week8

September 25, 2024

[1]:
```
import pandas as pd
fish = pd.read_csv("C:/Users/HP/Downloads/Fish.csv")
fish.head()
```

[1]:
```
   Category Species  Weight   Height   Width  Length1  Length2  Length3
0         1   Bream   242.0  11.5200  4.0200     23.2     25.4     30.0
1         1   Bream   290.0  12.4800  4.3056     24.0     26.3     31.2
2         1   Bream   340.0  12.3778  4.6961     23.9     26.5     31.1
3         1   Bream   363.0  12.7300  4.4555     26.3     29.0     33.5
4         1   Bream   430.0  12.4440  5.1340     26.5     29.0     34.0
```

[3]:
```
fish['Species'].unique()
```

[3]:
```
array(['Bream', 'Roach', 'Whitefish', 'Parkki', 'Perch', 'Pike', 'Smelt'],
      dtype=object)
```

[4]:
```
fish.isnull().sum()
```

[4]:
```
Category    0
Species     0
Weight      0
Height      0
Width       0
Length1     0
Length2     0
Length3     0
dtype: int64
```

[5]:
```
X = fish.iloc[:, 1:]
y = fish.loc[:, 'Species']
```

[6]:
```
print(X.head())
```

```
  Species  Weight   Height   Width  Length1  Length2  Length3
0   Bream   242.0  11.5200  4.0200     23.2     25.4     30.0
1   Bream   290.0  12.4800  4.3056     24.0     26.3     31.2
2   Bream   340.0  12.3778  4.6961     23.9     26.5     31.1
3   Bream   363.0  12.7300  4.4555     26.3     29.0     33.5
4   Bream   430.0  12.4440  5.1340     26.5     29.0     34.0
```

[7]:

```python
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
import pandas as pd
```

[8]:

```python
data = pd.DataFrame({
    'Fish': ['Bream', 'Salmon', 'Bream', 'Trout'],
    'Weight': [150, 300, 170, 220]
})
```

[9]:

```python
label_encoder = LabelEncoder()
data['Fish'] = label_encoder.fit_transform(data['Fish'])

# Separate features and target
X = data[['Fish']]  # Features
y = data['Weight']  # Target

# Apply MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)

print("Scaled Features:")
print(X_scaled)
```

```
Scaled Features:
[[0. ]
 [0.5]
 [0. ]
 [1. ]]
```

[10]:

```python
scaler = MinMaxScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
print("Scaled Features:")
print(X_scaled)
```

```
Scaled Features:
[[0. ]
 [0.5]
 [0. ]
 [1. ]]
```

[11]:

```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
```

[12]:

```python
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
```

```
y = label_encoder.fit_transform(y)
y
```

```
[12]: array([0, 3, 1, 2], dtype=int64)
```

```
[13]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test= train_test_split(X_scaled, y, test_size=0.2,␣
       ↪random_state=42)
```

```
[14]: from sklearn.linear_model import LogisticRegression
      logReg = LogisticRegression()
      logReg.fit(X_train, y_train)
```

```
[14]: LogisticRegression()
```

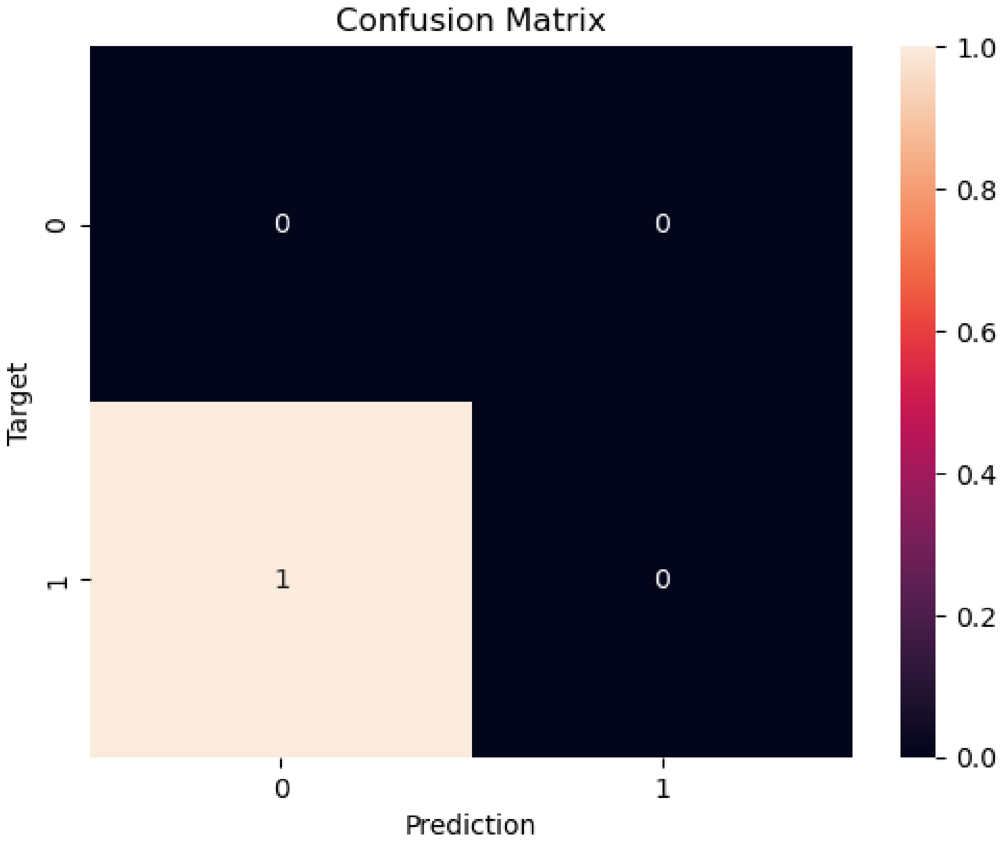
```
[15]: y_pred = logReg.predict(X_test)
```

```
[16]: from sklearn.metrics import accuracy_score
      accuracy = accuracy_score(y_test, y_pred)
      print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
Accuracy: 0.00%
```

```
[17]: from sklearn.metrics import confusion_matrix
      import matplotlib.pyplot as plt
      import seaborn as sns
      cf = confusion_matrix(y_test, y_pred)
      plt.figure()
      sns.heatmap(cf, annot=True)
      plt.xlabel('Prediction')
      plt.ylabel('Target')
      plt.title('Confusion Matrix')
```

```
[17]: Text(0.5, 1.0, 'Confusion Matrix')
```



[18]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
```

[20]:

```
data = pd.read_csv('C:/Users/HP/Downloads/apples_and_oranges.csv')
print(data)
```

```
   Weight  Size   Class
0      69  4.39  orange
1      69  4.21  orange
2      65  4.09  orange
3      72  5.85   apple
4      67  4.70  orange
5      73  5.68   apple
6      70  5.56   apple
7      75  5.11   apple
8      74  5.36   apple
9      65  4.27  orange
```

```
10       73  5.79   apple
11       70  5.47   apple
12       74  5.53   apple
13       68  4.47  orange
14       74  5.22   apple
15       65  4.48  orange
16       69  4.66  orange
17       75  5.25   apple
18       67  4.18  orange
19       74  5.50   apple
20       66  4.13  orange
21       70  4.83  orange
22       69  4.61  orange
23       68  4.08  orange
24       67  4.25  orange
25       71  5.35   apple
26       67  4.01  orange
27       70  4.22  orange
28       74  5.25   apple
29       71  5.26   apple
30       73  5.78   apple
31       66  4.68  orange
32       72  5.72   apple
33       73  5.17   apple
34       68  4.83  orange
35       69  4.11  orange
36       69  4.76  orange
37       74  5.48   apple
38       70  5.59   apple
39       73  5.03   apple
```

[21]:

