# Exercise8

November 2, 2025

[1]:
```
import pandas as pd
df=pd.read_csv('Iris (1).csv')
```

[8]:
```
df.head(5)
```

[8]:

| | 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 |

[9]:
```
df.variety.value_counts()
```

[9]:
```
variety
Setosa        50
Versicolor    50
Virginica     50
Name: count, dtype: int64
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[16]:
```
df["output"] = 0
```

[42]:
```
df.loc[:49, "output"] = 1
```

[43]:
```
df.loc[50:99,"output"]=2
```

[44]:
```
df.loc[100:149,"output"]=3
```

[45]:
```
df.head(5)
```

[45]:

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

[46]:
```
df.tail(5)
```

[46]:

| | sepal.length | sepal.width | petal.length | petal.width | variety | output |
|---|---|---|---|---|---|---|
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | Virginica | 3 |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | Virginica | 3 |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | Virginica | 3 |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | Virginica | 3 |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | Virginica | 3 |

[47]:
```
from sklearn.model_selection import train_test_split
```

[48]:
```
feature=df
label=df
```

[56]:
```
feature=df.drop("output",axis=1)
for col in feature.columns:
    if feature[col].dtype == 'object':
        le = LabelEncoder()
        feature[col] = le.fit_transform(feature[col])
label=df["output"]
```

[57]:
```
feature
```

[57]:

| | 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 x 5 columns]
```

[58]:
```
label
```

[58]:
```
0      1
1      1
2      1
3      1
4      1
      ..
145    3
146    3
147    3
148    3
149    3
Name: output, Length: 150, dtype: int64
```

[59]:
```
X_train,X_test,Y_train,y_test=train_test_split(feature,label,test_size=0.
↪2,random_state=1)
```

[60]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[61]:
```
op=KNeighborsClassifier(n_neighbors=5)
```

[62]:
```
op.fit(X_train,Y_train)
```

[62]:
```
KNeighborsClassifier()
```

[64]:
```
print(op.score(X_test,y_test))
```

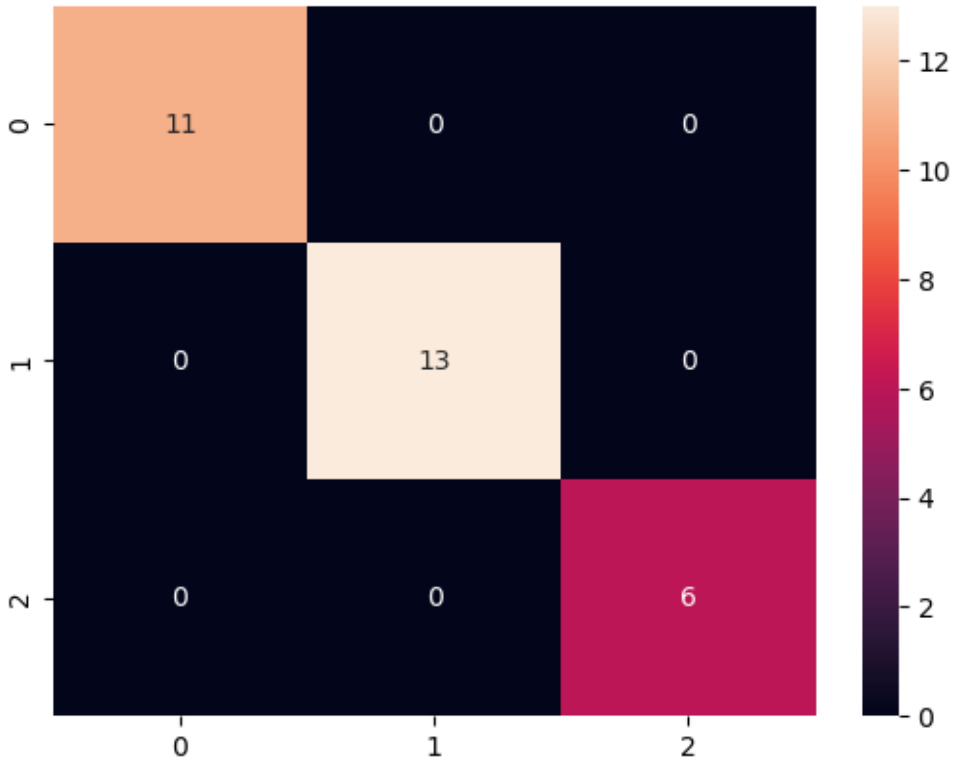
```
1.0
```

[65]:
```
from sklearn.metrics import confusion_matrix
y_pred=op.predict(X_test)
```

[69]:
```
c_n_m=confusion_matrix(y_test,y_pred)
```

[70]:
```
import seaborn as sn
sn.heatmap(c_n_m,annot=True)
```

[70]:
```
<Axes: >
```




[72]:
```
from sklearn.metrics import classification_report
print(classification_report(label,op.predict(feature)))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        50
           2       1.00      1.00      1.00        50
           3       1.00      1.00      1.00        50

    accuracy                           1.00       150
   macro avg       1.00      1.00      1.00       150
weighted avg       1.00      1.00      1.00       150
```

[ ]: