In [6]:

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

In [2]:

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

In [3]:

```
df.head(5)
```

Out[3]:

	sepallength	sepalwidth	petallength	petalwidth	class
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 [4]:

```
for col in df.columns:
    print(col)
```

```
sepallength
sepalwidth
petallength
petalwidth
class
```

In [5]:

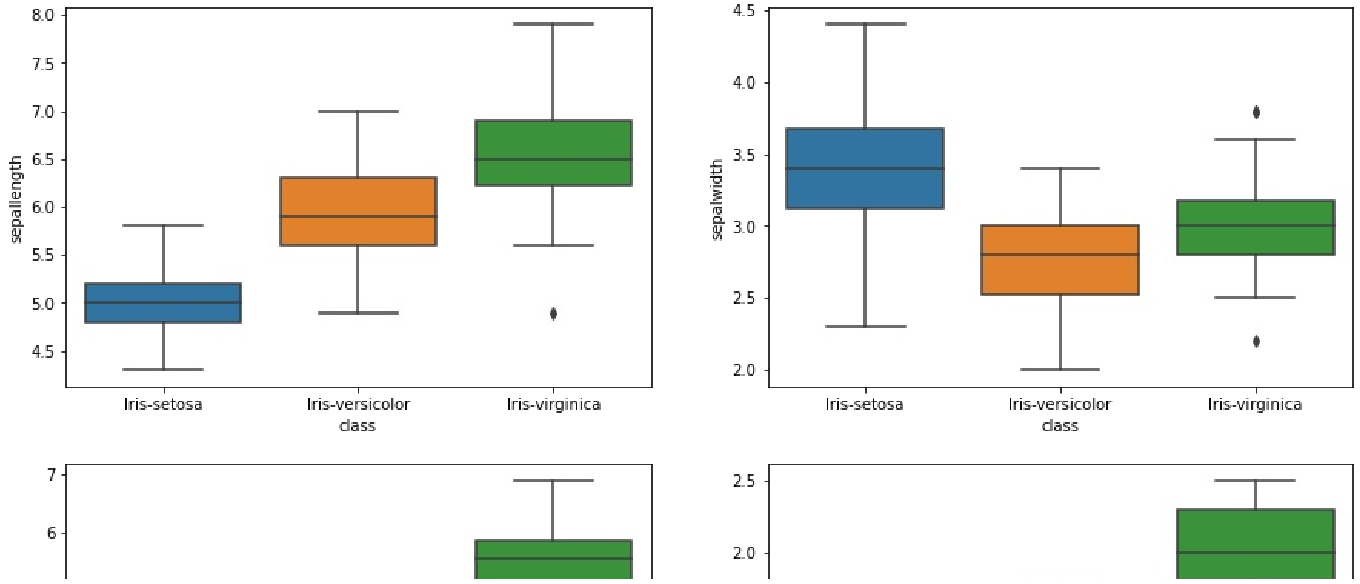
```
df.groupby('class').size()
```

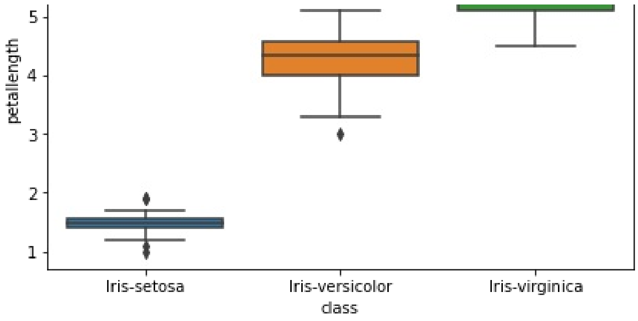
Out[5]:

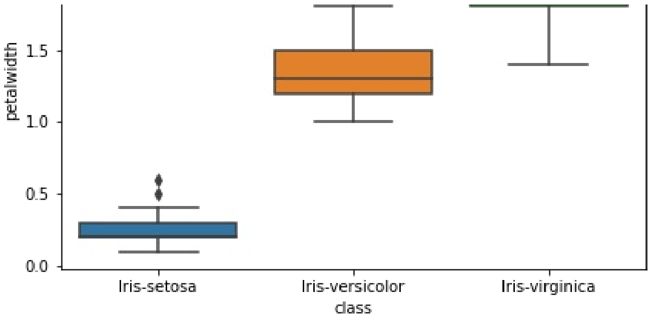
```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

In [8]:

```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.boxplot(x='class',y='sepallength',data=df)
plt.subplot(2,2,2)
sns.boxplot(x='class',y='sepalwidth',data=df)
plt.subplot(2,2,3)
sns.boxplot(x='class',y='petallength',data=df)
plt.subplot(2,2,4)
sns.boxplot(x='class',y='petalwidth',data=df)
plt.show()
```







In [9]:

```
df.isnull().values.any()
```

Out[9]:

```
False
```

In [10]:

```
df.info()
```

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

In [11]:

```
df.describe()
```

Out[11]:

	sepallength	sepalwidth	petallength	petalwidth
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 [12]:

```
from sklearn.model_selection import train_test_split
```

In [13]:

```
array = df.values
X = array[:,0:4]
y = array[:,4]
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

In [15]:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

In [16]:

```
svc = SVC(max_iter=1000,gamma='auto')
svc.fit(x_train,y_train)
y_pred = svc.predict(x_test)
acc_svc = round(accuracy_score(y_pred,y_test),2)*100
print("Accuracy :",acc_svc)
```

```
Accuracy : 98.0
```

In [17]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [18]:

```
decisiontree = DecisionTreeClassifier(random_state=0)
decisiontree.fit(x_train,y_train)
y_pred = decisiontree.predict(x_test)
acc_decisiontree = round(accuracy_score(y_pred,y_test),2)*100
print("Accuracy :",acc_decisiontree)
```

```
Accuracy : 98.0
```

In [19]:

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(max_iter=1000)
logreg.fit(x_train,y_train)
y_pred = logreg.predict(x_test)
acc_logreg = round(accuracy_score(y_pred,y_test),2)*100
print("Accuracy :",acc_logreg)
```

```
Accuracy : 98.0
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js