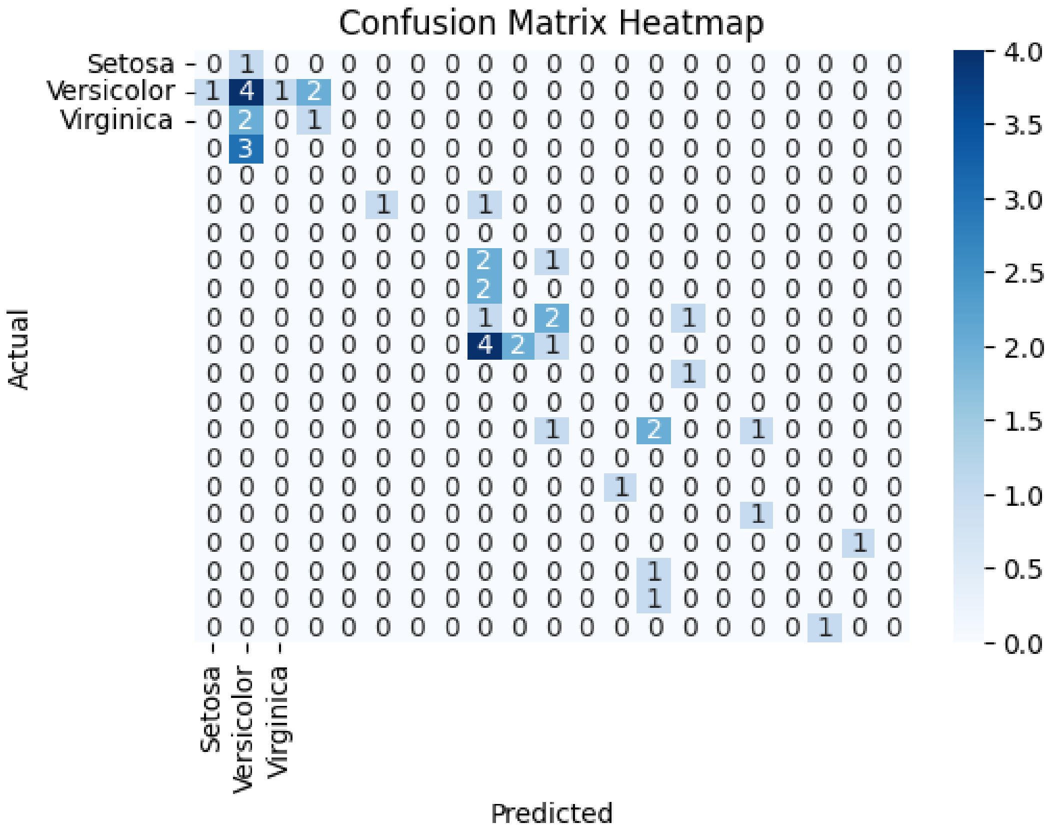
[39]:

```
accuracy = accuracy_score(y_test, pred)
precision = precision_score(y_test, pred, average='micro')
recall = recall_score(y_test, pred, average='micro')
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

```
Accuracy: 0.24
Precision: 0.24
Recall: 0.24
```

[40]:

```
plt.figure(figsize=(6,4))
sns.heatmap(matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Setosa',
'Versicolor','Virginica'],
yticklabels=['Setosa', 'Versicolor', 'Virginica'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```



[41]:

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.36      0.50      0.42         8
           2       0.00      0.00      0.00         3
           3       0.00      0.00      0.00         3
           5       0.00      0.00      0.00         1
           6       1.00      0.50      0.67         2
           8       0.00      0.00      0.00         3
           9       0.20      1.00      0.33         2
          10       0.00      0.00      0.00         4
          11       0.20      0.14      0.17         7
          12       0.00      0.00      0.00         1
          13       0.00      0.00      0.00         0
          14       0.50      0.50      0.50         4
          15       0.00      0.00      0.00         0
          16       0.00      0.00      0.00         1
          17       0.50      1.00      0.67         1
          18       0.00      0.00      0.00         1
          19       0.00      0.00      0.00         1
          20       0.00      0.00      0.00         1
          21       0.00      0.00      0.00         1

    accuracy                           0.24        45
   macro avg       0.14      0.18      0.14        45
weighted avg       0.20      0.24      0.20        45
```

[42]:

```
print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test,pred)))
print('Error Rate: ',(fp+fn)/(tp+tn+fn+fp))
print('Sensitivity (Recall or True positive rate) :',tp/(tp+fn))
print('Specificity (True negative rate) :',tn/(fp+tn))
print('Precision (Positive predictive value) :',tp/(tp+fp))
print('False Positive Rate :',fp/(tn+fp))
```

```

Accuracy: 0.24
Error Rate:  0.3333333333333333
Sensitivity (Recall or True positive rate) : 0.8
Specificity (True negative rate) : 0.0
Precision (Positive predictive value) : 0.8
False Positive Rate : 1.0
```

[ ]:

```

```