**IDE 1: Computational Intelligence Tutorial**

**Week 1**

April, 2018

**Introduction**: This week you will were introduced to the basic principles and intuition behind some main Machine Learning methods. This tutorial is aimed at familiarizing you with a sample selection of these methods at an implementation level. The exercises that follow introduce you to data loading, training and testing. They present this pipeline of activities in a step-by-step, fully explanatory fashion. You are expected to complete them by (i) following their detailed instructions on the doc and ipynb files and (ii) answer their qualitative questions to demonstrate a deeper understanding of the employed tool. On the implementation side, you are essentially expected to run all steps presented in the ipynb files in your own jupyter notebook environment.

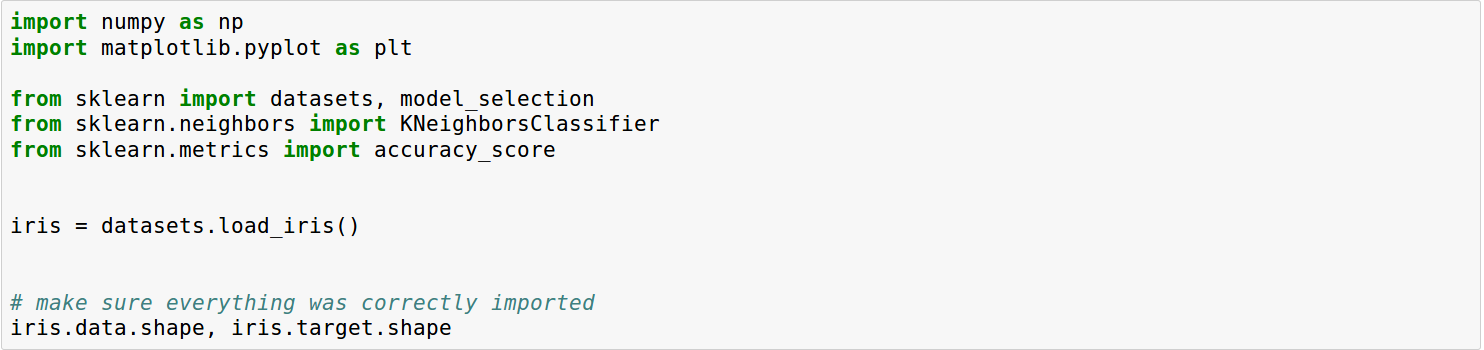
Completion of one out of three exercises will reflect a satisfactory familiarisation with the tutorial’s content. Completion of two or three exercises will reflect a fuller exposure and a top degree of familiarisation at this stage of the course.

**A. Classification: K-nearest neighbors classifier**

1. Lets jump right into it by importing the required libraries and load the iris dataset.

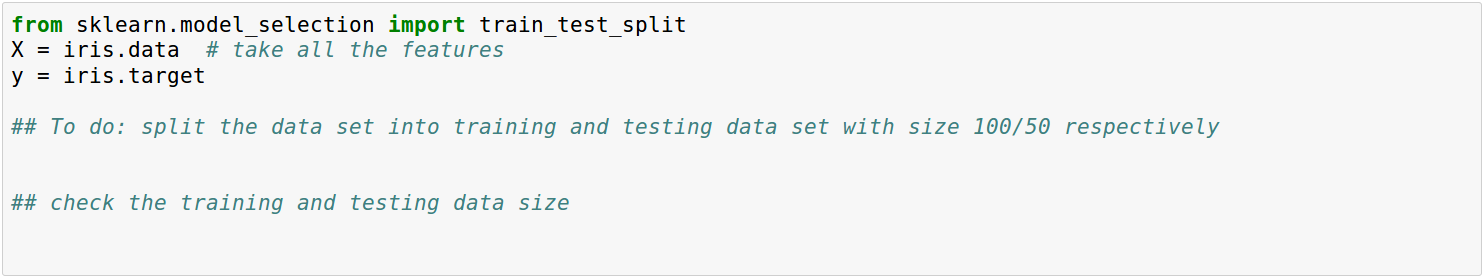
This data sets consists of 3 different types of irises’ (Setosa, Versicolour, and Virginica) petal and sepal length, stored in a 150x4 numpy.ndarray

The rows being the samples and the columns being: Sepal Length, Sepal Width, Petal Length and Petal Width.



2. Building data sets

Let's start the construction of the K-NN model by splitting the whole data set into train data set and test data set. Play around with 'random\_state' to generate different split.



