Aim:

Download IRIS dataset from UCI Repository The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

1. Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary.
2. Perform data-preparation (Train-Test Split)
3. Apply Logistic Regression Algorithm
4. Evaluate Model.

Importing necessary libraries

In [46]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Reading the dataset

In [24]:

```
A = pd.read_csv(r"C:\Users\RASIKA\Downloads\archive (8)\Iris.csv")
```

In [25]:

```
A.shape
```

Out[25]:

```
(150, 6)
```

Handling categorical values

In [26]:

```
A["Species"] = A["Species"].map({"Iris-setosa": 0, "Iris-versicolor": 1, "Iris-virginica":
```

Printing first five and last five records from the dataset

In [27]:

```
A.head(5)
```

Out[27]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

Type *Markdown* and LaTeX: α^2

In [28]:

```
A.tail(5)
```

Out[28]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	2
146	147	6.3	2.5	5.0	1.9	2
147	148	6.5	3.0	5.2	2.0	2
148	149	6.2	3.4	5.4	2.3	2
149	150	5.9	3.0	5.1	1.8	2

Printing the datatype of each attribute

In [29]:

```
A.dtypes
```

Out[29]:

```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species            int64
dtype: object
```

Setting the dependent and independent variables

In [30]:

```
X = A[["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm"]]
```

In [31]:

```
Y = A[["Species"]]
```

Splitting the dataset into training data and testing data

In [32]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
```

Training the logistic regression model

In [33]:

```
from sklearn.linear_model import LogisticRegression
```

In [34]:

```
reg = LogisticRegression()
```

In [35]:

```
model = reg.fit(X_train, Y_train)
```

```
C:\Users\RASIKA\AppData\Local\Programs\Python\Python310\lib\site-packages\sk
learn\utils\validation.py:1111: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to (n_samp
les, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\RASIKA\AppData\Local\Programs\Python\Python310\lib\site-packages\sk
learn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

In [36]:

```
Y_predict = model.predict(X_test)
```

In [37]:

```
print(Y_predict)
```

```
[0 0 1 1 1 0 0 2 1 1 2 0 2 0 1 0 2 2 0 1 0 1 0 2 0 2 0 2 2 0 1 0 0 2 2 1 0
 2 2 1 1 2 0 1 1]
```

Printing the confusion matrix

In [38]:

```
from sklearn.metrics import confusion_matrix
```

In [39]:

```
cm = confusion_matrix(Y_test, Y_predict)
```

In [40]:

```
print(cm)
```

```
[[17  0  0]
 [ 0 14  2]
 [ 0  0 12]]
```

Printing the classification report

In [41]:

```
from sklearn.metrics import classification_report
```

In [42]:

```
cr = classification_report(Y_test, Y_predict)
```

In [43]:

```
print(cr)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      0.88      0.93        16
           2       0.86      1.00      0.92        12

    accuracy                           0.96        45
   macro avg       0.95      0.96      0.95        45
weighted avg       0.96      0.96      0.96        45
```

Printing the accuracy

In [44]:

```
from sklearn.metrics import accuracy_score
acc = accuracy_score(Y_test, Y_predict)
print(acc)
```

```
0.9555555555555556
```