```
training_set,test_set = train_test_split(data,test_size=0.2,random_state=1)
print("train:",training_set)
print("test:",test_set)
```

```
train:     Weight  Size   Class
19       74  5.50   apple
26       67  4.01  orange
32       72  5.72   apple
17       75  5.25   apple
30       73  5.78   apple
36       69  4.76  orange
33       73  5.17   apple
28       74  5.25   apple
4        67  4.70  orange
14       74  5.22   apple
10       73  5.79   apple
35       69  4.11  orange
```

```
23      68  4.08  orange
24      67  4.25  orange
34      68  4.83  orange
20      66  4.13  orange
18      67  4.18  orange
25      71  5.35   apple
6       70  5.56   apple
13      68  4.47  orange
7       75  5.11   apple
38      70  5.59   apple
1       69  4.21  orange
16      69  4.66  orange
0       69  4.39  orange
15      65  4.48  orange
5       73  5.68   apple
11      70  5.47   apple
9       65  4.27  orange
8       74  5.36   apple
12      74  5.53   apple
37      74  5.48   apple
test:     Weight  Size   Class
2       65  4.09  orange
31      66  4.68  orange
3       72  5.85   apple
21      70  4.83  orange
27      70  4.22  orange
29      71  5.26   apple
22      69  4.61  orange
39      73  5.03   apple
```

[22]:

```
x_train = training_set.iloc[:,0:2].values  # data
y_train = training_set.iloc[:,2].values  # target
x_test = test_set.iloc[:,0:2].values  # data
y_test = test_set.iloc[:,2].values  # target
print(x_train,y_train)
print(x_test,y_test)
```

```
[[74.    5.5 ]
 [67.    4.01]
 [72.    5.72]
 [75.    5.25]
 [73.    5.78]
 [69.    4.76]
 [73.    5.17]
 [74.    5.25]
 [67.    4.7 ]
 [74.    5.22]
 [73.    5.79]
 [69.    4.11]
 [68.    4.08]
 [67.    4.25]
 [68.    4.83]
 [66.    4.13]
 [67.    4.18]
 [71.    5.35]
 [70.    5.56]
 [68.    4.47]
 [75.    5.11]
 [70.    5.59]
 [69.    4.21]
 [69.    4.66]
 [69.    4.39]
 [65.    4.48]
 [73.    5.68]
 [70.    5.47]
 [65.    4.27]
 [74.    5.36]
 [74.    5.53]
 [74.    5.48]] ['apple' 'orange' 'apple' 'apple' 'apple' 'orange' 'apple'
'apple'
 'orange' 'apple' 'apple' 'orange' 'orange' 'orange' 'orange' 'orange'
 'orange' 'apple' 'apple' 'orange' 'apple' 'apple' 'orange' 'orange'
 'orange' 'orange' 'apple' 'apple' 'orange' 'apple' 'apple' 'apple']
[[65.    4.09]
 [66.    4.68]
 [72.    5.85]
 [70.    4.83]
 [70.    4.22]
 [71.    5.26]
 [69.    4.61]
 [73.    5.03]] ['orange' 'orange' 'apple' 'orange' 'orange' 'apple' 'orange'
'apple']
```

[23]:
```
classifier = SVC(kernel='rbf',random_state=1,C=1,gamma='auto')
classifier.fit(x_train,y_train)
```

[23]:
```
SVC(C=1, gamma='auto', random_state=1)
```

[24]:
```
y_pred = classifier.predict(x_test)
print(y_pred)
```

```
['orange' 'orange' 'apple' 'apple' 'orange' 'apple' 'orange' 'apple']
```

[25]:
```
cm = confusion_matrix(y_test,y_pred)
print(cm)
accuracy = float(cm.diagonal().sum())/len(y_test)
```

```
print('model accuracy is:',accuracy*100,'%')
```

```
[[3 0]
 [1 4]]
model accuracy is: 87.5 %
```

```
[ ]:
```