p6

March 30, 2024

[2]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("iris.csv")
br = "\n\n\n\n"
print("Datatypes: \n", df.dtypes)
print(br, "Shape is: ", df.shape)
```

```
Datatypes:
 sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object




 Shape is:  (150, 5)
```

[3]:

```
x = df.drop("species", axis=1)

y = df["species"]
```

[5]:

```
print("Counts: ",br, y.value_counts())
```

```
Counts:



 setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

[6]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x, y, random_state=42,␣
  ↪test_size=0.25)
```

[8]:

```python
print("X train shape: ", X_train.shape)
print(br, "X test shape: ", X_test.shape)
```

```
X train shape:  (112, 4)



 X test shape:  (38, 4)
```

[9]:

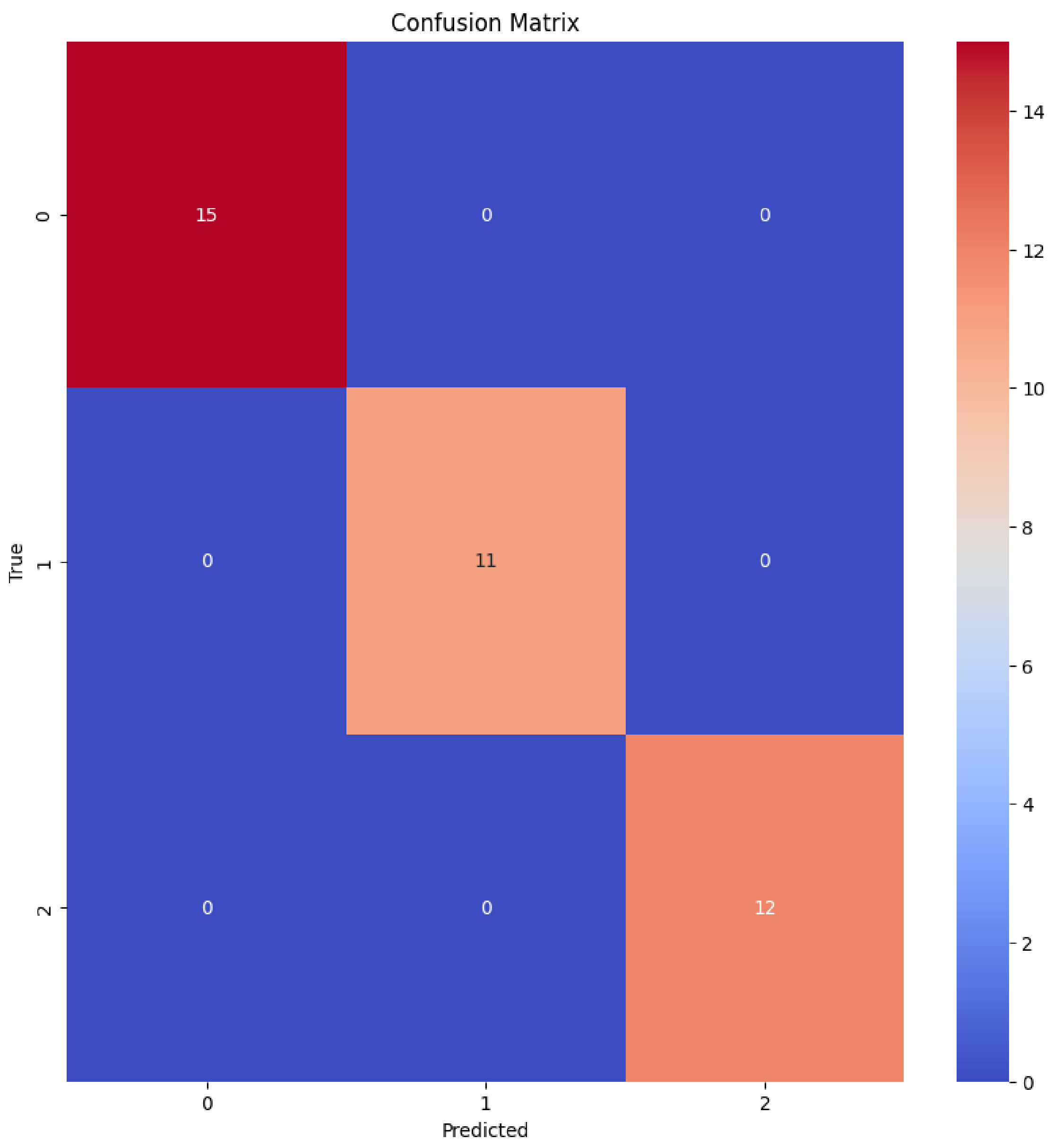
```python
from sklearn.naive_bayes import GaussianNB
cls = GaussianNB()
cls.fit(X_train, Y_train)
Y_pred = cls.predict(X_test)
```

[11]:

```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

[15]:

```python
cm = confusion_matrix(Y_test, Y_pred)

# Create a new figure and a set of subplots
fig, ax = plt.subplots(figsize=(10, 10))

# Create a heatmap of the confusion matrix with a custom color map
sns.heatmap(cm, annot=True, fmt='d', cmap='coolwarm', ax=ax)

# Set the title of the confusion matrix
ax.set_title('Confusion Matrix')

# Set the x and y axis labels
ax.set_xlabel('Predicted')
ax.set_ylabel('True')

# Display the confusion matrix
plt.show()
```



```
[17]: print("Accuracy Score is ", accuracy_score(Y_test,Y_pred))
```

```
Accuracy Score is  1.0
```

```
[19]: cls.predict_proba(X_test)
```

```
[19]: array([[5.97926853e-094, 9.95499546e-001, 4.50045384e-003],
             [1.00000000e+000, 1.52146968e-013, 1.07136902e-020],
             [1.71330808e-299, 6.04667826e-012, 1.00000000e+000],
             [2.88508207e-096, 9.76485329e-001, 2.35146713e-002],
             [1.36797133e-109, 8.51147229e-001, 1.48852771e-001],
             [1.00000000e+000, 4.74962788e-013, 4.28854236e-021],
             [3.25153316e-053, 9.99959350e-001, 4.06500520e-005],
             [1.09216160e-176, 1.05015117e-006, 9.99998950e-001],
             [3.27169186e-098, 9.93646597e-001, 6.35340277e-003],
             [1.17401351e-060, 9.99944993e-001, 5.50073382e-005],
             [1.00765817e-153, 5.02929583e-004, 9.99497070e-001],
             [1.00000000e+000, 7.95517827e-017, 5.80301835e-025],
             [1.00000000e+000, 1.83324108e-016, 2.96899989e-024],
             [1.00000000e+000, 1.81709952e-016, 2.19054140e-024],
             [1.00000000e+000, 2.58611124e-016, 6.24907433e-024],
             [4.90534771e-110, 5.45081346e-001, 4.54918654e-001],
             [3.16184825e-207, 5.32942939e-007, 9.99999467e-001],
             [5.54971964e-057, 9.99985948e-001, 1.40522914e-005],
             [2.35216801e-087, 9.98060492e-001, 1.93950811e-003],
             [8.52134069e-195, 3.31416502e-006, 9.99996686e-001],
             [1.00000000e+000, 2.01691401e-015, 2.19989447e-023],
             [1.10030136e-129, 9.28827573e-002, 9.07117243e-001],
             [1.00000000e+000, 3.55180650e-013, 3.14309037e-021],
             [4.61090739e-188, 1.99740486e-005, 9.99980026e-001],
             [5.38997715e-243, 9.37394931e-010, 9.99999999e-001],
             [8.41240591e-181, 1.33362177e-006, 9.99998666e-001],
             [7.84665916e-184, 7.04291221e-004, 9.99295709e-001],
             [4.06568213e-222, 1.32017082e-008, 9.99999987e-001],
             [1.00000000e+000, 7.64632371e-015, 2.10335817e-023],
             [1.00000000e+000, 4.36746748e-015, 3.82841755e-023],
             [1.00000000e+000, 1.67045858e-018, 1.11343221e-026],
             [1.00000000e+000, 6.19486746e-016, 6.98399326e-023],
             [1.29430190e-090, 9.62168850e-001, 3.78311504e-002],
             [1.00000000e+000, 6.72672239e-016, 1.17370919e-023],
             [1.00000000e+000, 5.25411235e-017, 2.85220284e-025],
             [4.98861144e-142, 3.13601455e-002, 9.68639854e-001],
             [2.84242806e-097, 9.05614884e-001, 9.43851163e-002],
             [1.00000000e+000, 2.90496807e-016, 5.87418518e-024]])
```

[23]:

```
newl=[[4.5,2.9,3.1,0.4]]
print("Row1 prediction: ", cls.predict(newl)[0])

newl=[[5.5,3.1,1.0,0.8]]
print("Row2 prediction: ", cls.predict(newl)[0])

newl=[[6.5,3.3,4.9,1.8]]
print("Row3 prediction: ", cls.predict(newl)[0])
print(br)
```

```
Row1 prediction:  versicolor
Row2 prediction:  setosa
```

```
Row3 prediction:  virginica
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
```

[24]:

```
print(classification_report(Y_test,Y_pred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       1.00      1.00      1.00        11
   virginica       1.00      1.00      1.00        12

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38
```

[ ]:

```

```