In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC #import Support vector classifier
from sklearn.model_selection import train_test_split #split train test data
```

In [8]:
```
#read the iris data
df = pd.read_csv('IRIS.csv')
df.head()
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width	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 [9]:
```
#check for null values
df.isnull().sum()
```

Out[9]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [12]:
```
#label encoding
from sklearn.preprocessing import LabelEncoder
labels = ['species']
for label in labels:
    label_encoder = LabelEncoder()
    label_encoder.fit(df[label])
df[label] = label_encoder.transform(df[label])
df.head()
```

Out[12]:

	sepal_length	sepal_width	petal_length	petal_width	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

In [13]:
```
print(df['species'])
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: species, Length: 150, dtype: int64
```

In [15]:

```
# separate independent and dependent variable
x = df.iloc[:,0:3] # sepal length-width, petal length-width
y = df.iloc[:,-1:] #species
```

In [16]:

```
y.head()
```

Out[16]:

	species
0	0
1	0
2	0
3	0
4	0

In [17]:

```
#split the data into train and test set
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, random_sta
X_train.head()
```

Out[17]:

	sepal_length	sepal_width	
60	5.0	2.0	3.5
116	6.5	3.0	5.5
144	6.7	3.3	5.7
119	6.0	2.2	5.0
108	6.7	2.5	5.8

In [18]:

```
y_train.head()
```

Out[18]:

	species
60	1
116	2
144	2
119	2
108	2

In [19]:
```
from sklearn.svm import LinearSVC
clf = LinearSVC() #by default LinearSVC follows one -vs- rest approach.
#train the alorithm or model
clf.fit(X_train,y_train)
#test the model
yp = clf.predict(X_test)
print (yp)
print ("accuracy is ", clf.score(X_test,y_test)*100)
```

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

In [20]:
```
from sklearn import svm
#SVC that follows one-vs-one approch
clf = svm.SVC(decision_function_shape='ovo')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(yp)
print("accuracy is ", clf.score(X_test,y_pred)*100)
```

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

In [ ]: