Protein Sequences Classification Problem

Federica Lupo
Politecnico di Torino
Student id: s302644
s302644@studenti.polito.it

***Abstract*—This report offers a solution to protein sequences classification problem. The aim is to predict the functions associated to a protein given its sequence of amino-acids. The most interesting phase is sequence preprocessing and features extraction, in order to obtain numerical data to train the model. Followed by the selection of the most suitable model and its hyperparameters. This report[1] wants to offer a baseline for future studies.**

I. Problem overview

The assignment 13 of the course BioQuants focuses on the use of Machine Learning techniques to predict functions of proteins. The first step foresees the recovery of the dataset, UniProt Knowledgebase (UniProtKB[2]) is the resource used to extract it. It offers two database for the proteins:

- a *reviewed* version, counting 569 516 rows, characterized by information verified by experts, considered accurate and complete, since validated experimentally;
- an *unreviewed* version, comprising 249 308 459 records, includes automatically annotated proteins by bioinformatics tools and not yet validated by experts. Information could be incomplete and not experimentally verified.

For this project, the first dataset has been considered. Here, a brief description of the main information included in the dataset:

- ***Entry***: identifier of the protein;
- ***Entry Name***
- ***Protein names***
- ***Gene Names***
- ***Organism***
- ***Sequence***
- ***Gene Ontology (molecular function)***

A representation of the top 6 organisms[3] presented in the dataset is shown in figure 1. Due to computational reasons[4], only 17 000 entries are considered. Initially were considered separately Human and Mouse proteins, since each one counted a reasonable amount of proteins, but due to some Google Colab limitations, first entries were considered. Here, fig. 2, an overview of the top 6 organisms represented in the narrowed dataset. One of the possible reasons could be that the length of the sequences considered in the narrowed dataset differs

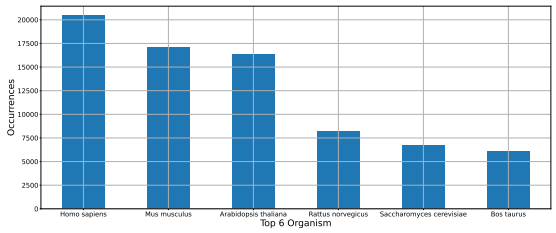

Fig. 1. Top 6 organisms based on the number of proteins associated

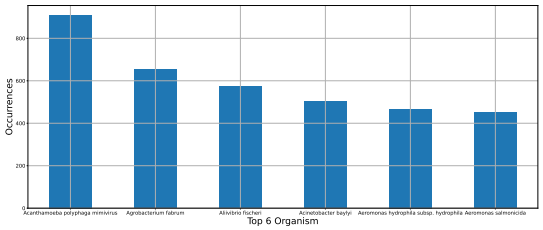

Fig. 2. Top 6 organisms based on the number of proteins associated (narrowed dataset)

from that of the Human or Mouse dataset. From now on, observations refer to the narrowed dataset. By analyzing the missing values presence we noticed that 2 964 rows do not have a defined function and 630 entries do not have gene names. After removing this rows we obtained a dataset with 13 648 rows.
Finally, giving a look to the target classes: one protein could be associated to a lot of functions. Indeed, a protein could be involved in different biological processes [1]:

- some proteins are enzymes: they act as catalysts for chemical reactions. For example, each step of the conversion from nutrients to energy in human body is catalyzed by an enzyme;
- some proteins are structural: they give stability and shape, for example in connective tissue.

Here, fig. 3, a representation of the distribution of protein occurrences with respect to the number of functions associated to each. The majority of proteins counts 2 functions each. The total amount of functions is 1 448, figure 4 identifies the 5 functions most diffuse.

[1]Here the link to the notebook: https://colab.research.google.com/drive/1GnP1hhhpQ7smGsLBW_I2GC-E0CE3uG-Z?usp=sharing, for a better view of graphs and understanding of code part
[2]https://www.uniprot.org/uniprotkb?query=reviewed:true
[3]Organisms abbreviations have been considered
[4]Google Colab provides 12.7 GB of RAM

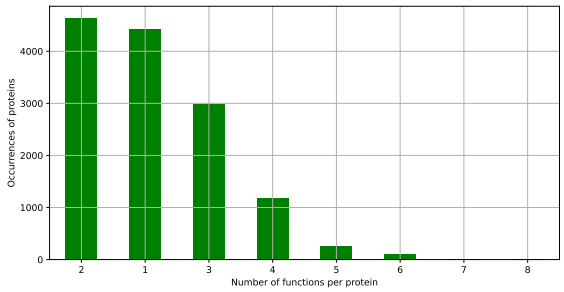

Fig. 3. Distribution of protein occurrences with respect to number of functions associated

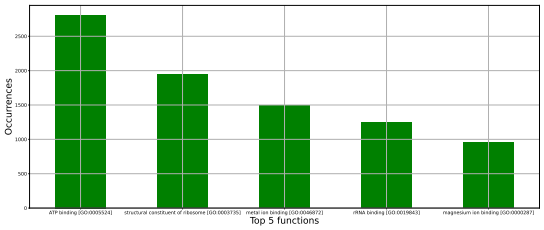

Fig. 4. Top 5 functions

II. Proposed approach

A. Preprocessing

In the suggested solution, we focus all preprocessing steps and features creation on the sequence of the protein, other columns have been discarded.

Initially, label encoder has been considered by transforming each sequence in a number, but since this creates an implicit sorting, other techniques have been applied.

The second trial consisted in the conversion of each sequence in a corresponding array[5], in which each element was represented by a number associated to the amino acid (24 numbers). This could be an optimal starting point since allowed to keep the order of the amino acids, but not suitable as input of the classification models, after that some processing of these arrays have been done. The measurement that we decided to extract are: sum, mean, and standard deviation. Since the sum does not improve results has been discarded.

Some duplicates have been detected, 2 072 rows, but we decided to keep them since associated to different entries and organisms.

Finally, in order to preserve a detail about the type of amino acids present in each sequence, columns represented them have been added as features, with the aim to count for each sequence the number of different simple molecules.

The total number of columns considered is 26.

Before proceeding with training, ***standardization*** has been applied, in order to remove implicit sorting given by possible changes in orders of magnitude, through MaxAbsScaler was possible to standardize values.

[5]In order to have array of the same length, a study was done in order to consider a mean length of the sequences, set at 500 elements

B. Model selection

In the choice of the models, the nature of the problem needs to be considered. In this case, we have a multi-label problem, in which each entity could be associated to more target values. Five models provided by scikit-learn [2] have been tested, four of these belong to the classifiers family, one is a regression model:

1) ***DecisionTreeClassifier***: classification model based on a decision tree. It is a supervised model that uses a tree structure to classify entries. Each node in the tree represents a test on a feature and each leave represents the target class;
2) ***RandomForestClassifier***: classification model composed by an ensemble of combined decision trees. Each decision tree is trained on a random subset of features and data, the use of the prediction mean allows to reduce the error of each decision tree;
3) ***ExtraTreeClassifier***: model similar to the DecisionTreeClassifier. Differently from it, the choice of the best test of the separation node is random and not optimal;
4) ***BaggingClassifier***: classification model based on the bootstrapping, a repeated sampling technique. Different subsets of the dataset are created, each one is used to train a classification model. Finally, final prediction is obtained by combining the prediction of all the training step;
5) ***RidgeClassifier***: classification model based on the logistic regression with regularization L2. Regularization is used to reduce overfitting and to increase stability of the prediction. This model uses a linear function to model the relationship between input and output variables.

C. Hyperparameters tuning

In order to determine the best hyperparameters for each model, and therefore the best one, the dataset was split into two subsets: *"train_valid"* and *"test"*, with the hold-out method 80/20.

In the hyperparameters tuning phase, the first set was used by the *GridSearchCV*, with the aim of evaluating the model. Then, the best configuration was tested on unseen data, represented by the second set. In this way, we obtained a final estimate of the goodness of the model. Through the *GridSearchCV* tool, provided by scikit-learn [2], it was possible to perform a cross-validation with 5 splits on the *train_valid* set, exploiting *StratifiedKFold* instrument. Furthermore, since *GridSearchCV* refits the best configuration with the entire dataset (in our case *train_valid* set), it was possible to directly make the prediction on the test set.

In table I, a recap of all combinations tried.

In figure 5 is shown a comparison of the different best performance for each model.

TABLE I
HYPERPARAMETERS TUNING

Model	Parameters	Values
DecisionTreeClassifier	criterion	[”gini”, ”log_loss”]
RandomForestClassifier	n_estimators	[100]
ExtraTreeClassifier	criterion	[”gini”, ”log_loss”]
	splitter	[”random”, ”best”]
BaggingClassifier	n_estimators	[5, 10, 15]
RidgeClassifier	alpha	[0.5, 1.5, 2, 5]

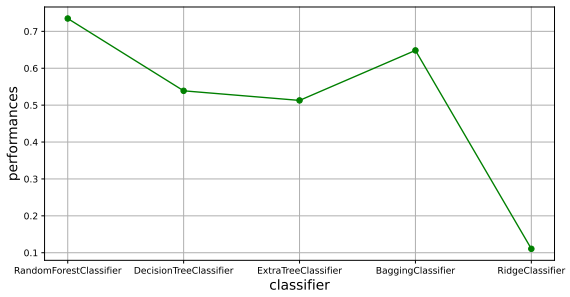

Fig. 5. Accuracy of the best configuration for each model

III. RESULTS

The best model resulting from the previous analysis is the Random Forest classifier.
The accuracy score obtained in the prediction of our test set is about 0.73.

IV. DISCUSSION

This solution offers a baseline to the problem of protein sequences classification. Different weak points arise:

- it could be interesting to consider the entire dataset or at least to consider proteins belonging to the same organism, since composition of sequences could differ with respect to it;
- an interesting point to consider is the imbalanced nature of the problem, since there is no homogeneous distribution of the target classes;
- it could be interesting deepen the metrics used to validate the models, since some metrics like the multilabel confusion matrix are still black-boxes, and difficult to manage since the big number of target classes;
- with the techniques applied the order of the amino acids in the sequences is discarded, it could be useful to keep track of it, also with the use of Neural Networks (e.g. Convolutional Neural Networks or Language models).

REFERENCES

[1] M. Carrigan, “Deep learning with proteins,” 2022.
[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.