In [20]:

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

In [21]:

```
iris = pd.read_csv('C:\datasets\iris.csv')
```

In [22]:

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [23]:

```
iris.head()
```

Out[23]:

	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 [24]:

```
iris.tail()
```

Out[24]:

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

In [25]:

```
iris.count()
```

Out[25]:

```
Id               150
SepalLengthCm    150
SepalWidthCm     150
PetalLengthCm    150
PetalWidthCm     150
Species          150
dtype: int64
```

In [26]:

```
iris.shape
```

Out[26]:

```
(150, 6)
```

In [27]:

```
iris.columns
```

Out[27]:

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthC
m',
       'Species'],
      dtype='object')
```

In [28]:

```
iris.describe()
```

Out[28]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [29]:

```
iris.groupby('Species').size()
```

Out[29]:

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

In [30]:

```
iris.sample(10)
```

Out[30]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
46	47	5.1	3.8	1.6	0.2	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
119	120	6.0	2.2	5.0	1.5	Iris-virginica
136	137	6.3	3.4	5.6	2.4	Iris-virginica
23	24	5.1	3.3	1.7	0.5	Iris-setosa
34	35	4.9	3.1	1.5	0.1	Iris-setosa
100	101	6.3	3.3	6.0	2.5	Iris-virginica
84	85	5.4	3.0	4.5	1.5	Iris-versicolor
1	2	4.9	3.0	1.4	0.2	Iris-setosa
103	104	6.3	2.9	5.6	1.8	Iris-virginica

Removing the un-necessary column from the dataset

In [31]:

```
iris.drop("Id", axis=1, inplace = True)
```

In [32]:

```
iris.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 [33]:

```
iris.head()
```

Out[33]:

	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 [34]:

```
iris.isnull()
```

Out[34]:

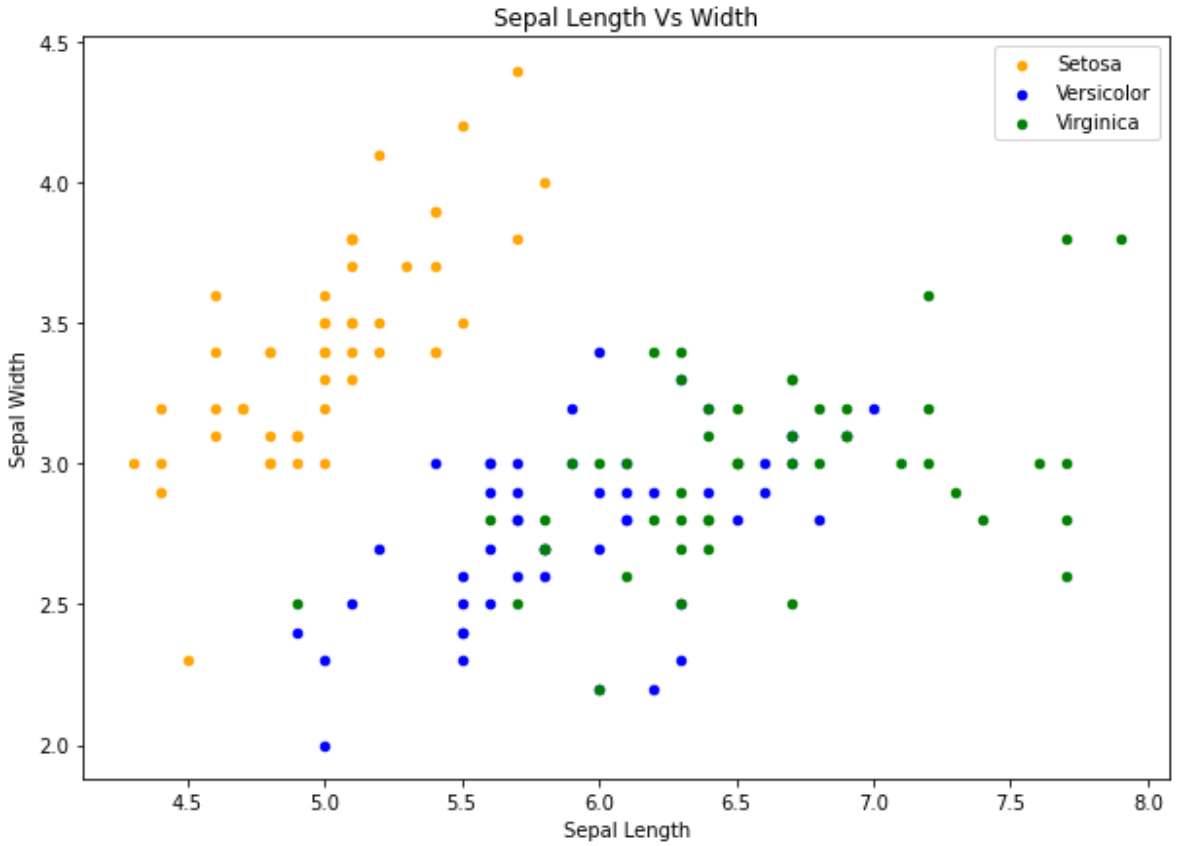
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

