Train the Iris data and Test the Model

- Iris is perhaps the best known database to be found in the pattern recognition literature.

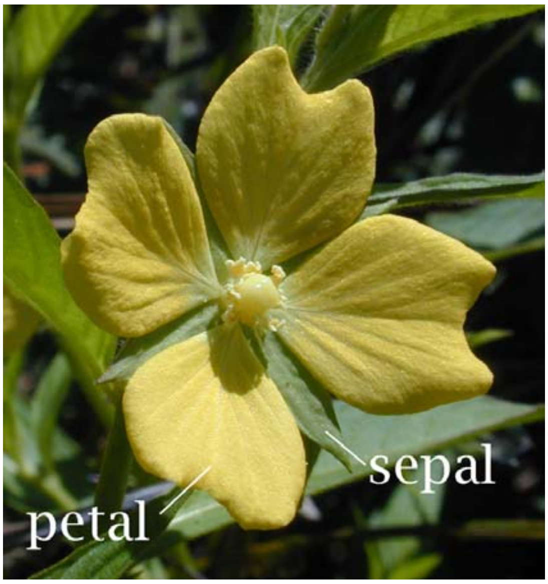

Figure 1: Flower showing petal and sepal [1].

- The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.
- One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.
  - Number of Instances: 150 (50 in each of three classes)
  - Number of Attributes/features: 4 numeric, predictive attributes and the class
  - Attribute Information:
    1. sepal length in cm
    2. sepal width in cm
    3. petal length in cm
    4. petal width in cm
    5. class:
       - Iris Setosa
       - Iris Versicolour
       - Iris Virginica

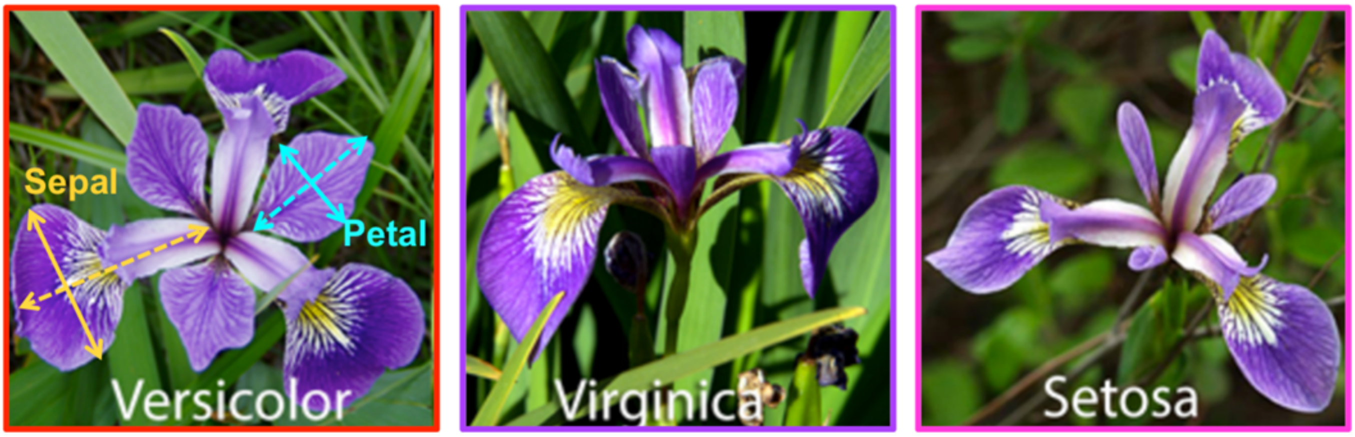

Figure 2: Three categories of Iris flowers [2].

In [ ]:
```
# Load pandas library. We want to use its DataFrame which supports tabular form.
import pandas as pd
```

In [ ]:
```
# Load the dataset from the current directory into a DataFrame
iris=pd.read_csv("iris.arff")
```

In [ ]:
```
# See the content of housing
iris
```

In [ ]:
```
# Want to see a few rows (actually 4, but 5 including the header)
iris.head()
```

In [ ]:
```
# Want to see a few last rows
iris.tail()
```

In [ ]:
```
# Information about the dataset
iris.info()
```

In [ ]:
```
# Some Statistical info. of the dataset
iris.describe()
```

In [ ]:
```
# Want to see the column
iris.columns
```

In [ ]:
```
# I am intertested to see the unique values in the class column because I want to replace the text with the numeric values
iris['class '].unique()
```

In [ ]:
```
# Visualizing pairwise relationships
import seaborn as sns # for more on seaborn, see https://seaborn.pydata.org/
```

In [ ]:
```
sns.pairplot(iris); #';' avoid outputing the internal location info here
```

In [ ]:
```
# Want to see the histogram of the numerical columns using malplotlib
import matplotlib.pyplot as plt
%matplotlib inline
iris.hist()
plt.show()
```

In [ ]:
```
# I want to replace 'Iris-setosa' with 0, 'Iris-versicolor' with 1, 'Iris-virginica' with 2
iris.replace("Iris-setosa",0)
```

In [ ]:
```
# But the above table is a view - and the replacement will not be a permanent change [we need to use option: inplace=True]
iris
```

In [ ]:
```
# We also do not want to change the original dataset, so we make a copy

iriscp=iris.copy()
iriscp
```

In [ ]:
```
iriscp.replace(to_replace="Iris-setosa",value=0,inplace=True)
#iriscp.replace("Iris-setosa",0,inplace=True) # This will work as well
```

In [ ]:
```
iriscp
```

In [ ]:
```
# Instead of replace them one-by-one I want to replace them all at once
# So I make a dictionary (dict) first
myreplacementlist= {"Iris-setosa":0, "Iris-versicolor":1,"Iris-virginica":2}
```

In [ ]:
```
myreplacementlist
# Note: I want the replacement to work only for column 'class '
```

In [ ]:
```
# Testing the dict
myreplacementlist['Iris-versicolor']
```

In [ ]:
```
iriscp.replace({'class ': myreplacementlist}, inplace=True)
```

In [ ]:
```
iriscp
```

In [ ]:
```
# Now, all columns are numerical column - I want to run the pairplot again
sns.pairplot(iriscp);
```

In [ ]:
```
# Want to examine how features help separate classes
sns.relplot(x='petal_length',y='petal_width',data=iriscp, hue='class ', style='class ');
```

In [ ]:
```
# I want to save this table into a file
iriscp.to_csv('myiriscp.csv')
```

In [ ]:
```
# read the file to check whether it is saved or not
!cat 'myiriscp.csv'
```

In [ ]:
```
# I can also use window's type command
!type myiriscp.csv
```

In [ ]:
```
# Now, I see index of each row is saved as well, which is a new item and it is now the 1st column of the table.
# I do not want the index column to be saved - so, I use a modified command below:
iriscp.to_csv('myiriscp_nonewcolumn.csv', index=False) # So, I made the index=False and it worked (see below)
```

In [ ]:
```
# I want to read from this file - which I might need to do in future.
# I am reading in, say, 'irisnewcp' DataFrame
irisnewcp = pd.read_csv('myiriscp_nonewcolumn.csv')
irisnewcp
```

In [ ]:
```
# I see the index column above but not in the file - so it is created on the fly. Check the columns:
irisnewcp.columns
```

In [ ]:
```
# I see the dataset is originally sorted (class-wise), which is not a good idea for machine learning - let us unsort it
from sklearn.utils import shuffle # NOTE: sklearn (Scikit-learn) will be our main Machine Learning python library
```

In [ ]:
```
irisnewcp_sh=shuffle(irisnewcp, random_state=345) # 'random_state' is used for initializing the internal random number generator
```

In [ ]:
```
irisnewcp_sh
```

In [ ]:
```
X=irisnewcp_sh.iloc[:,0:4] # 'iloc' is integer index based, so you have to specify rows and columns by their integer value of the index
X
```

In [ ]:
```
y=irisnewcp_sh.iloc[:,4:5]
```

In [ ]:
```
y
```

In [ ]:
```
# Let us use kNN, with k=5, from sklearn
from sklearn.neighbors import KNeighborsClassifier
```

In [ ]:
```
# create an instance of KNeighborsClassifier along with necessary parameters
knn = KNeighborsClassifier(n_neighbors=5)
```

In [ ]:
```
# print the instance variable to see the parameters of knn
print(knn)
```

In [ ]:
```
# Train the classifier with the dataset
knn.fit(X,y)
```

In [ ]:
```
# Since index column (& header) is a problem now, I need to drop the index column (& header) from both X and y
# Also, sklearn expects X, y in array
X=X.values.tolist() # 'values' are the content without the header and index of the DataFrame. toList converts into array
X
```

In [ ]:

```
# flatten() will remove the header and will convert y in a 1d array.
#You can also use .ravel().  .ravel() returns a view and .flatten() return a copy
y=y.values.flatten()
y
```

In [ ]:

```
# Now, try to train again
knn.fit(X,y)
```

In [ ]:

```
# create sample test dataset => expected answers are: 0, 2, 1
X_test = [4.8, 2.9, 1.54, 0.15], [5.9, 2.5, 5.5, 1.2], [5.9, 3.0, 4.6, 1.4]
```

In [ ]:

```
# predict the class to which the sample falls into
knn.predict(X_test)
```

Save and Load the Model

In [ ]:

```
# Python pickle module is used for serializing and de-serializing a Python object structure
import pickle
# Note: you can also use joblib
# joblib is optimized to be fast and robust on large data in particular
# to write use 'joblib.dump'  & to read use 'joblib.load'
```

In [ ]:

```
# Save the model
f1=open('iris_saved_knn_model','wb') # wb => write binary
pickle.dump(knn, f1)
```

In [ ]:

```
# better close (or flush) a file when done.
f1.close()
```

In [ ]:

```
# Load the model & Test
f2=open('iris_saved_knn_model', 'rb')
loaded_model = pickle.load(f2)
```

In [ ]:

```
X_test = [4.8, 2.9, 1.54, 0.15], [5.9, 2.5, 5.5, 1.2], [5.9, 3.0, 4.6, 1.4]
```

In [ ]:

```
loaded_model.predict(X_test)
```

In [ ]:

```
# If you know the test answers and want to compute the accuracy then do the following
Y_test=[0,2,1]
accuracy = loaded_model.score(X_test, Y_test)
```

In [ ]:

```
print(accuracy)
```

In [ ]:

```
f2.close()
```

References:

[1] https://en.wikipedia.org/wiki/Sepal (https://en.wikipedia.org/wiki/Sepal)

[2] http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/ (http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/)