# Support Vector Machine and Supervised Learning Methods

1. Dividing dataset for training and testing\ train_test_split(arrays, options).
   *Options*

   - arrays for data and lables
   - test_size: default value is 0.25
   - train_size: the remaining
   - random_state: can set a seed number
   - shuffle: default is true
   - stratify: default is none, meaning that the shares of groups are not changed.

In [ ]:

```
# Importing tools
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split
import seaborn as sns
```

In [ ]:

```
# Getting familiar with train_test_split function
# Create X dataset (100 by 2) containing elements from 0 to 199
# Create y dataset (100 by 1) containing elements from 0 to 99
X, y = np.arange(200).reshape((100,2)), np.arange(100)
print(X.shape, y.shape)
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=1011)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
print(X[0:7,:], y[0:7])
print(X_train[0:7,:], y_train[0:7])
```

In [ ]:

```
# reading iris dataset
iris_data = pd.read_csv('data/iris.csv')
#print(iris_data.head())

# Separating features and lable
X_iris = iris_data[['SepalL', 'SepalW', 'PetalL', 'PetalW']]
y_iris = iris_data['Name']

# Making training data and test data
X_train, X_test, y_train, y_test = train_test_split(X_iris, y_iris, random_state=1011)
#print(X_train.head(),y_train.head())   # check if X data are matching with labels!
#print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

# Training with SVM algorithm
svmt = svm.SVC(gamma='auto')   # SVC for classification (RBF) and SVR for regression
svmt.fit(X_train, y_train)     # Learning
class_prediction = svmt.predict(X_test)   # Classification, prediction

# Calculating accuracry
ac_score = metrics.accuracy_score(y_test, class_prediction)
print(f'Accuracy rate = {ac_score:.5f}')

# Checking incorrect predictions: a list comprehension
wrong = [(p, e) for (p, e) in zip(class_prediction, y_test) if p != e]
print(wrong)

confusion = metrics.confusion_matrix(y_true=y_test,y_pred=class_prediction)
print(confusion)

plt.figure(figsize=(6,5))
confusion_df = pd.DataFrame(confusion, index=['setosa','versicolor','virginica'], columns=range(
3))
axes = sns.heatmap(confusion_df, annot=True, cmap='nipy_spectral_r')
plt.show()
```

In [ ]:

```
# K-nearist Neighbors Algorithm
from sklearn.neighbors import KNeighborsClassifier
```

In [ ]:

```
# Naive-Bayes Gaussian Kernel
from sklearn.naive_bayes import GaussianNB
```