README:

The repository can be accessed at the following URL:

https://hub.docker.com/r/zhuweida/docker2/

Pulling the docker image:

Use the command:

docker pull zhuweida/docker2

Running the docker image:

Use the following commands:

docker run -it zhuweida/docker2

cd src

To copy the folder into your local machine, use:

docker cp (container id):/src c:/

The command "docker –ps" can be used to find the container ID.

Scripts in the image:

1. pr2.py is used to preprocess the data and calculate summary statistics of interest for a given year. Different years require separate runs of pr2.py.

   How to run the code:
   Spark-submit pr2.py dir_2004.csv
   Spark-submit pr2.py dir_2006.csv
   Spark-submit pr2.py dir_2008.csv
   Spark-submit pr2.py dir_2010.csv
   Spark-submit pr2.py dir_2012.csv

   The "dir_****.csv" file contains all the columns which we are interested in and we manually put them into this csv file from the original csv file. "****" represents the year the data is taken from.

   Before running the code, you should change the name of the folder you want to save in line 108 of pr2.py. This .csv file stores all of the correlations between macronutrients and calorie intake for a given year. After running the code, you should enter the newly-generated folder and change the file format to .csv.

2. K-means.py is used to identify bounds for the continuous data, which we then used to cluster the data into several classes.

   How to run the code:
   Spark-submit k-means.py calorie.txt 3 1

Spark-submit k-means.py protein.txt 3 1
Spark-submit k-means.py fat.txt 3 1
Spark-submit k-means.py carbon.txt 3 1

(3 represents K, 1 represents the number of dimensions in the data)

NOTE: Missing data in this step has already been addressed.

3. xin.py is used to encode data into a new csv file.

   How to run the code:
   spark-submit xin.py after_merge_butbefore_reclustering.csv

   From this code, you can get a new .csv file which contains all the clustered data.

   You can change the name of output folder which will be generated after running the code on line 126. After running the code, you should enter the newly-generated folder and change the format of the file to .csv. “directly-used_for_decisiontree.xml” is the output file after encoding and rearrangement.

4. lib.py is used to convert .csv files into libsvm format in order to use them with MLlib for the decision tree classification:

   How to run the code:
   python lib.py fat2.csv fat2.txt 1 1
   python lib.py carbon2.csv carbon.txt 1 1
   python lib.py protein.csv protein.txt 1 1
   python lib.py calorie.csv calorie.txt 1 1

   After running the code, the user should use a text editor like Notepad++ to remove the empty lines (For Notepad++, the command is Edit -> Line Operations -> Remove Empty Lines). “calorie.txt”, “protein.txt”, “carbon.txt” and “fat2.txt” are the files in libsvm format from running the lib.py and removing the empty lines manually.

   NOTE: We do not own the copyright of lib.py; the script is found at
   https://github.com/zygmuntz/phraug/blob/master/csv2libsvm.py

5. DecisionTreeV2.py is used to generate the decision tree model in order to do the prediction of dietary data based on the information of demographic data.

   How to run the code:
   spark-submit DecisionTreeV2.py

   Before running the code, you should change the line 35 and input the correct path to the libsvm file of the data you want to use to build the decision tree.

In line 41, the “numClasses” is the number of features in the result column plus one, due to zero-indexing. For example, there are four different classes in “fat” column, so the numClasses is set to 5.

In line 41, “categoricalFeaturesInfo”, 0 represents the education level column and 1 represents the income:poverty ratio column. There are four different categories (1 to 4) in the education level and four categories (1 to 4) in the income:poverty ratio, so “categoricalFeaturesInfo” is set as “{0:5,1:5}” due to zero-indexing.

NOTE: The decision tree script is based on an example available at the following link: https://github.com/apache/spark/blob/master/examples/src/main/python/mllib/decision_tree_classification_example.py

ABOUT:
This README file was originally prepared by Weida Zhu, and revised by Giovanni Pagano. Weida also prepared the Docker image, and the scripts pr2.py, xin.py and k-means.py. Giovanni prepared the DecisionTreeV2.py script.

For the extended abstract, the Methodology and Future Work were originally prepared by Weida Zhu. The Motivation and Results were prepared by Giovanni Pagano, and Giovanni revised the extended abstract with Weida’s input.

The poster was prepared by both Giovanni Pagano and Weida Zhu, with feedback and revisions from Dr. Michela Taufer.