EDA Analysis

In [35]:

```
fig = iris[iris.Species == 'Iris-setosa'].plot(kind='scatter', x='SepalLengthCm', y='SepalW
iris[iris.Species == 'Iris-versicolor'].plot(kind='scatter', x='SepalLengthCm', y='SepalWid
iris[iris.Species == 'Iris-virginica'].plot(kind='scatter', x='SepalLengthCm', y='SepalWidt

fig.set_xlabel('Sepal Length')
fig.set_ylabel('Sepal Width')
fig.set_title('Sepal Length Vs Width')

fig=plt.gcf()
fig.set_size_inches(10, 7)
plt.show()
```



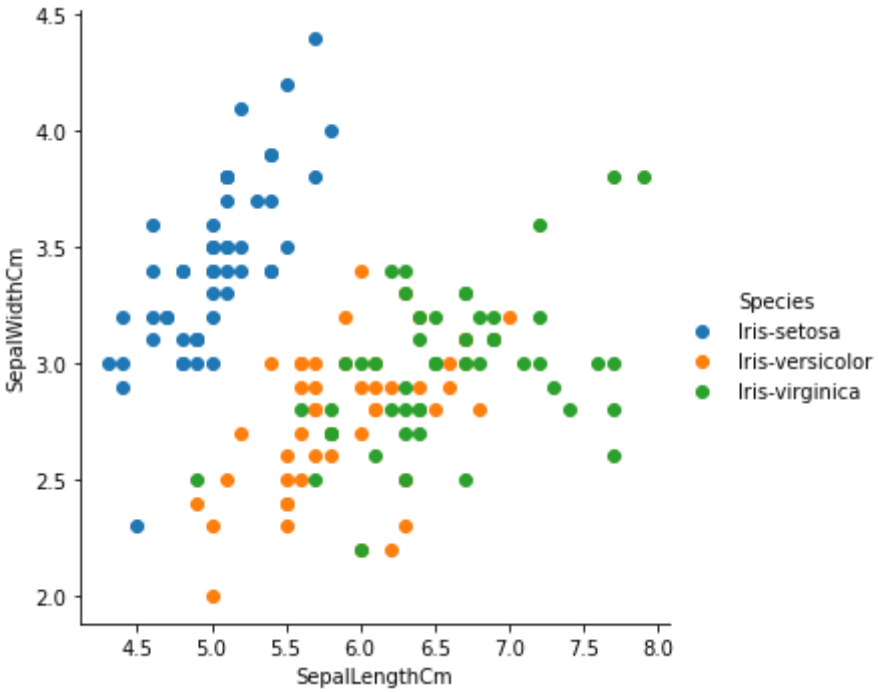
In [36]:

```
sns.FacetGrid(iris, hue='Species', size=5)\
   .map(plt.scatter, 'SepalLengthCm', 'SepalWidthCm')\
   .add_legend()
```

```
C:\ProgramData\Anaconda\lib\site-packages\seaborn\axisgrid.py:316: UserWarni
ng: The `size` parameter has been renamed to `height`; please update your co
de.
  warnings.warn(msg, UserWarning)
```

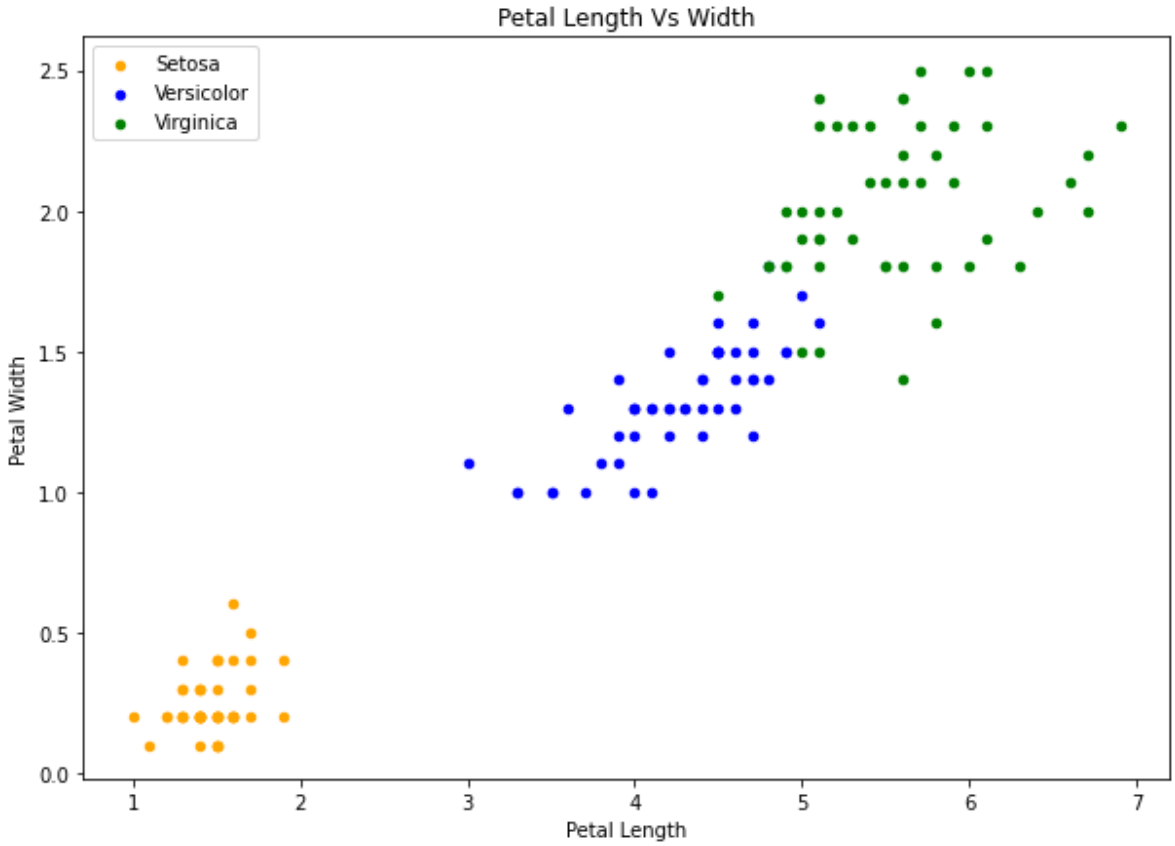
Out[36]:

```
<seaborn.axisgrid.FacetGrid at 0x29995f39700>
```



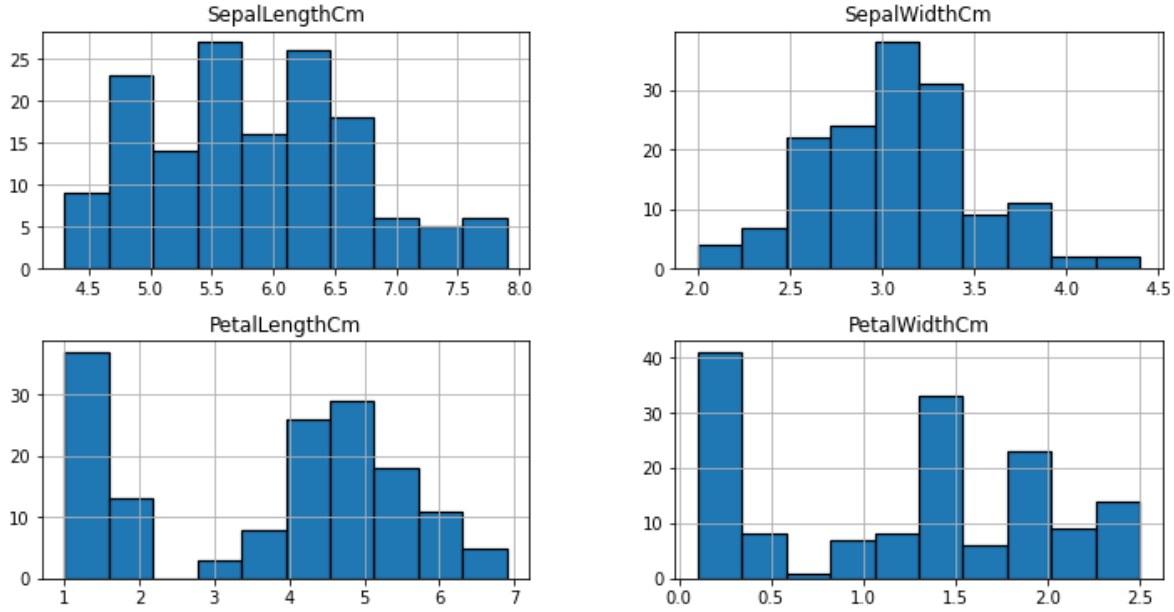
In [37]:

```
fig = iris[iris.Species == 'Iris-setosa'].plot(kind='scatter', x='PetalLengthCm', y='PetalW
iris[iris.Species == 'Iris-versicolor'].plot(kind='scatter', x='PetalLengthCm', y='PetalWid
iris[iris.Species == 'Iris-virginica'].plot(kind='scatter', x='PetalLengthCm', y='PetalWidt

fig.set_xlabel('Petal Length')
fig.set_ylabel('Petal Width')
fig.set_title('Petal Length Vs Width')

fig=plt.gcf()
fig.set_size_inches(10, 7)
plt.show()
```



In [38]:

```
iris.hist(edgecolor='black', linewidth=1.2)
fig = plt.gcf()
fig.set_size_inches(12,6)
plt.show()
```