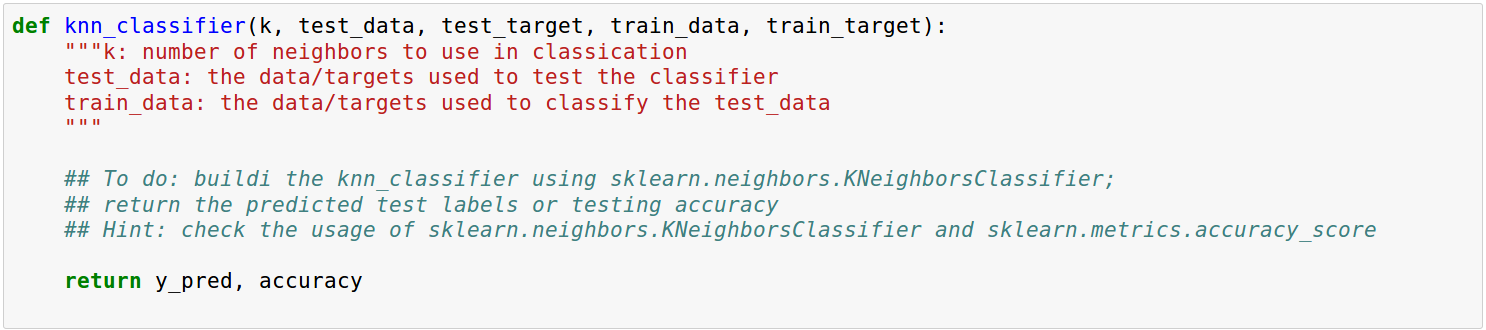
3. Visualize training data

Visualize the first two features of the training data. Optional: visualize the last two features of the data.



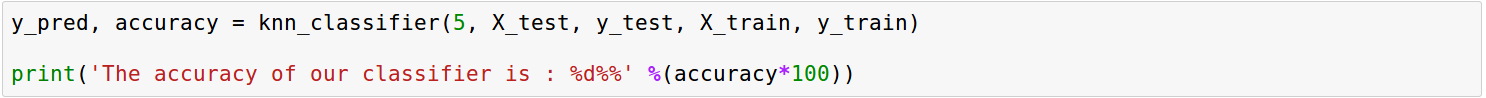
4. Building the model

We will start by putting the Scikit-Learn K-NN model into a function so we can easily call it and adjust it.



5. Testing

Now lets see how this model performs on test sets.

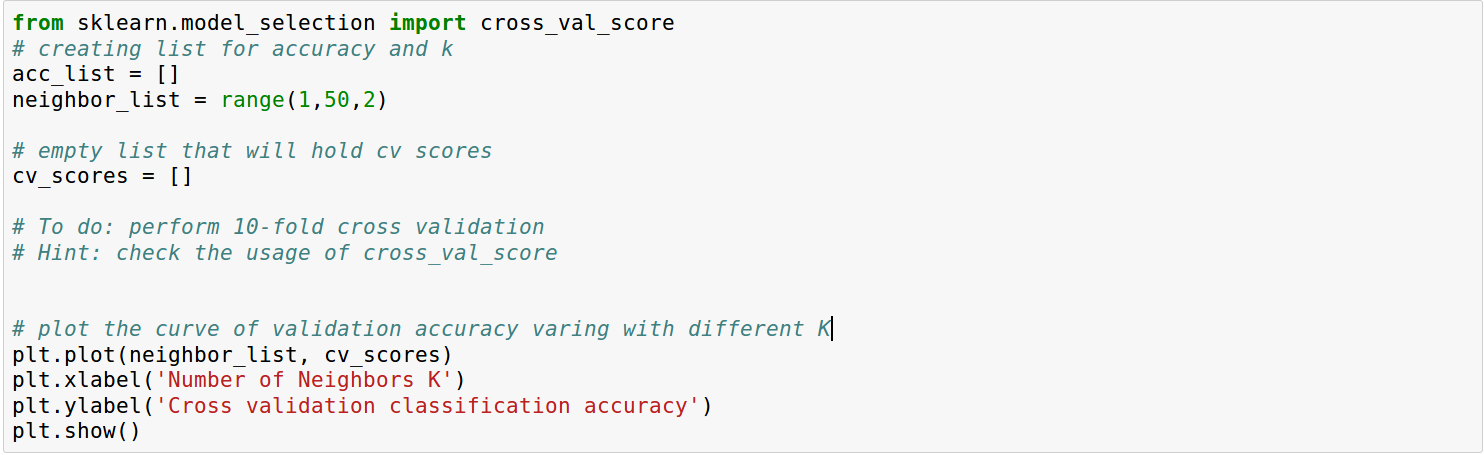


Question: Change the variable K in knn model to see how it will affect the accuracy on test data.

6. Parameter Tuning with Cross Validation

We’ll explore a method that can be used to tune the hyperparameter K. Cross-validation can be used to estimate the test error associated with a learning method in order to evaluate its performance, or to select the appropriate level of flexibility.

k-fold cross validation (the k is totally unrelated to K) involves randomly dividing the training set into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k−1 folds. The misclassification rate is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error which are then averaged out.



Question: Did you notice the difference of best K when tuned on test data and on validation data? How would you interpret this?

**B. Regression: Linear and logistic regression**

FiveSixEight bar at Imperial College London launched a new app to assure the school that students going to the bar were ending up safely at home after attending the bar. The app tracks how many pints (or alcoholic drinks) each student has, and then asks the students in the morning whether they ended up safely in their own home. Students at Imperial are extremely honest, especially when they are hung over, so assume that all information from the app is valid.

Use the python and data files sent to you as a starting point.

1. Why can’t this problem be handled by linear regression? What benefits does logistic regression give?

2. Only changing the num\_beers parameter in the code:

a. What is the chance that somebody drinking 1 beer will make it home?

b. What is the chance that somebody drinking 10 beers will make it home?

c. What can you say about these results? Do they seem reasonable? What might be going wrong with the model?

d. Plot the values of the cost and each of the coefficients over the number of iterations. What can this tell you about problems you have been having?

3. Change the parameters marked with # \* #. Using information from plots and the final output (the chance of a student having *n* beers arriving home), justify how the performance of the model is influenced by changes in:

a. The number of iterations (num\_iters)

b. The initial coefficients (theta)

c. The learning rate (learning\_rate)

4. What do you think are the optimal values for these 3 parameters? Explain your answer in real-world terms that could be applied to other problems, considering that most real-world and industrial problems are much more complex than this.

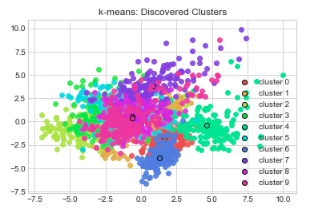
**C. Clustering: k-means**

In this tutorial, you will explore clustering algorithms. Learning outcomes:

* Familiarise yourself with the concept of clustering
* Implement basic clustering algorithms (k-means)
* Understand some of the common problems with clustering

# What is clustering?

Data clustering falls under **unsupervised learning** problem. Unlike supervised learning (classification/regression), we often do not have labels (**most real-world problems fall under this category! ☹**), but we tend to have large amount of unstructured data. In this case, we can still learn something useful from data by grouping the data points together based on some similarity measures. In this tutorial, we will implement **k-means** clustering ourselves (it’s very simple!) and try to fit it to toy data.



# k-means Algorithm

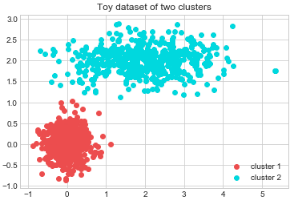
* Given dataset of size n, the goal is to split them into K disjoint clusters. k-means algorithm achieves this by discovering K centroids, which means, the average locations for each data cluster.
* This is solved using iterative algorithm: weiteratively update the estimate of the centroids which fits the data better and better every time. The idea is to first initialise the centroids randomly. Then, we alternate between the following two steps:
  + (Step 1) Find a cluster assignment for each data point based on current estimate of the centroids. In particular, for each point we compute the distances between the point and all the centroids. Then, we pick cluster j that is the closest. We assign j to the data point.
  + (Step 2) Update the centroid locations based on the newly assigned data points. In particular, for each cluster j, we compute a brand-new centroid location by simply taking the average of all the points in that cluster! We repeat this for all clusters j=1…K.
* The first step essentially splits the space based on the current estimate of the centroids. The second step updates the centroid location to fit the data even better. After a few iterations, the method will converge to a "stable" solution. That's it! Let's Implement it.

# Exercise: Implement k-means algorithm

In below, fill in the blank to complete the implementation of k-means. You only need to implement two functions to make the algorithm work. Take your time, discuss with your neighbours if needed. You can use any library including numpy.

* Read and understand the **k\_means(X, k, num\_iter)** function. Explain to your neighbour what are the parameters and how the code works.
* **Exercise (1)** Implement function: **get\_argmin\_dist(distances)**
* **Exercise (2)** Implement function: **compute\_mean(S\_j)**

After you successfully run it, we will apply the algorithm to the following toy dataset:



Question (1) What do you observe? Did it behave as you expected?

Answer: k-means assumes each cluster has equal variance. In practice, this may not be the case, as we see for this dataset. However, k-means nevertheless is a simple, efficient and effective algorithm that is used often.

Question (2) You are given a function called **compute\_loss(X, mu)**. This function computes *loss term*L, which measures how good the centroids are fitting the data X. Lower the better!

* Run k-means with k∈{1,2,3,4,5}k∈{1,2,3,4,5} and compute loss term L (use the code below). Which k fits the best? We know that the true data only has 2 components. What does this mean?

**Exercise (3):** Answer this question!

**\*\* For this exercise, you would need to install a library called *seaborn*, which is a wrapper on top of matplotlib which gives you pretty colours! You can install this by**

* **pip install seaborn**