Exercise 5

Use SKLearn Iris flower dataset to train your model using Logistic Regression. You need to figure out accuracy of your model and use that to predict different samples in your test dataset. In iris dataset there are 150 samples containing following features

1. Sepal length
2. Sepal width
3. Petal length
4. Petal width

Use above 4 features and classify the flower in one of the three categories

1. Setosa
2. Versicolour
3. Virginica

Importing all libraries

In [15]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

Loading Dataset

In [16]:
```
data = load_iris()
```

In [17]:
```
dir(data)
```

Out[17]:
```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

Checking data at index 0

In [25]:
```
data.data[0]
```

Out[25]:
```
array([5.1, 3.5, 1.4, 0.2])
```

Train, Test Splitting

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.2)
```

Creating Logistic Regression model

In [37]:
```
model = LogisticRegression()
```

Checking data at index 50

In [53]:
```
data.data[50]
```

Out[53]:
```
array([7. , 3.2, 4.7, 1.4])
```

Creating dataframe out of the data and adding class column that contains target value (Not necessary)

In [46]:
```
df = pd.DataFrame(data.data)
df['Class'] = data.target
df.head()
```

Out[46]:

	0	1	2	3	Class
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

Training the model

In [43]:
```
model.fit(X_train, y_train)
```

Out[43]:
```
▾ LogisticRegression
LogisticRegression()
```

Checking accuracy score

In [45]:
```
model.score(X_test, y_test)
```

Out[45]:
```
0.9
```

Predicting a specific value which i copied from index 50

In [54]:
```
model.predict([[7. , 3.2, 4.7, 1.4]])
```

Out[54]:
```
array([1])
```

Predicting Test dataset and printing the actual and predicted value side by side then displaying how many wrong prediction we got

In [70]:
```
predict = model.predict(X_test)
wrong_count = 0
for i in range(len(predict)):
    print(f"Actual {y_test[i]}, Predicted {predict[i]}")
    if y_test[i] != predict[i]:
        wrong_count += 1
print("Wrong prediction count :", wrong_count)
```

```
Actual 0, Predicted 0
Actual 2, Predicted 1
Actual 2, Predicted 2
Actual 1, Predicted 2
Actual 1, Predicted 1
Actual 2, Predicted 2
Actual 2, Predicted 2
Actual 1, Predicted 1
Actual 2, Predicted 2
Actual 2, Predicted 2
Actual 2, Predicted 2
Actual 1, Predicted 1
Actual 1, Predicted 1
Actual 2, Predicted 2
Actual 1, Predicted 1
Actual 1, Predicted 1
Actual 2, Predicted 1
Actual 0, Predicted 0
Actual 2, Predicted 2
Actual 0, Predicted 0
Actual 1, Predicted 1
Actual 0, Predicted 0
Actual 1, Predicted 1
Actual 0, Predicted 0
Actual 1, Predicted 1
Actual 2, Predicted 2
Actual 0, Predicted 0
Actual 0, Predicted 0
Actual 0, Predicted 0
Actual 0, Predicted 0
Wrong prediction count : 3
```

As you can see above, the model got 3 wrong.

1. Actual value is 2, but predicted 1

2. Actual value is 1, but predicted 2
3. Actual value is 2, but predicted 1

Now, using the confusion matrix plot, you can see that 2 times actual value was 2, but model predicted one, and 1 time actual value was 1, and model predicted 2

Using Confusion Matrix to check where our model didnt perform well

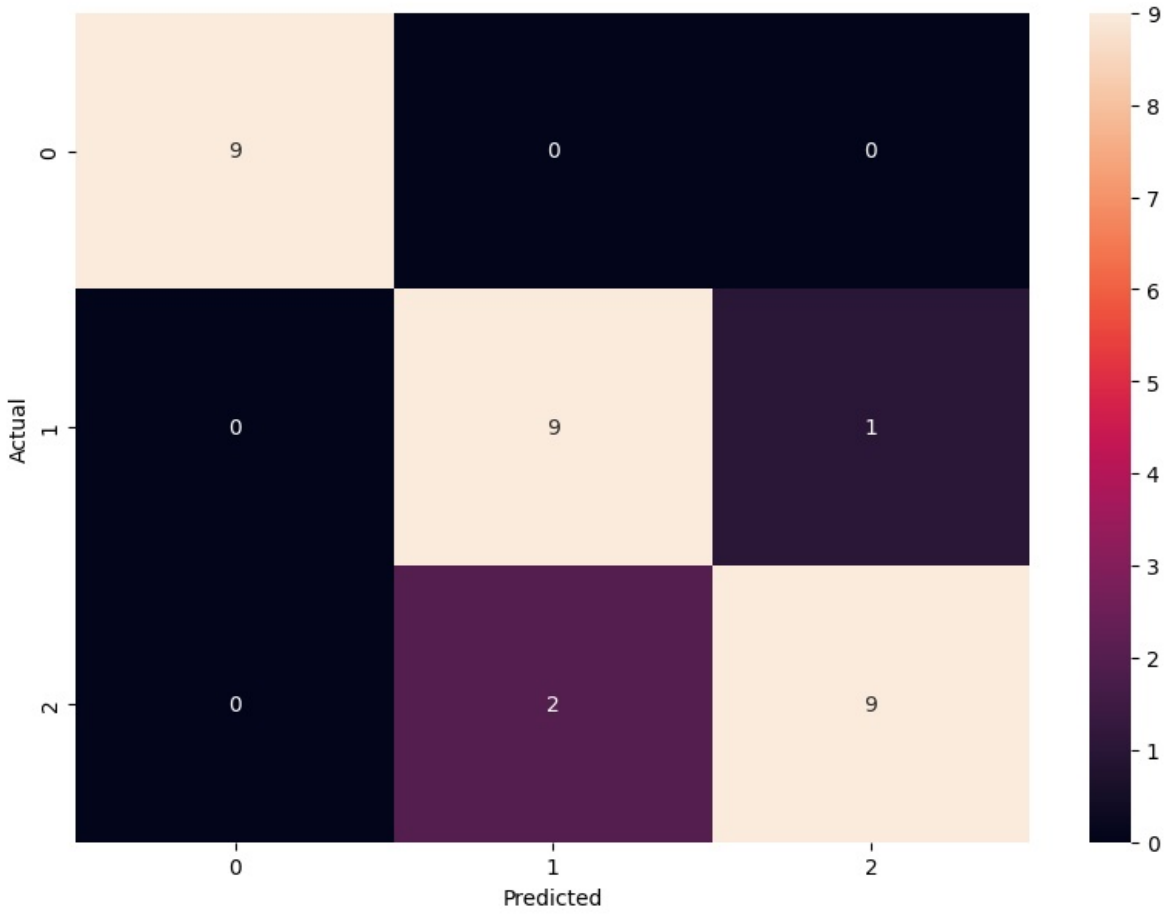
In [75]:
```
from sklearn.metrics import confusion_matrix
```

In [76]:
```
cm = confusion_matrix(y_test, predict)
```

Plotting the confusion matrix

In [77]:
```
plt.figure(figsize=(10, 7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

Out[77]:
```
Text(95.72222222222221, 0.5, 'Actual')
```



In [ ]:
```

```