untitled6

May 7, 2024

[47]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

[7]:
```
df = pd.read_csv('iris.csv')
df
```

[7]:
```
      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]
```

[8]:
```python
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
```

[ ]:

[ ]:

[ ]:

[ ]:

[16]:
```python
df['Species'].replace({'Iris-setosa':1,'Iris-versicolor':2,'Iris-virginica':
↪3},inplace=True)
```

```
C:\Users\PUSHKAR\AppData\Local\Temp\ipykernel_6712\2189725869.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Species'].replace({'Iris-setosa':1,'Iris-versicolor':2,'Iris-
virginica':3},inplace=True)
C:\Users\PUSHKAR\AppData\Local\Temp\ipykernel_6712\2189725869.py:1:
FutureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
```

```
`pd.set_option('future.no_silent_downcasting', True)`
  df['Species'].replace({'Iris-setosa':1,'Iris-versicolor':2,'Iris-
virginica':3},inplace=True)
```

[18]:
```
df
```

[18]:
```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0      1            5.1           3.5            1.4           0.2        1
1      2            4.9           3.0            1.4           0.2        1
2      3            4.7           3.2            1.3           0.2        1
3      4            4.6           3.1            1.5           0.2        1
4      5            5.0           3.6            1.4           0.2        1
..   …            …             …              …             …       …
145  146            6.7           3.0            5.2           2.3        3
146  147            6.3           2.5            5.0           1.9        3
147  148            6.5           3.0            5.2           2.0        3
148  149            6.2           3.4            5.4           2.3        3
149  150            5.9           3.0            5.1           1.8        3

[150 rows x 6 columns]
```

[19]:
```
features = df.iloc[:,:-1]
target = df.iloc[:,-1]
```

[20]:
```
features.head(5)
```

[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
```

[32]:
```
x_train,x_test,y_train,y_test = train_test_split(features,target,test_size=0.40)
```

[26]:
```
x_train.shape
```

[26]:
```
(90, 5)
```

[27]:
```
y_train.shape
```

[27]:
```
(60, 5)
```

[28]:
```
x_test.shape
```

[28]:
```
(90,)
```

```
[29]: y_test.shape
```

```
[29]: (60,)
```

```
[33]: model = GaussianNB()
      model.fit(x_train,y_train)
```

```
[33]: GaussianNB()
```

```
[35]: pred = model.predict(x_test)
      pred
```

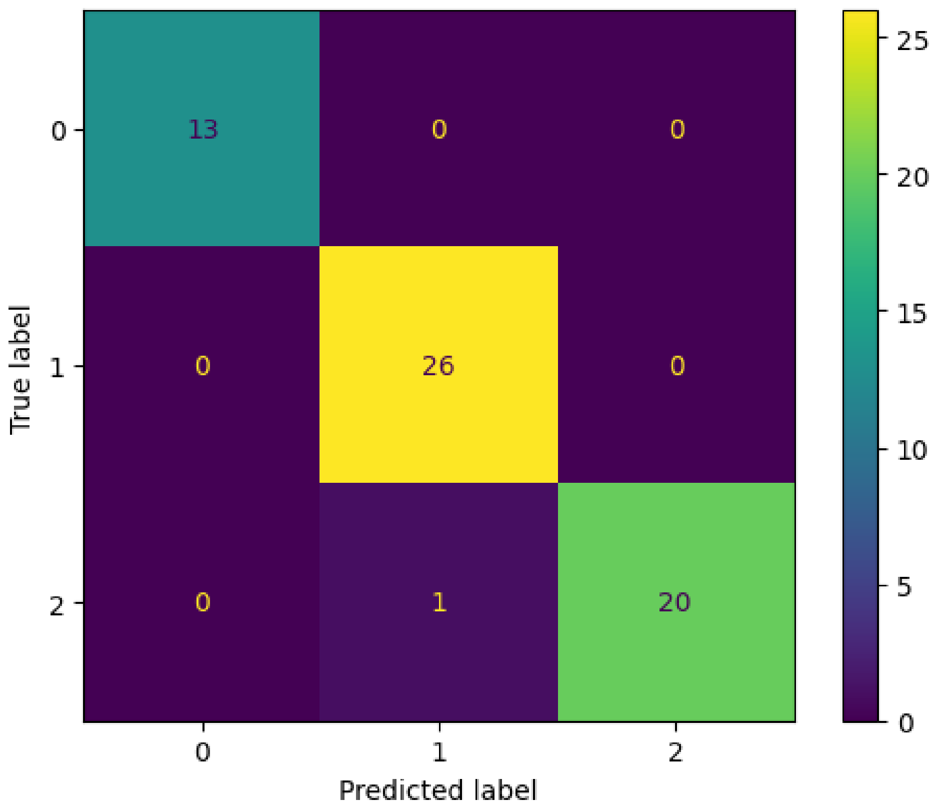
```
[35]: array([2, 3, 3, 3, 3, 3, 3, 3, 2, 1, 2, 2, 3, 3, 1, 2, 2, 2, 2, 2, 1, 2,
             1, 1, 1, 2, 2, 3, 1, 2, 2, 2, 3, 3, 3, 2, 3, 2, 1, 2, 1, 2, 2, 2,
             3, 1, 3, 3, 3, 1, 3, 3, 2, 2, 2, 2, 1, 1, 2, 2], dtype=int64)
```

```
[41]: cm = confusion_matrix(y_test,pred)
      cm
```

```
[41]: array([[13,  0,  0],
             [ 0, 26,  0],
             [ 0,  1, 20]], dtype=int64)
```

```
[44]: disp = ConfusionMatrixDisplay(confusion_matrix = cm)
      disp.plot()
```

```
[44]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2a5df2ec470>
```



[48]:
```
accuracy_score(y_test,pred)
```

[48]:
```
0.9833333333333333
```

[50]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        13
           2       0.96      1.00      0.98        26
           3       1.00      0.95      0.98        21

    accuracy                           0.98        60
   macro avg       0.99      0.98      0.99        60
weighted avg       0.98      0.98      0.98        60
```

[ ]:
```

```