Assignment 6 – big data

In assignment 6, we use smaller proteins (smaller that 90%) to predict the function of the protein, using classification. First, I will explain how I did the data processing, so it was ready to test and train classifying models on. Then I will explain which models I used and how the data, model and their performance is saved and how these saved files could be used with the respective script for further research.

Since we are working with vast amounts of data, we use Dask to increase performance. Dask is a Python library for parallel computing that allows users to work with larger-than-memory datasets. With Dask, users can perform common data analysis and machine learning tasks on datasets that would be too large to fit into memory using traditional approaches.

# Data processing

First, we read in the data and extract the data we need, using the file\_cleaner(df) function from the processing file. This function takes a df as argument and extracts and returns the data we will use in further processing.

After this, we use the function find\_long\_short(data) to find which protein lengths are greater than 90% and which are smaller that 90%. This data will later be used as the labels and the features.

The extracted log and short proteins are merged into one dataset, called ‘merged’, since this makes further preprocessing (deleting rows, categorizing, etc. while keeping the correct index) easier.

I then use create\_matrix(data) to create a pivot table, which includes the sum of short proteins in a protein function. The long proteins are then merged to the pivot table on the interpro\_accession. I did this, to get the correct index again. From this merged (and cleaned from NaN) dataset I extract the features and labels.

These features and labels are then split into a training and testing dataset, with ratio 70/30 respectively.

# Machine learning

The data described above was used to train a Random Forest classifier and a Decision Tree classifier.

The trained model could then be saved, together with the processed data, so repeatability of training and or predicting is available. Importing a model could be done by uncommenting line 41 and 42 and changing the filename to the desired model.

# Future improvements

* **Hyper parameter tuning:** to get a higher accuracy score, the models could be optimized using hyper parameter tuning.
* **Multiple models:** aside from the tested random forest and decision tree, more models like XGboost, neural networks, etc. could have been tested and evaluated.
* **Data processing optimization:** I feel like I am splitting and merging the data quite often, which requires computing power. The performance of the script could be improved by programming this script more efficient.
* **Object oriented programming:** the script is in my opinion sufficiently structured, but not object oriented. This could be improved in the future.
* **More data:** I extracted 10,000 lines from the all-bacilli.tsv file, which is way larger. The accuracy score could increase when more data is used to train the model.