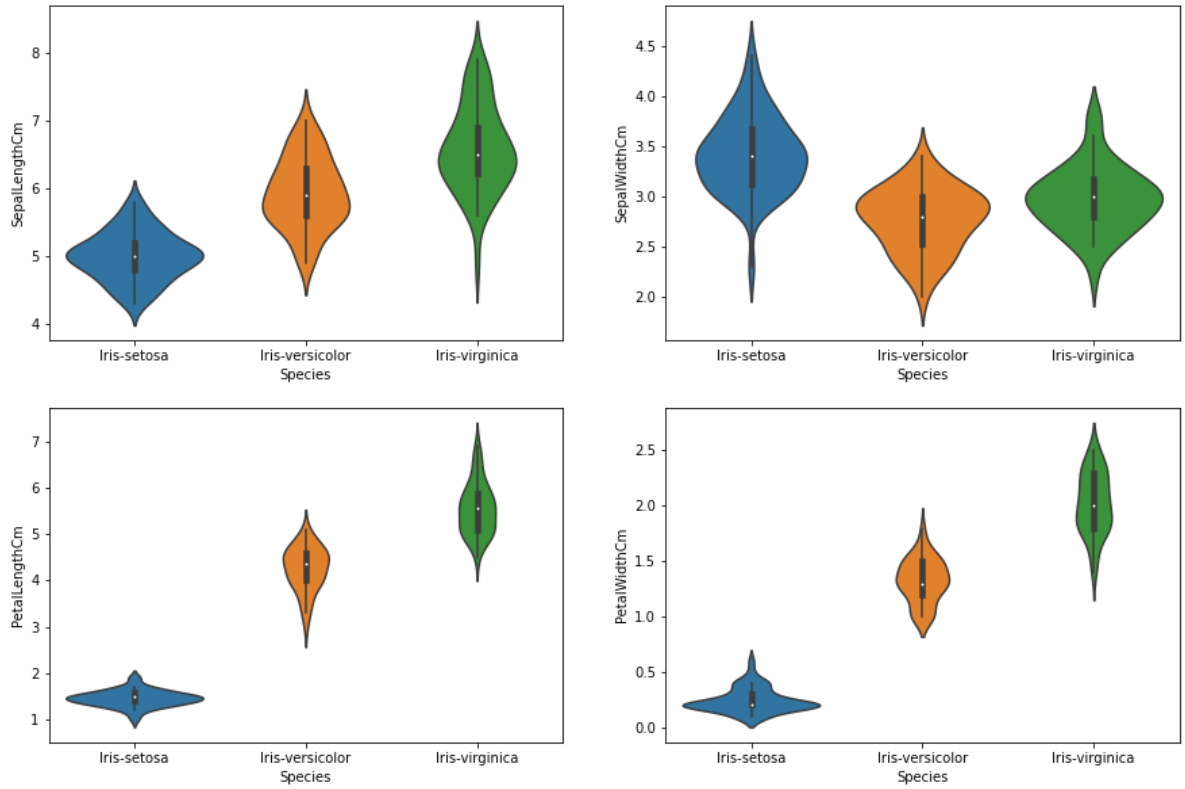


In [39]:

```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species', y = 'SepalLengthCm', data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='Species', y = 'SepalWidthCm', data=iris)

plt.subplot(2,2,3)
sns.violinplot(x='Species', y = 'PetalLengthCm', data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='Species', y = 'PetalWidthCm', data=iris)
```

Out[39]:

```
<AxesSubplot:xlabel='Species', ylabel='PetalWidthCm'>
```



In [40]:

```
from sklearn import svm # for SVM Algorithm
from sklearn.linear_model import LogisticRegression # for Logistic Regression Algorithm
from sklearn.model_selection import train_test_split # to split the dataset for training an
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [41]:

```
train, test = train_test_split(iris, test_size=0.3) # our main data split into train and te
# the attribute test_size=0.3 splits the data into 70% and 30% ratio. train=70% and test=30
print(train.shape)
print(test.shape)
```

```
(105, 5)
(45, 5)
```

In [42]:

```
train_X = train[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']] # taking t
train_y = train.Species # output of the training data

test_X = test[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']] # taking tes
test_y = test.Species # output value of the test data
```

In [43]:

```
train_X.head()
```

Out[43]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
74	6.4	2.9	4.3	1.3
28	5.2	3.4	1.4	0.2
147	6.5	3.0	5.2	2.0
94	5.6	2.7	4.2	1.3
34	4.9	3.1	1.5	0.1

In [44]:

```
test_X.head()
```

Out[44]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
125	7.2	3.2	6.0	1.8
29	4.7	3.2	1.6	0.2
7	5.0	3.4	1.5	0.2
20	5.4	3.4	1.7	0.2
57	4.9	2.4	3.3	1.0

In [45]:

```
train_y.head()
```

Out[45]:

```
74     Iris-versicolor
28         Iris-setosa
147     Iris-virginica
94     Iris-versicolor
34         Iris-setosa
Name: Species, dtype: object
```

Decision Tree

In [48]:

```
model = DecisionTreeClassifier()
model.fit(train_X, train_y)
prediction = model.predict(test_X)
print('The accuracy of Decision Tree is: ', metrics.accuracy_score(prediction, test_y))
print('The confusion matrix for decision tree is: ',metrics.confusion_matrix(prediction,tes
print('The classification matrix for decision tree is: ',metrics.classification_report(test
```

```
The accuracy of Decision Tree is:  0.9777777777777777
The confusion matrix for decision tree is:  [[19  0  0]
 [ 0 14  0]
 [ 0  1 11]]
The classification matrix for decision tree is:                   precision
recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor       1.00      0.93      0.97        15
 Iris-virginica       0.92      1.00      0.96        11

       accuracy                           0.98        45
      macro avg       0.97      0.98      0.97        45
   weighted avg       0.98      0.98      0.98        45
```

Logistic Regression

In [47]:

```
model = LogisticRegression()
model.fit(train_X, train_y)
prediction = model.predict(test_X)
print('The accuracy of Decision Tree is: ', metrics.accuracy_score(prediction, test_y))
print('The confusion matrix for decision tree is: ',metrics.confusion_matrix(prediction,tes
print('The classification matrix for decision tree is: ',metrics.classification_report(test
```

```
The accuracy of Decision Tree is:  0.9777777777777777
The confusion matrix for decision tree is:  [[19  0  0]
 [ 0 14  0]
 [ 0  1 11]]
The classification matrix for decision tree is:                   precision
recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor       1.00      0.93      0.97        15
 Iris-virginica       0.92      1.00      0.96        11

       accuracy                           0.98        45
      macro avg       0.97      0.98      0.97        45
   weighted avg       0.98      0.98      0.98        45
```

In [ ]: