Related datasets proposal and integration example

The integration with smart fridges is neither required in this project nor feasible since it is a fresh technology and still a niche. Nevertheless, adapting the data structure from an eventual smart fridge to the one of the ingredients table, could already be enough to store the quantity, the remaining quantity and the expire date of ingredients. The technology to store items in a smart fridge is left to the relative producer. As for the recipes part, if new ingredients need to get inserted to the system, it should be enough to adapt the data structure of those to the already existing data provided in this project.

A very important information that is not present at all in the source data is the cuisine type (French, Japanese, thai, …). A useful way to get this information is provided now. It is possible to predict the cuisine type given the ingredients. To do that, a training set is required, containing for a set of ingredients a type of cuisine. Such a training set with 39774 examples was found on Kaggle. In Rapidminer, auto-modelling was used with different kind of models, like naïve bayes, generalized linear model, deep learning, decision trees and random forests. After training these models on the training set itself with a classic splitting into train and test set, the naïve bayes got a hugely higher accuracy than decision trees (around 75% against 20%). The ingredient table is the one that needed to be used to predict the cuisine type of recipes. It got adapted to the training dataset and then the prediction algorithm got started. It is not possible to have an accuracy of the results, but by checking manually a couple of predictions, it seems to work quite well. This procedure will be explained more precisely now.

The training set is a json file containing recipes with an id, a cuisine type and some ingredients. This data is transformed into a table that is imported in Rapidminer. The table has every ingredient contained in the whole dataset as columns. Every row, instead, is a recipe with an id, a label and has a Boolean value for each ingredient. Since a table of 39774x6716 elements is very large, the python importer script divided this task into 4 different batches. After the previously described auto-modeling, the ingredient table of this project needed to be adapted in order to be used for predictions. Thus, the same columns of the training set table are adopted and each ingredient of each recipe is compared to every column using the Levenshtein’s distance (string similarity). If there is an ingredient with a resulting distance less than a threshold (in this case 5), then the column value is set to true; if there are more, instead, the one with the lowest similarity is chosen. This way, the small inconsistencies between ingredient names are not a problem among the different data sources. Now, prediction is possible by using the generated pipeline of the auto-modeling in Rapidminer. The only issue arisen is that the pipeline automatically splitted the training set in order to produce a test set, but it was not the goal of the application. So, with some tweaking, the just described adapted ingredient dataset is inserted into the pipeline. Repeating this last step of the procedure with all three datasets lead to cuisine type predictions for every data source.