Practical Session 2

In this week's practical session, we will recap the programs of our lectures. To practically read and write algorithms that we have learned in the class, which is really beneficial. You should understand the logic behind them. Later we will have our interesting leetcode session and Q&A session (anything that is not clear, including programs or concepts in the class).

Your Tasks

1. Start-up Jupyter Notebook, Colab or transform the ipynb files into python then

use Pycharm. (any python interpreter that you like)

2. Warm-up

Try to recap the program demos that we saw in the lecture and try out as many of

these examples as possible.

3. Exercises of Linear Regression Algorithms

* Download the “linear\_reg” algorithm from the home page that I shared.
* Run the python code and debug your Python code line-by-line.

4. Leetcode programming session

* This session is designed to let you use python more proficiently. Leetcode is an amazing code practice website, and we could probably solve a leetcode problem in each practice class. Many programming problems are published to improve your coding skills.
* Open <https://leetcode.com> and register.
* Choose “problems”, and select “two sum”.
* Choose a python compiler and complete this task (without looking at the solutions).

5. One of the tasks that you need to perform prior to training your machine learning model is data preprocessing. Data cleansing is one main part of the data preprocessing task, and usually involves removing rows with empty values, or replacing them with some imputed values (i.e, handling missing values). In statistics, imputation is the process of replacing missing data with substituted values. In this exercise, you will use SimpleInputer in sklearn to fill the missing values of the following iris dataset.

from sklearn.datasets import load\_iris

# obtain the iris dataset, and add some missing values to it

X, y = load\_iris(return\_X\_y=True)

mask = np.random.randint(0, 2, size=X.shape).astype(bool)

X[mask] = np.nan

X\_train, X\_test, y\_train, \_ = train\_test\_split(X, y, test\_size=100, random\_state=0)

6. Load the wine dataset in sklearn (from sklearn.datasets import load\_wine) and split it into a training set and a test set (80% for training, and the remaining for testing). Then train various classifiers (e.g., Random Forest classifier and SVM - any classifiers you like). Next, try to combine classifiers to develop an ensemble using a soft or hard voting classifier. Try this model on the test set. Did it perform better compared to the individual classifiers? Use the evaluation metrics such as precision, recall and F1 score to compare the performances of your models. Discuss and compare the model performances with your peers to learn their observations.

7. Q&A session

You could ask any questions that are not clear, including programs or concepts/jargon in the class.