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

```
df=pd.read_csv('Iris.csv')
```

```
df.shape
```

```
(150, 6)
```

```
df.head()
```

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

```
df.drop('Id',axis=1,inplace=True)
```

```
df.head()
```

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

```
df['Species'].value_counts()
```

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

```
df.info()
```

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

```
df.describe(include='all')
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-setosa

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
df['Species']=le.fit_transform(df['Species'])
```

min	4.300000	2.000000	1.000000	0.100000	NaN

```
df.head()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

```
df.Species.value_counts()
```

```
0    50
1    50
2    50
Name: Species, dtype: int64
```

```
cols=df.columns[:-1]
```

```
for i in cols:
  sns.boxplot(df[i])
  plt.title(i)
  plt.show()
```

```
#Outlier removal using IQR Method
def outliers(data,feature):
  q1=data[feature].quantile(0.25)
  q3=data[feature].quantile(0.75)
  iqr=q3-q1
  ul=q3+1.5*iqr
  ll=q1-1.5*iqr
  return ul,ll
```

```
for i in cols:
  ul,ll=outliers(df,i)
  df=df[(df[i]<ul) & (df[i]>ll)]
```

```
df.shape
```

```
(146, 5)
```

```
X=df.drop('Species',axis=1)
y=df['Species']
```

```
from sklearn.model_selection import train_test_split as tts
```

```
X_train,X_test,y_train,y_test=tts(X,y,test_size=0.3)
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler=StandardScaler()
scaler.fit(X_train)
X_train=scaler.transform(X_train)
X_test=scaler.transform(X_test)
```

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

```
lr=LogisticRegression()
lr.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

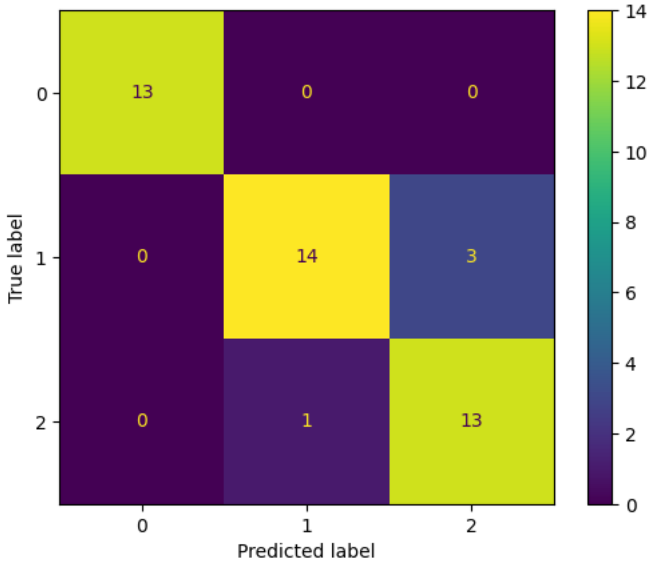
```
y_pred=lr.predict(X_test)
```

```
from sklearn.metrics import accuracy_score,f1_score,precision_score,recall_score,confusion_matrix,ConfusionMatrixDisplay
```

```
cm=confusion_matrix(y_test,y_pred)
```

```
ConfusionMatrixDisplay(cm).plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fe6306bbcd0>
```



```
accuracy_score(y_test,y_pred)
```

```
0.9090909090909091
```

```
f1_score(y_test,y_pred,average='macro')
```

```
0.9138888888888889
```