From UCI Machine Learning Repository

In [ ]:

```
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
plt.style.use('ggplot')
```

Select the dataset

Read data from archive.

Look at the data and try to understand if:

1. if it is a `csv` file or other
2. for `csv`, what is the *separator* character (`,` , `;` , `\t` , ...)
3. for `csv`, is there a *header*? it is a first row containing column names
4. if there is no header, look for reasonable names, e.g. for *UCI* a `.names` file
5. if there is no header, look at the documentation of `read_csv` to see how to specify column names
6. try to understand if the dataset is supervised, and what is the *target Class*

The download url is https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Use the read_csv() method of pandas dataframe https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

Use `df` as the dataframe name

Assign column names if necessary

In [ ]:

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
url= './iris.csv'
# adjust the line below, if necessary
df = pd.read_csv(url, names=["Sepal Length", "Sepal Width", "Petal Length", "Petal
```

Show column names

Use the `columns` attribute of pandas on `df`

In [ ]:

```
df.columns
```

Out[ ]:

```
Index(['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width', 'Class'], dty
pe='object')
```

Show portion of data

Use the `head` method of pandas dataframe

In [ ]:

```
df.head()
```

Out[ ]:

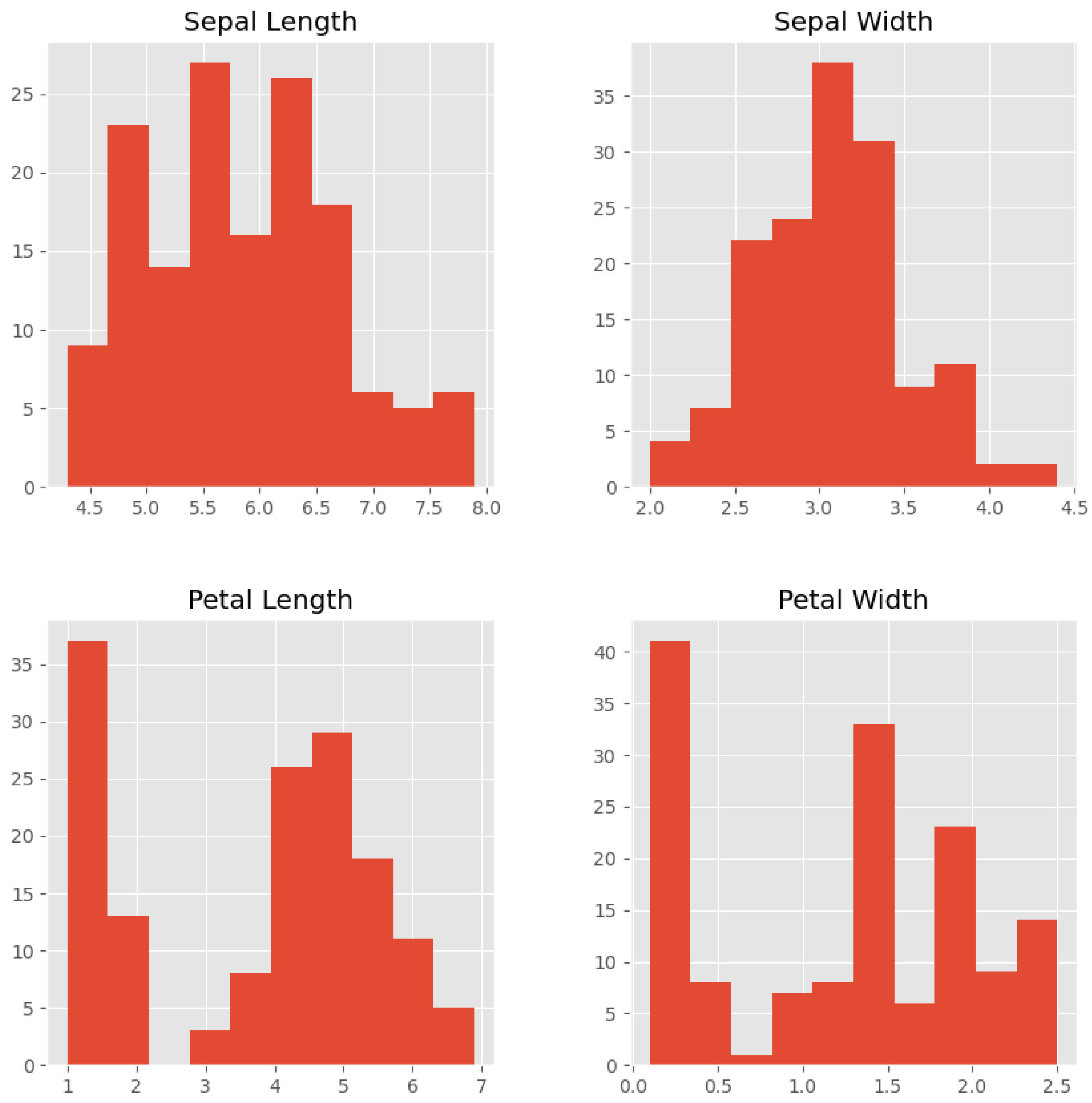
	Sepal Length	Sepal Width	Petal Length	Petal Width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Show histograms for all numeric values

Use the `DataFrame.hist` method of Pandas. You can set the `figsize` parameter to adjust size

In [ ]:

```
pd.DataFrame.hist(df
                  , figsize = [10,10]
                 );
```



Is there anything to observe? balanced distributions? skewed distributions? outliers?

Show synthetic description

The **describe** method of pandas dataframes gives a short summary

Examine in the documentation if there are interesting options in the method

In [ ]:

```
df.describe()
```

Out[ ]:

	Sepal Length	Sepal Width	Petal Length	Petal Width
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

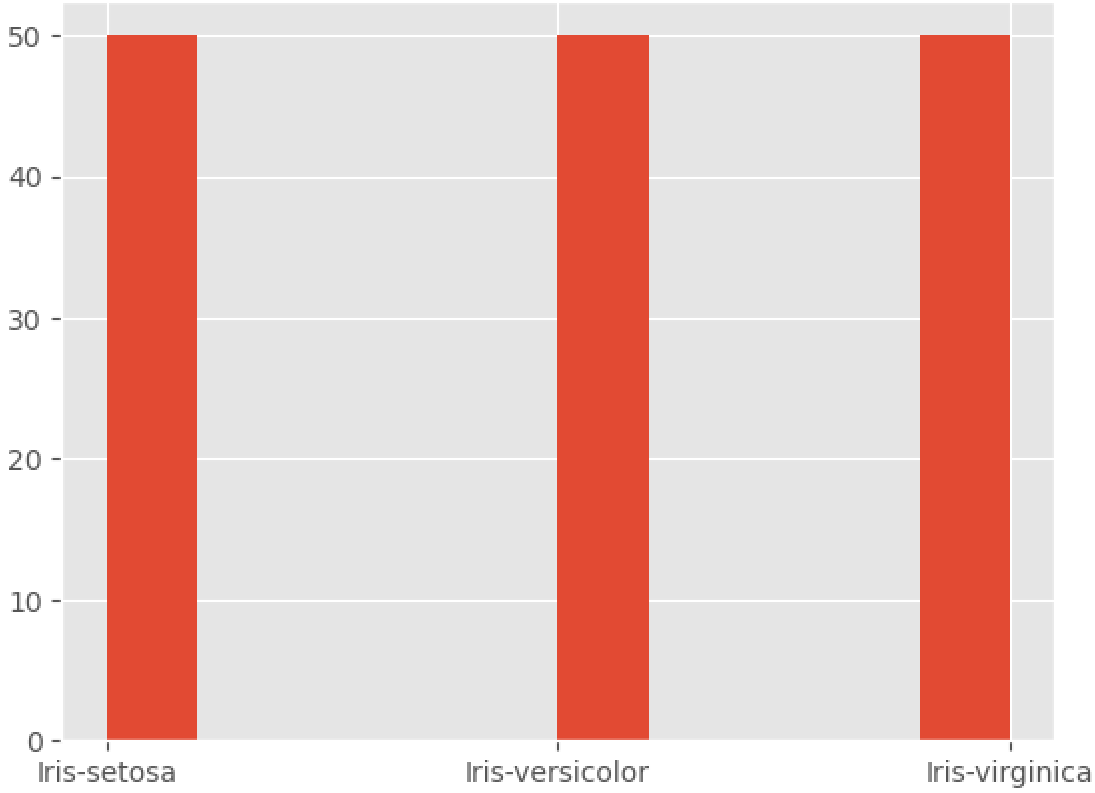
Are there *missing values*? How could we see it from the description?

Plot an histogram for "the target column"

Use the `hist` method of `matplotlib.pyplot` applied to the target column of `df`

In [ ]:

```
### adjust the line below
plt.hist(df['Class'])
plt.show()
```

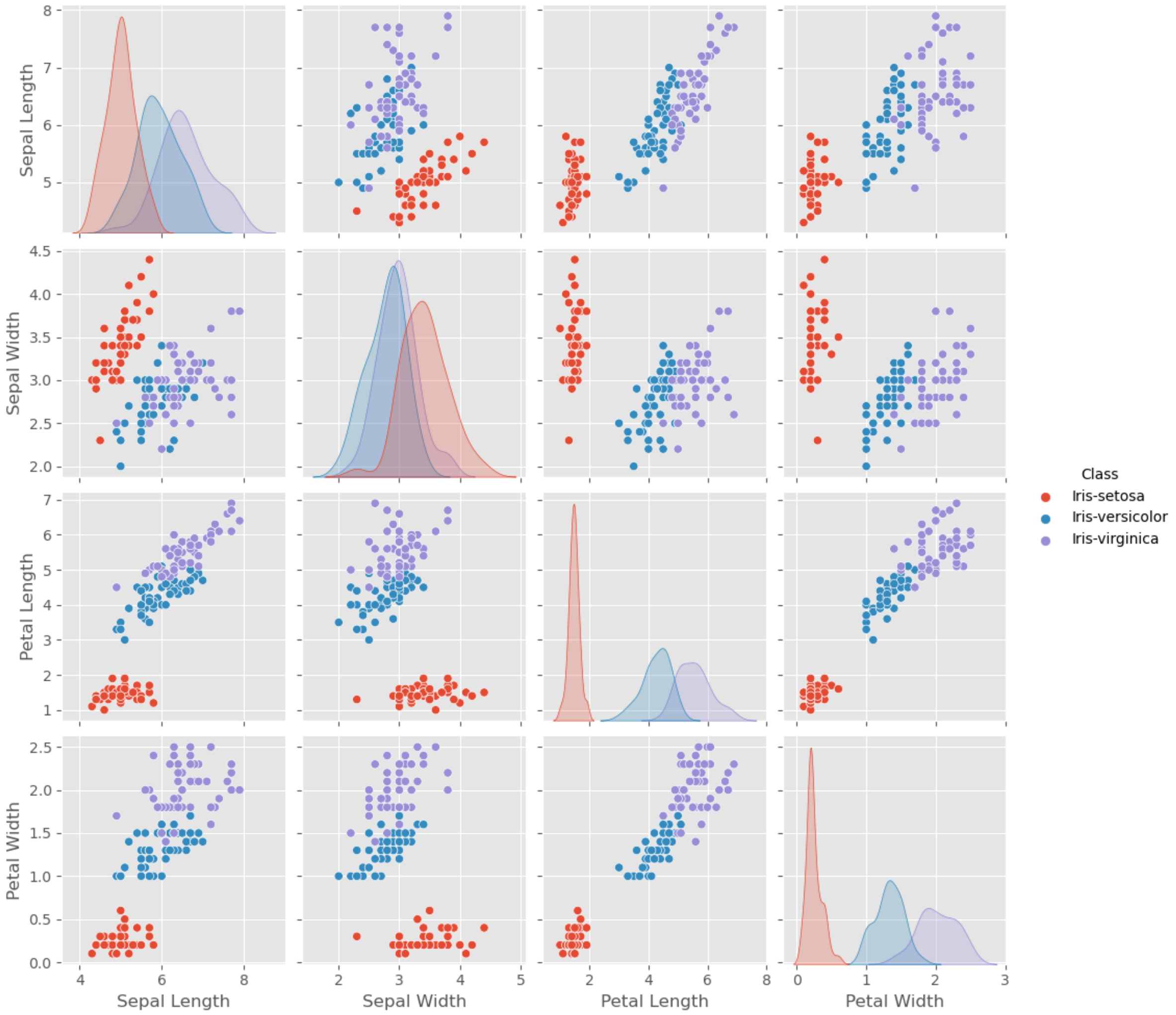


Pairplot

The `pairplot` of the *Seaborn* library is a powerful data exploration tool. It shows a plot of pairs of numeric attributes, and may represent as color the attribute chosen as Class (the `hue` parameter). In this specific case the high number of attributes makes the representation not very clear.

Use df as argument to the `pairplot` method of Seaborn, specifying also `hue = '...'` and `diag_kind='kde'` (try also other options)

```
In [ ]: sns.pairplot(df, hue = 'Class'
                     , diag_kind='kde'
                    );
```

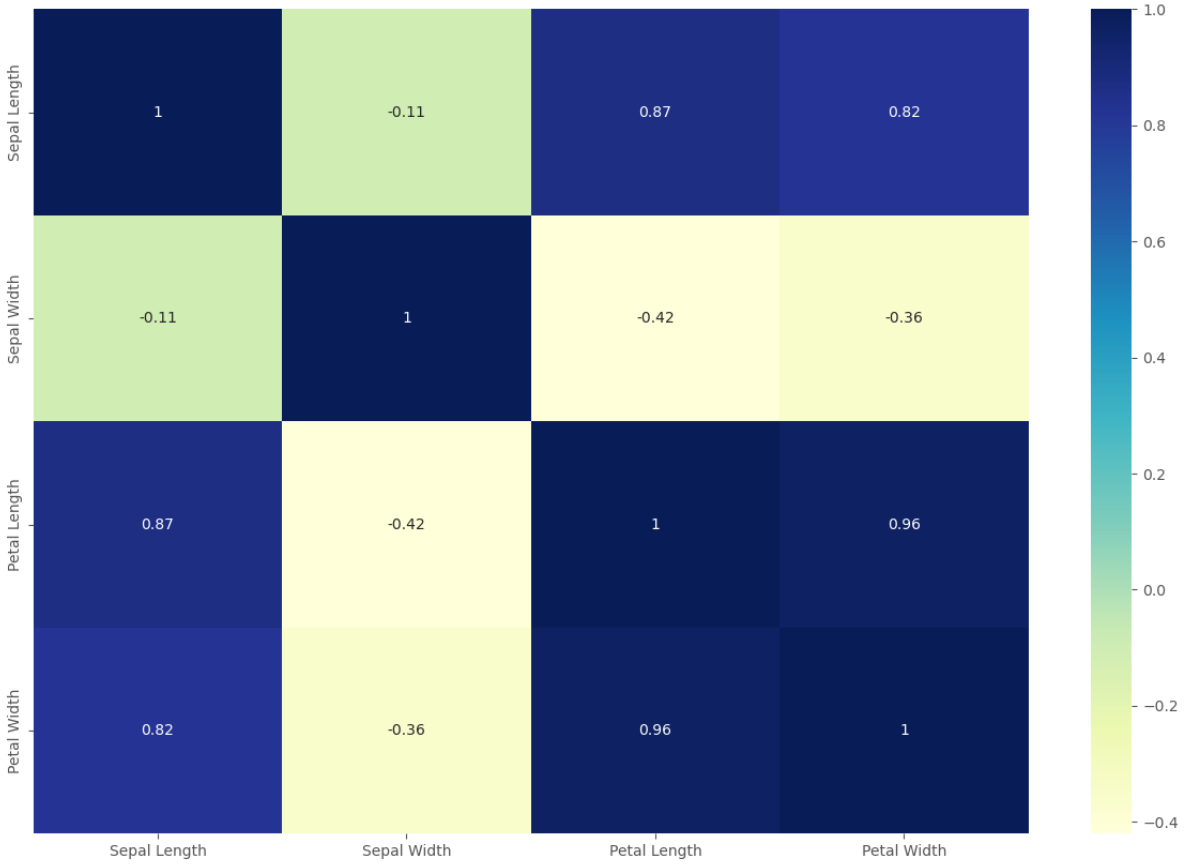


Show the *correlation*

Documentation Wikipedia Reference

In [ ]:

```
corr = df[df.columns].corr()
plt.figure(figsize=(15,10)) # set X and Y size
sns.heatmap(corr, cmap="YlGnBu", annot=True);
```

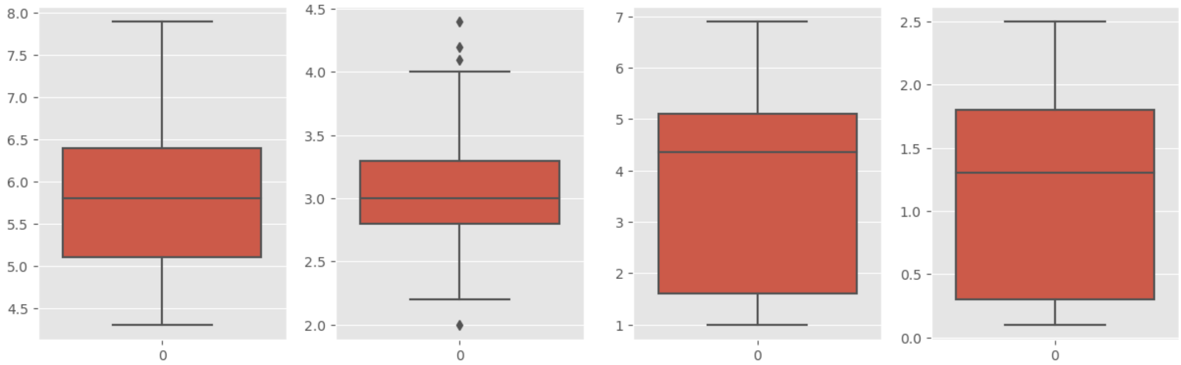


Boxplot

Now we will explore the distribution of the values inside each column using the `boxplot`.

This kind of plot shows the three quartile values of the distribution along with extreme values. The “whiskers” extend to points that lie within 1.5 IQRs of the lower and upper quartile, and then observations that fall outside this range are displayed independently. This means that each value in the boxplot corresponds to an actual observation in the data (*from the official Seaborn documentation*)

In [ ]:
```
#--Checking Outliers
plt.figure(figsize=(15,15))
pos = 1
true_columns = df.columns[0:-1]
for i in true_columns:
    plt.subplot(3, 4, pos)
    sns.boxplot(data=df[i])
    pos += 1
```

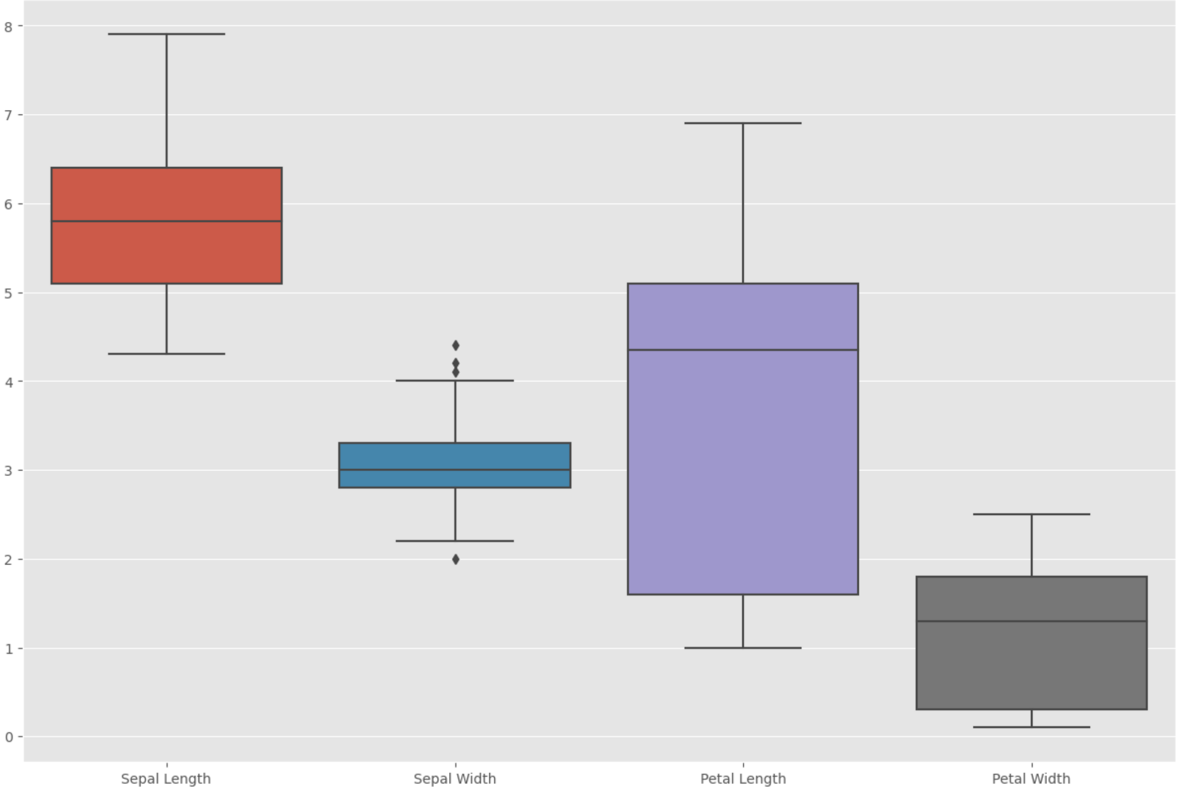


Comment what you see, are there relevant situations? outliers?

Another way to produce a *boxplot*

In [ ]:

```
plt.figure(figsize=(15,10))
sns.boxplot(data = df);
```



A boxplot for an attribute and the target

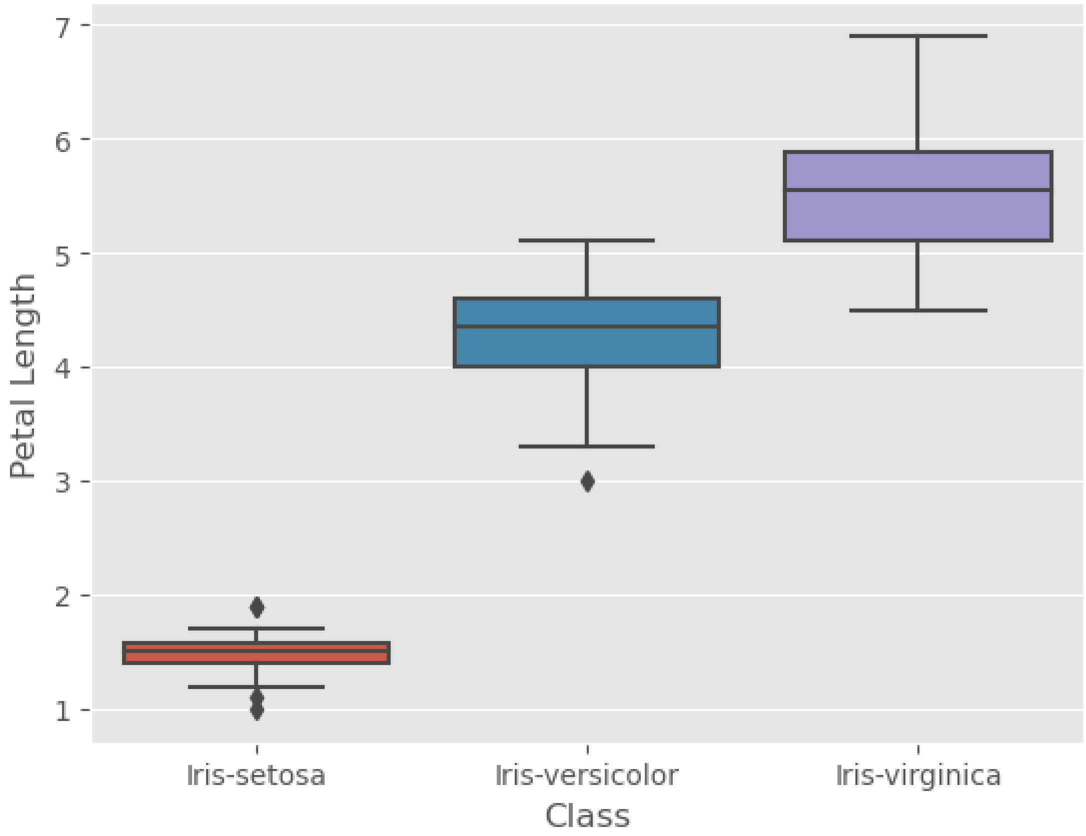
Put the attribute in the `y` axis, the target in the `x` axis

In [ ]:

```
sns.boxplot(x='Class', y='Petal Length', data = df)
```

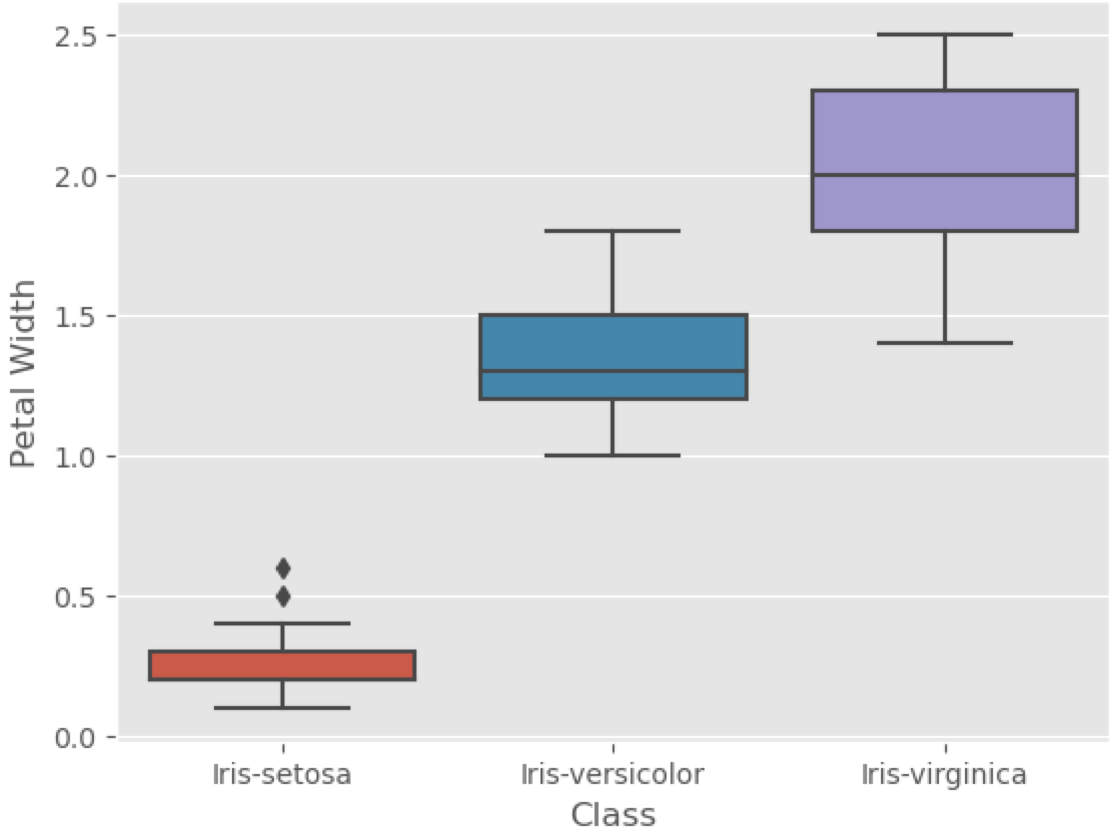
Out[ ]:

```
<AxesSubplot:xlabel='Class', ylabel='Petal Length'>
```



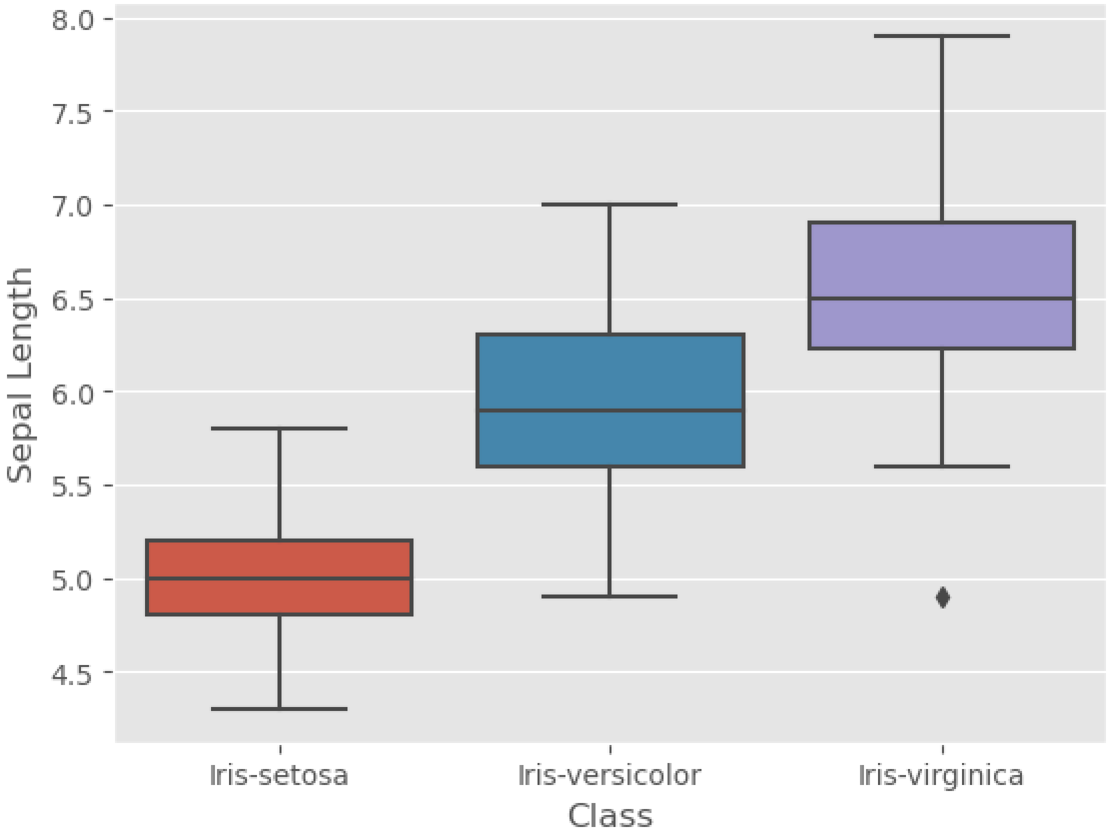
In [ ]:
```
sns.boxplot(x='Class', y='Petal Width', data = df)
```

Out[ ]:
```
<AxesSubplot:xlabel='Class', ylabel='Petal Width'>
```



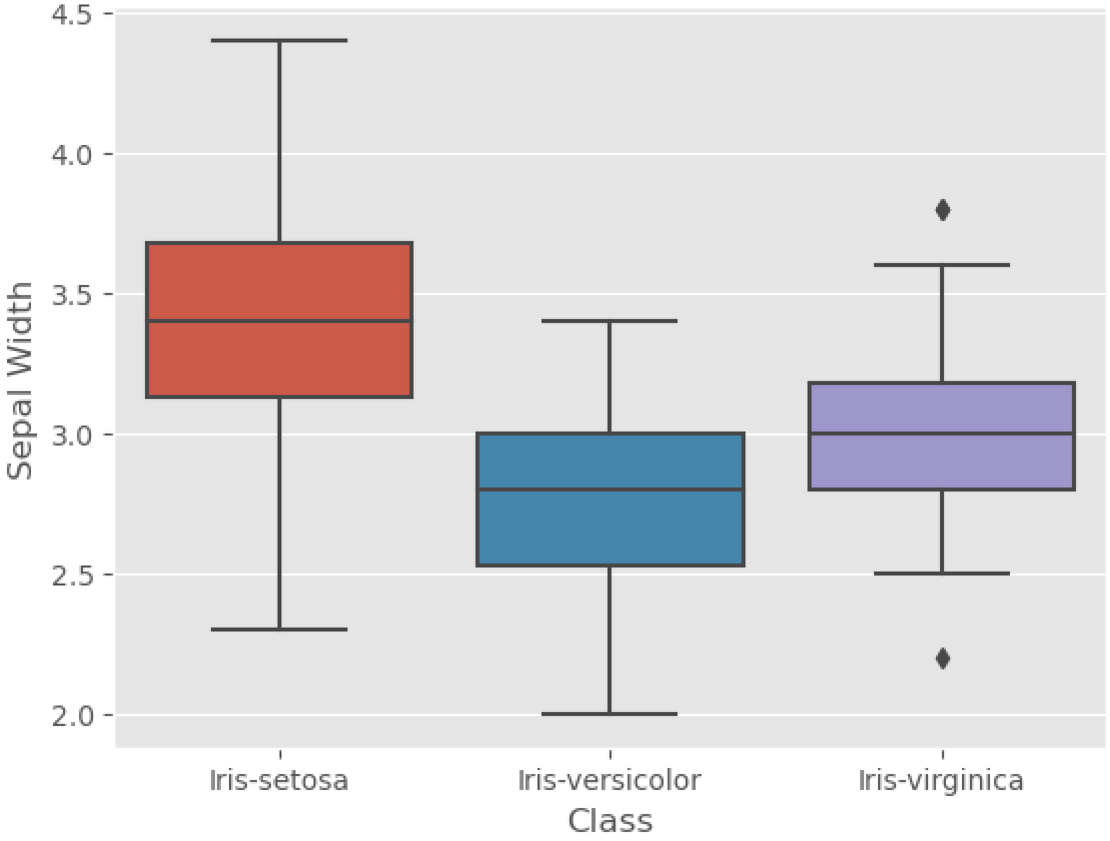
In [ ]:
```
sns.boxplot(x='Class', y='Sepal Length', data = df)
```

Out[ ]:
```
<AxesSubplot:xlabel='Class', ylabel='Sepal Length'>
```



```
In [ ]: sns.boxplot(x='Class', y='Sepal Width', data = df)
```

```
Out[ ]: <AxesSubplot:xlabel='Class', ylabel='Sepal Width'>
```


Sepal Width
Iris-setosa
Iris-versicolor
Iris-virginica
Class