In [235]:
```
#importing necessary libraries
import pandas as pd
from sklearn import model_selection
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn import metrics
```

In [236]:
```
#Loading the dataset

data_frame = pd.read_csv('Iris.csv')
#printing dataset
data_frame.head()
```

Out[236]:

	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

In [237]:
```
#dropping the unnecessary column id

data_frame=data_frame.drop(columns =['Id'],axis=1)
data_frame.head()
```

Out[237]:

	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

In [238]:
```
# displaing statistics of our dataset
data_frame.describe()
```

Out[238]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [239]:
```
#displaying basic info about datatype
data_frame.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
```

In [240]:
```
#displaying number of sample on each class
data_frame['Species'].value_counts()
```

Out[240]:
```
Iris-setosa        50
Iris-virginica     50
Iris-versicolor    50
Name: Species, dtype: int64
```

from above code we can see that we have 50 samples of Iris-setosa ,50 samples of Iris-virginica and 50 samples of Iris-versicolor

Preprocessing the dataset

In [241]:
```
#Check the null values

data_frame.isnull().sum()
```

Out[241]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

from above code we can see that there are no any null values in our dataset

In [242]:
```
#converting species column into numerical using label encoder

from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data_frame['Species']=label_encoder.fit_transform(data_frame['Species'])
data_frame.head(100)
```

Out[242]:

	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
...	...	...	...	...	...
95	5.7	3.0	4.2	1.2	1
96	5.7	2.9	4.2	1.3	1
97	6.2	2.9	4.3	1.3	1
98	5.1	2.5	3.0	1.1	1
99	5.7	2.8	4.1	1.3	1

100 rows × 5 columns

In [243]:
```
# taking x as our input and y as target
X=data_frame.iloc[0:,:4]
X.head()
```

Out[243]:

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

In [244]:
```
y=data_frame.iloc[0:,-1]
y.head()
```

Out[244]:
```
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int64
```

Training our Model

In [245]:
```
from sklearn.model_selection import train_test_split

#train - 70%
# test -30


X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size= 0.3, random_state = 1)
```

In [246]:
```
# importing logistic regression from sklearn.linear model and applying logistic regression algorithm for our mylti class model

model = linear_model.LogisticRegression(multi_class='ovr',solver='liblinear')
model.fit(X_train, y_train)
```

Out[246]:
```
LogisticRegression(multi_class='ovr', solver='liblinear')
```

since we arre using multi class classification ie. Ove Vs Rest algorithm we set multi_class ='ovr' and we must use liblinear solver with it

In [247]:
```
model.predict(X_test)
```

Out[247]:
```
array([0, 1, 1, 0, 2, 2, 2, 0, 0, 2, 1, 0, 2, 1, 2, 0, 1, 2, 0, 0, 1, 2,
       2, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 2, 2,
       1])
```

In [248]:
```
print("Accuracy= ",model.score(X_test,y_test)*100)
```

```
Accuracy=  88.88888888888889
```

from above code we can see that our model accuracy is 88.88 percent which is a good score

In [249]:
```
predict=model.predict(X_test)
predict
```

Out[249]:
```
array([0, 1, 1, 0, 2, 2, 2, 0, 0, 2, 1, 0, 2, 1, 2, 0, 1, 2, 0, 0, 1, 2,
       2, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 2, 2,
       1])
```

In [250]:
```
from sklearn.metrics import confusion_matrix
confusion_metrics = confusion_matrix(y_test,predict)
confusion_metrics
```

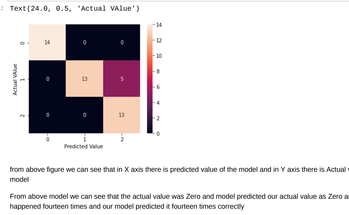
Out[250]:
```
array([[14,  0,  0],
       [ 0, 13,  5],
       [ 0,  0, 13]])
```

In [251]:
```
#In order to explain the confusion matrix better we are going to use seaborn plot
import seaborn as sn

%matplotlib inline

plt.figure(figsize=(5,4))
sn.heatmap(confusion_metrics,annot=True)
plt.xlabel('Predicted Value ')
plt.ylabel('Actual VAlue')
```

Out[251]:
```
Text(24.0, 0.5, 'Actual VAlue')
```



from above figure we can see that in X axis there is predicted value of the model and in Y axis there is Actual value of the model

From above model we can see that the actual value was Zero and model predicted our actual value as Zero and it happened fourteen times and our model predicted it fourteen times correctly

likewise for One we can see the actual value was one and our model predicted thirteen times correctly

likewise for Two we can see that our model has predicted thirteen times correctly but our model predicted the value of Two as one five times when its actual value was two

In [252]:
```
print(metrics.classification_report(y_test, model.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      0.72      0.84        18
           2       0.72      1.00      0.84        13

    accuracy                           0.89        45
   macro avg       0.91      0.91      0.89        45
weighted avg       0.92      0.89      0.89        45
```

Classification report is used to measure the quality of prediction

precision: Indicates the correct classes classified by our model

recall: Indicates what proportion of actual positive was identified correctly

f1-score:This is a weighted average of the true positive rate (recall) and precision.