Logistic Regression using Iris Dataset

1. Import Libraries :

In [1]:
```
# Import imp libraries
import pandas as pd, numpy as np
import matplotlib.pyplot as plt, seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

2. Import Dataset :

In [2]:
```
# Load the data set

df = sns.load_dataset('iris')
```

In [3]:
```
df
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [4]:
```
#
df.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
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

3. Pre-processing :

In [5]:
```
df.isna()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [6]:
```
df.isna().sum()
```

Out[6]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [7]:
```
# Check info of the null and non-null values
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   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [8]:
```
df.species.unique()
```

Out[8]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [9]:
```
df.species.value_counts()
```

Out[9]:
```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

In [10]:
```
df.species.values
```

Out[10]:
```
array(['setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica'], dtype=object)
```

In [11]:
```
# x = df.drop(df['species']=='setosa', axis=0)
# We want the data in binary classification format
# So, remove one of the class
df = df[df['species'] != 'setosa']
```

In [12]:
```
df.info()
# Now, only 100 rows and all 5 cols are apresent
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 50 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  100 non-null    float64
 1   sepal_width   100 non-null    float64
 2   petal_length  100 non-null    float64
 3   petal_width   100 non-null    float64
 4   species       100 non-null    object 
dtypes: float64(4), object(1)
memory usage: 4.7+ KB
```

In [13]:
```
df
```

Out[13]:

	sepal_length	sepal_width	petal_length	petal_width	species
50	7.0	3.2	4.7	1.4	versicolor
51	6.4	3.2	4.5	1.5	versicolor
52	6.9	3.1	4.9	1.5	versicolor
53	5.5	2.3	4.0	1.3	versicolor
54	6.5	2.8	4.6	1.5	versicolor
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

100 rows × 5 columns

In [14]:
```
# Now, replace the string values with the integers
df['species'].replace({'versicolor':0, 'virginica':1}, inplace=True)
```

4. Splitting the Dataset :

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
# Divide the dataFrame in independent(input) and dependent(output) variables

x = df.drop('species', axis=1)
y = df['species']
```

In [17]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_sta
```

5. Training the Model :

In [18]:
```
from sklearn.linear_model import LogisticRegression
```

In [19]:
```
cls = LogisticRegression()
cls.fit(x_train, y_train)
```

Out[19]:
```
LogisticRegression
LogisticRegression()
```
(https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.LogisticR

6. Result Prediction :

In [20]:
```
y_pred = cls.predict(x_test)
```

7. Model Evaluation :

In [21]:
```
cls.score(x_test, y_test)
```

Out[21]:
```
0.95
```

In [22]:
```
from sklearn.metrics import accuracy_score, classification_report
```

In [23]:
```
score = accuracy_score(y_pred, y_test)
score
```

Out[23]:
```
0.95
```

In [24]:
```
rp = classification_report(y_pred, y_test)
```

In [25]:
```
print(rp)
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        11
           1       1.00      0.89      0.94         9

    accuracy                           0.95        20
   macro avg       0.96      0.94      0.95        20
weighted avg       0.95      0.95      0.95        20
```

- **AUC-ROC Curve :**

In [26]:
```
# Plotting of AUC-ROC

from sklearn.metrics import roc_curve, auc
```

In [27]:
```
fpr, tpr, threshold = roc_curve(y_test, cls.predict_proba(x_test)[:,1])
```

In [28]:
```
fpr
```

Out[28]:
```
array([0.        , 0.        , 0.        , 0.08333333, 0.08333333,
       1.        ])
```

In [29]:
```
tpr
```

Out[29]:
```
array([0.   , 0.125, 0.5  , 0.5  , 1.   , 1.   ])
```

In [30]:
```
threshold
```

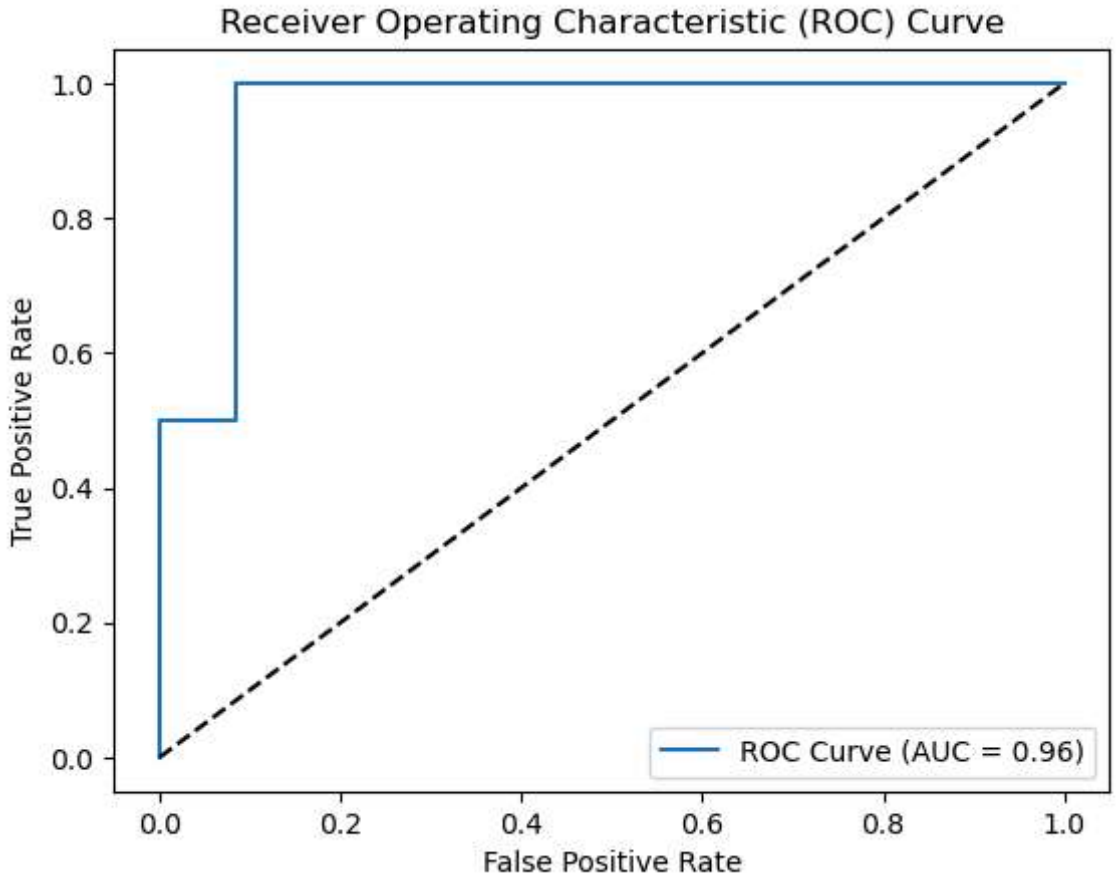
Out[30]:
```
array([       inf, 0.97308082, 0.90635673, 0.66125877, 0.52799486,
       0.01327977])
```

In [31]:
```
area = auc(fpr, tpr)
```

In [32]:
```
print('AUC', area)
```

```
AUC 0.9583333333333333
```

In [33]:
```
plt.plot(fpr, tpr, label='ROC Curve (AUC = {:.2f})'.format(area))
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```



In [ ]: