ProFAB: OPEN PROTEIN FUNCTIONAL ANNOTATION BENCHMARK

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF INFORMATICS OF

THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

AHMET SAMET ÖZDILEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF BIOINFORMATICS

August 2023

ProFAB: Open Protein Functional Annotation Benchmark

submitted by **Ahmet Samet Özdilek** in partial fulfillment of the requirements for the degree of **Master of Science in Health Informatics Department, Middle East Technical University** by,

Prof. Dr. Banu Günel Kılıç
Dean, **Graduate School of Informatics**

Assoc Prof. Dr. Yeşim Aydın Son
Head of Department, **Health Informatics**

Prof. Dr. Mehmet Volkan Atalay
Supervisor, **Computer Engineering Dept, METU**

Examining Committee Members:

Assist. Prof. Dr. Aybar Can Acar
Health Informatics, METU

Prof. Dr. Mehmet Volkan Atalay
Computer Engineering Dept., METU

Assoc. Prof. Dr. Tunca Doğan
Computer Engineering Dept. Hacettepe University

Date: 21/08/2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Ahmet Samet Özdilek

Signature : ___________________

ABSTRACT

ProFAB: OPEN PROTEIN FUNCTIONAL ANNOTATION BENCHMARK

Özdilek Ahmet Samet

M.Sc Department of Health Informatics

Supervisor: Prof. Dr. Mehmet Volkan Atalay

August, 62 pages

As the number of protein sequences increases in biological databases, computational methods are required to provide accurate functional annotation with high coverage. Although several machine learning methods have been proposed for this purpose, there are still two main issues: (i) construction of reliable positive and negative training and validation datasets, and (ii) fair evaluation of their performances based on predefined experimental settings. To address these issues, we have developed ProFAB: Open Protein Functional Annotation Benchmark, which is a platform providing an infrastructure for a fair comparison of protein function prediction methods. ProFAB provides filtered and preprocessed protein annotation datasets and enables the training and evaluation of function prediction methods via several options. We believe that ProFAB will be useful for both computational and experimental researchers by enabling the utilization of ready-to-use datasets and machine learning algorithms for protein function prediction based on Gene Ontology terms and Enzyme Commission numbers.

Keywords: Bioinformatics, Proteomics, Function Prediction, Classification

ÖZ

ProFAB: PROTEİN FONKSİYONEL ANOTASYONU KIYASLAMA

Özdilek Ahmet Samet

Yüksek Lisans Sağlık Bilişimi Bölümü

Danışman: Prof. Dr. Mehmet Volkan Atalay

Ağustos, 62 sayfa

Protein dizilerinin biyolojik veritabanlarında artmasıyla birlikte, yüksek kapsamda doğru fonksiyonel açıklama sağlamak için hesaplama yöntemleri gereklidir. Bu amaçla birçok makine öğrenimi yöntemi önerilmiş olmasına rağmen, hala iki temel sorun bulunmaktadır: (i) güvenilir pozitif ve negatif eğitim ve doğrulama veri setlerinin oluşturulması ve (ii) önceden belirlenmiş deneysel ayarlar temelinde performanslarının adil bir şekilde değerlendirilmesi. Bu sorunları ele almak için, ProFAB: Açık Protein Fonksiyonel Açıklama Karşılaştırma Platformu'nu geliştirdik. ProFAB, filtrelenmiş ve önceden işlenmiş protein açıklama veri setleri sağlar ve Gene Ontology terimleri ve Enzim Komisyon numaralarına dayalı protein fonksiyonu tahmin yöntemlerinin eğitimini ve değerlendirmesini birkaç seçenek üzerinden mümkün kılar. ProFAB'ın, hazır veri setlerinin ve makine öğrenimi algoritmalarının kullanılmasını sağlayarak hem hesaplama hem de deneysel araştırmacılar için faydalı olacağını düşünüyoruz.

Anahtar Kelimeler: : Biyoenformatik, Proteomic, Fonksiyon Tahmin Etmek, Sınıflandırma

To My Family

ACKNOWLEDGMENTS

I would like to extend my gratitude to several individuals who have played a significant role in the completion of this study. First and foremost, I am indebted to my supervisor, Prof. Dr. Mehmet Volkan Atalay, without whose guidance this study would not have been possible. Under her mentorship, I not only delved into a new field but also acquired invaluable knowledge.

I would also like to express my appreciation to my friends: Arda Eskin from Bioinformatics, Oguz Ulaş Yaman from Chemical Engineering. Their enduring companionship and support. I would like to acknowledge the essential contributions of Ahmet Rifaioğlu, Ahmet Atakan, Gökhan Özsarı. Their unwavering support has played a pivotal role through my progress.

Finally, I would like to express my deepest gratitude to my parents, Zeynep Özdilek and Şevket Özdilek and my sister Elif Özdilek. Their boundless support, love, and unwavering belief in me have been the driving force behind my achievements. Even in the face of adversity, my parents have stood by me, and I recognize that none of this would have been possible without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

EC	Enzyme commission
GO	Gene ontology
BP	Biological Process
MF	Molecular Function
CC	Cellular Component
ProtT5	ProtTrans T5 XL Uniref50
ProtBert	ProtTrans Bert
ML (ml)	Machine learning
NB	Naïve bayes
SVM	Support vector machine
RNN	Recurrent neural network
CNN	Convolutional neural network
GBoost	Gradient boosting
XGBoost	Extreme gradient boosting
LightGBM	Light gradient boosting machine
NLP	Natural language processing
TP	True positive
TN	True negative
FN	False negative
FP	False positive
MCC	Matthew's correlation coefficient

CHAPTER 1

INTRODUCTION

Identification of protein functions is crucial to understand the mechanisms of life and to develop new treatments against diseases. Traditionally, functional annotation of proteins is done by wet-lab experiments; however, experimental procedures are extremely expensive, labor-intensive and take a long time to complete. Therefore, machine learning-based protein function prediction has become an important area of research that aims to aid experimental work. In the field of protein function prediction, the Critical Assessment of Functional Annotation (CAFA) challenge is an initiative where the aim is to evaluate the performances of automated protein function prediction methods [1]. CAFA challenge is organized about every 2 years; however, it is a one-time challenge, and it is not trivial to complete the challenge tasks with the exact same experimental settings afterward. In addition, CAFA challenge does not provide a standard training dataset and data splitting strategies. These issues prevent fair and easy comparison of function prediction methods that are developed in different time frames.

From the machine learning perspective, protein function prediction can be formulated as a multi-label or binary classification problem. In binary classification, a single model is trained for each protein function using positive and negative training datasets. The main challenge in binary classification is the construction of negative training samples, as negative annotations are scarce in databases. Negative training datasets can be created either randomly or based on heuristic algorithms. On the other hand, in multi-class classification, a single model is trained to classify input protein sequences into multiple protein functions. Here, functional annotations of proteins in databases such as UniProtKB can be directly used to train the models, without further processing. However, unlike binary classification, the main problem for multi-label classification is that it should take into account dataset sizes of classes, which introduces potential imbalances between data points. To address these challenges, one effective strategy involves the use of well-designed balanced datasets. By carefully curating and adjusting the positive-negative training data, the inherent imbalances in multi-label classification can be mitigated, enabling prediction algorithms to learn more effectively and to yield improved results. In application of data splitting methods, there is not many studies that take into account different methods other than random splitting like sequence similarity or data annotation time. These considerations can make machine learning models to predict datasets for harsh conditions and gathered in different timelines. As the final problem, lack of application in various models and evaluation criteria. While various studies strive to highlight their methods, these approaches might not be easily comparable, because the suggested models can heavily rely on study-specific parameters and reproducibility couldn't be sustained. In addition, some algorithms may not be suitable for specific datasets, that's why different types of training models should be considered to get the optimal. In terms of performance evaluation, metrics might be poor due to chosen metrics fail to understand data or the model or it can be inadequate that it may over-evaluate or under-evaluate the performance.

To overcome these limitations, we present here an open-source Python framework called ProFAB, Protein Functional Annotation Benchmark. The idea behind ProFAB is to create a platform for a fair comparison of protein function prediction methods on Enzyme Commission (EC) numbers and Gene Ontology (GO) terms. ProFAB provides positive and negative datasets (construction inspired by [2,3]) for each GO term and EC number term based on three data splitting strategies, which are: random, similarity-based and annotation-time-based (i.e. temporal). Also, it has numerous amounts of machine learning techniques and evaluation metrics to compare the models in all way.

As a basis, ProFAB includes four main functionalities:

1. providing ready-to-use training datasets with predefined splitting strategies,
2. dataset featurization & scaling where the goal is to create sequence-based numerical features and to scale them to fit in a specific range,
3. predictive model training for GO term and EC number prediction by providing several classifiers and wrapper functions to train models using the pre-calculated feature vectors and
4. performance evaluation of models by means of various metrics.

ProFAB is designed as a fair and open-access benchmarking platform where researchers can make use of training, validation and test datasets based on predefined settings, and train and evaluate their predictive models. In this way, ProFAB enables a principled assessment and comparison of protein function prediction methods. To provide baseline results to be used for comparison, we also trained several models based on different settings with ProFAB model training functionality.

Background information about CAFA challenge, its deficits and how ProFAB can overcome them to improve protein function prediction are given in Chapter 1. In Chapter 2, literature review is provided for protein function prediction and benchmark studies for computational methods. Protein functions are investigated under two categories: enzyme commission numbers and gene ontology terms. In Chapter 3, methods and datasets provided by ProFAB are explained. This chapter presents how data is collected and prepared for machine learning algorithms. It also provides information regarding data sources and preprocessing steps. Additionally, numerical descriptors for protein sequences, implemented machine learning algorithms and evaluation metrics are presented in terms of tables. In Chapter 4, results for selected cases are given, and discussed in detail. Case selection is made to compare recent and traditional methods for both protein descriptors and machine learning methods. In Chapter 5, conclusions are given. In Appendix chapter, definitions of GO terms, ranges of hyperparameter values of learning algorithms, protein-species statistics and additional results are provided.

CHAPTER 2

BACKGROUND INFORMATION AND RECENT STUDIES

In this chapter, background information about EC number and GO terms and recent studies for their computational predictions will be introduced.

2.1 Enzyme Commission Number

The Enzyme Commission number (EC number) is a system of numerical classification for enzymes, which categorizes enzymes according to the specific chemical reactions they facilitate [4]. It is important to note that EC numbers primarily describe the enzyme-catalyzed reactions themselves rather than the enzymes directly but it is utilized to define enzymes and enzyme genes. It is actually not to classify enzymes but is to classify enzyme-catalyzed reactions.

In total, there are seven main EC classes in this system. These classes and their reaction are as follows.

1. Oxidoreductase reactions	$AH_2 + B = A + AH_2$
	or $AH_2 + B^+ = A + BH + H^+$
2. Transferases reactions	$AX + B = A + BX$
3. Hydrolases reactions	$A\text{-}B + H_2O = AH + BOH$
4. Lyases reactions	$A{=}B + X\text{-}Y = X\text{-}A\text{-}B\text{-}Y$
5. Isomerases reactions	$A{=}B$
6. Ligases reactions	$A + B + NTP = A\text{-}B + NDP + P$
	or $A + B + NTP = A\text{-}B + NMP + PP$
7. Translocases reactions	catalyze the movement of ions

The hierarchical structure of EC nomenclature offers a framework for function prediction (Figure 1). Each class was divided into smaller categories, and each enzyme was assigned a unique four-digit code. The second digit of the code indicates the subclass and provides information about the type of compound or group involved. In the case of oxidoreductases reactions (EC 1), the subclass denotes the type of group in the donor that undergoes oxidation (e.g., 1.1 acts on donors with a CH-OH group, while 1.3 acts on donors with a CH-CH group). The third digit, known as the sub-subclass, specifies the type of reaction. For oxidoreductases, 1.1.1 indicates NAD or NADP as the acceptor, while 1.1.2 indicates cytochrome as the acceptor and etc. The fourth digit serves as a sequential number to identify individual enzymes within a sub-subclass such as 1.1.2.1 is alcohol dehydrogenase. To illustrate a few more examples can be given; for transferases (EC 2), 2.3 indicates acyltransferases reactions while 2.3.1 includes reactions transferring groups other than amino-acyl groups and 2.3.1.11 is the reaction of thioethanolamine S-acetyltransferase. For isomerases reactions (EC 5), 5.4 is for intramolecular transferases and 5.4.1 is transferring acyl groups. 5.4.1.4 is the reaction of D-galactarolactone isomerase. For translocases reactions (EC 7), 7.2 catalyzing the translocation of hydrons. 7.2.2 is linked to the hydrolysis of a nucleoside triphosphate and 7.2.2.9 is transporting P-type Cu^{2+}. This hierarchy shows a progression in EC number classification from broader classes to very specialized ones for both reaction type and their enzymes.

This classification system is still in use and its numbering method not only describes the nature of the catalyzed reaction but also allows for easy inclusion of newly discovered enzymes by assigning the next available number within the appropriate sub-subclass [5].

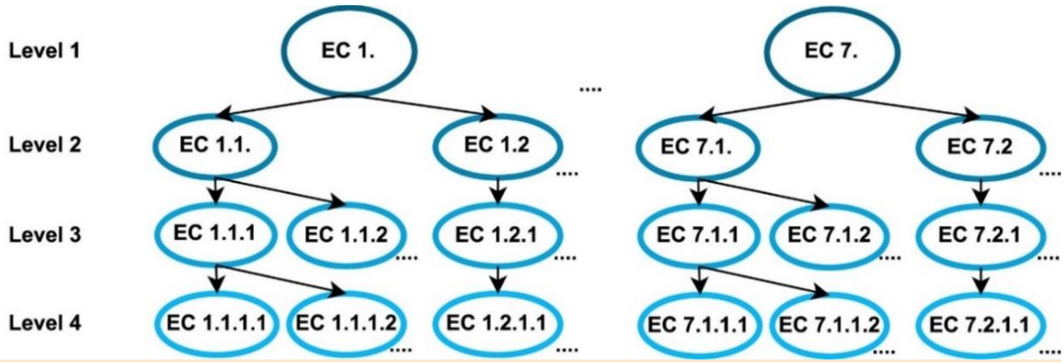

Figure 1: Hierarchical Presentation of Enzyme Commission number.

2.2 Gene Ontology Terms

Gene Ontology (GO) is a linguistic tool to categorize the roles of gene products. It offers comprehensive explanations of biological systems [6]. The primary purpose of GO is to establish a standardized language for describing gene functions. By employing this vocabulary, scientists can effectively communicate and share information regarding the diverse functionalities of genes. The construction of the GO system is built similar to the scheme of EC number hierarchy but GO term annotation considers non-enzymatic processes proteins involved and EC number scheme cannot describe diverse protein interactions [7].

The organization of GO can be grouped into three fundamental aspects: molecular function, biological process, and cellular component. The molecular function aspect focuses on the specific tasks and actions carried out by gene products, such as enzyme activity or receptor binding. Biological process elucidates the series of events and interactions that occur within a living organism, encompassing phenomena like cell division or immune response. Cellular component deals with the physical locations or structures within a cell where gene products are active, like the nucleus or mitochondria [8].

GO terms are interrelated and organized in a specific manner resembling a directed acyclic graph (DAG) structure. The DAG structure employed by GO ensures a hierarchical arrangement of terms, wherein broader concepts encompass narrower ones [9]. For example, GO term GO:0000811 is a specific gene function which is a heterotetrameric protein complex that associates with replication origins where it is required for the initiation of DNA replication while GO:0032991 which is the ancestor of GO:0000811 is a protein-containing complex that is wide-ranged functionality.

2.3 Prediction Methods for EC Number and GO Terms

Various methods and tools have been proposed for enzyme classification. To distinguish whether a given protein sequence belongs to enzyme or not, [10] is using discrete wavelet transform and support vector machine while [11] uses 56-D non-alignment features of proteins (20 of them are percentage of residues, 20 of them are percentage of surface area of each residue, 8 of them are binary value of appearances of newly added heterogens to proteins, 3 of them are alpha, 3_{10} and π helices of secondary structure, and 5 is coming from surface pockets and their size, number of residues and disulfide bonds) and ensemble method (SVM and k-NN each are modeled and both model are used to improve scores) for prediction. To classify enzymes, [12] uses reactions' fingerprints (difference between fingerprints of reactant molecules and product molecules) to assign proteins to their corresponding EC numbers, and [13] uses structure of substrates (similarity in chemical structures between substrates and product molecules) and SVM with binary classification. [14–16] are studies for multi-label classification.

Several protein function prediction methods have been proposed using either one of these perspectives. [17] utilizes three different protein descriptors SPMap, PAAC and CTriad, and applies multi-task feed forward neural network while [2] uses SPMap, Blast-kNN and Pepstats-SVM (peptide statistics combined with SVM) with SVM and kNN. [18] integrates 5 different models that are trained with different features including GO term frequency, sequence alignment, amino acid trigram (LR-3mer, probability of three consecutive AAs), biophysical properties (uses ProFET to extract features from protein sequences in six different categories: local potential features, biophysical quantitative properties, letter-based features, information based statistics AA scale-based features and CTD features) and motifs of different protein families (extracted from a tool InterPro that uses 14 different protein family databases to cover domains and motifs). [19,20] utilize sequence-based features (embedding AA sequence with CNN) and protein-protein interaction network features. After learning these features, [20] combines both sequence and interaction-based features into an ontology-aware loss function that penalizes the output based on whether it violates the GO hierarchy or not.

2.4 Benchmarking Platforms for Bioinformatic Studies

Machine learning algorithms have been widely used in the fields of biology and chemistry to solve problems such as understanding the inner mechanisms of cells, identifying protein functions and developing new drugs. These algorithms are becoming a standard part of the research and development pipelines in the life sciences. In recent years, several methods have been proposed that use machine learning algorithms to tackle problems such as protein structure and function prediction, both of which have direct implications for developing new drugs and understanding disease mechanisms [2,17,21–24].

One of the critical issues in the development of predictive models in machine learning is the construction of reliable training, validation and test datasets. Another related issue is the lack of benchmarking platforms to fairly compare the performances of the generated models. Recently, a few benchmarking platforms have been proposed to provide datasets, machine learning algorithms and evaluation metrics in various areas of bioinformatics. Therapeutics data commons (TDC; [25]) provides ready-to-use biomedical datasets for drug-target prediction, toxicity prediction, de novo molecule generation and antibody development together with several evaluation metrics. MoleculeNet [26] is another benchmarking platform that provides machine learning algorithms, applicable metrics and several datasets about quantum mechanics, cheminformatics and physiology. ChemML [27] is a machine learning and informatics package that enables users to execute machine learning algorithms in the chemical and materials domain. Open Graph Benchmark [28] is a platform that provides datasets for machine learning on graphs coming from different domains such as social, biological and chemical networks. OpenChem [29] is a toolkit designed for computational chemistry and drug design that employs deep learning techniques. Tape [30] consists of five relevant supervised tasks that gauge the efficacy of acquired protein embeddings in areas like protein structure prediction, remote homolog detection and protein engineering, while also being segmented to mimic biologically relevant generalization. DeepPurpose [31] facilitates the creation of models for predicting drug-target interactions by utilizing 15 encoders for compounds and proteins with more than 50 neural architectures. BOOTABLE [32] is another benchmarking platform that allows provides several tools used in bioinformatics such as SPAdes [33] for sequence assembly, sequence alignment and molecular dynamics. It also includes some machine learning algorithms to run benchmarks and various types of datasets taken from GenBank [34], 1000 Genomes Project [35] and etc.

CHAPTER 3

DATA AND METHODS

3.1 An Overview of ProFAB

ProFAB is a benchmarking platform for GO term and EC number prediction. It provides several datasets, featurization and scaling methods, machine learning algorithms, and evaluation metrics. These are collected in four independent modules as shown in Figure 2.

- Dataset module can be used to construct individual datasets for each GO term and EC number.
- Preprocessing module provides three submodules for data preprocessing, i.e., splitting, featurization, and scaling.
- Training module consists of several machine learning algorithms for binary and multilabel classification. In this module, hyperparameter optimization is automatically done to determine the best performing models.
- Evaluation module provides several evaluation metrics to assess the performance of the trained models.

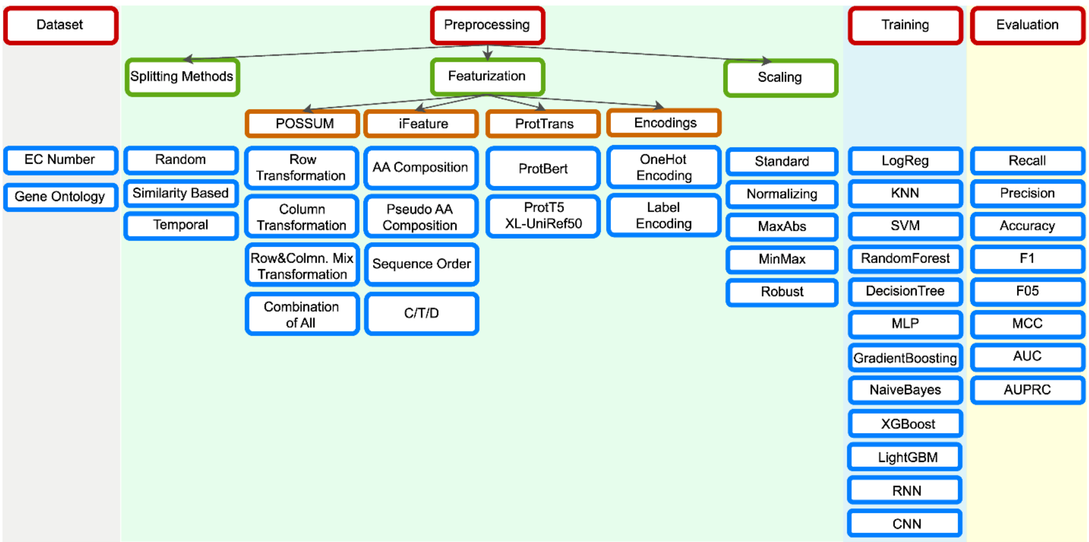

Figure 2: Overview of the modules provided by ProFAB.

3.2 Dataset Construction

One of the main goals of ProFAB is to provide ready-to-use datasets for EC number prediction (273,958 protein annotations for 978 EC numbers) and GO term prediction (1,080,343 protein annotations for 6,678 GO terms). We prepared individual datasets for each function term by collecting protein sequences from UniProtKB-SwissProt, creating positive and negative annotation datasets, and partitioning these datasets based on the chosen splitting methodology. The species names of the proteins utilized in this study are given together with their statistics in Table A4 in Appendix A-Supplementary Tables.

3.2.1 EC Number Datasets (EC-data)

The protein sequences and EC number annotations were obtained from the 2020-05 release of the UniProtKB/SwissProt database. We only used the representative proteins from the UniRef50 clusters [36] to create ProFAB's training, validation, and test datasets in order to avoid bias. Positive and negative training dataset construction was performed based on the proposed methodology in a previous study [3], which is shown in Figure 3. For the positive dataset of a class, we selected proteins that were annotated with that class as the positive samples. The negative training dataset consisted of proteins that were not annotated with the class, as well as their parent proteins and non-enzyme proteins. The non-enzymes dataset comprised proteins with annotation scores of 4 or 5, which lacked EC number annotations in UniProtKB/SwissProt. The statistics of EC-data is given in Table 1.

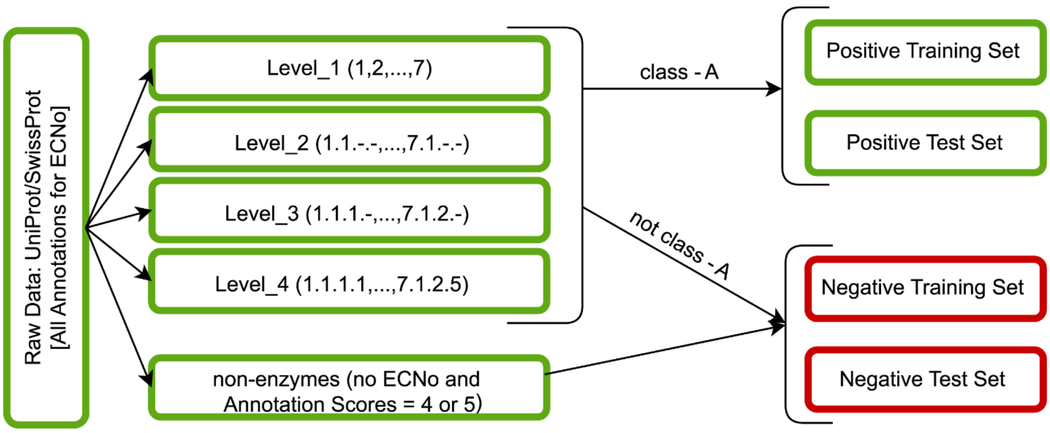

Figure 3: EC Number dataset preparation for an arbitrary class-A: UniProt/SwissProt dataset from UniProtKB was used for positive and negative dataset construction for each EC Number .To construct a positive set for class-A, proteins that are annotated with class-A are used as positive samples. The proteins that are not annotated with class A and its parents and non-enzyme proteins are included in the negative training dataset. Non-enzymes dataset includes proteins that have annotation scores of 4 or 5 and have no EC number annotation in UniProtKB/SwissProt.

Table 1: Statistics of EC-data for main enzyme classes.

Enzymatic Functions	Statistics (# of EC terms, # of proteins, # of annotations)				
		EC Level-1	**EC Level-2**	**EC Level-3**	**EC Level-4**
Oxidoreductases	# of EC terms:	1	21	55	102
	# of proteins:	33,819	32,463	29,294	20,294
	# of annotations:	35,987	33,125	29,766	20,294
Transferases	# of EC terms:	1	9	28	275
	# of proteins:	96,112	95,970	85,385	78,182
	# of annotations:	98,277	98,134	87,531	78,182
Hydrolases	# of EC terms:	1	9	36	159
	# of proteins:	60,821	60,464	54,617	39,040
	# of annotations:	64,089	63,716	57,192	39,040
Lyases	# of EC terms:	1	6	11	76
	# of proteins:	26,045	25,914	23,045	22,061
	# of annotations:	26,125	25,994	23,125	22,061
Isomerases	# of EC terms:	1	6	14	41
	# of proteins:	14,642	14,616	12,822	12,626
	# of annotations:	14,677	14,651	12,857	12,626
Ligases	# of EC terms:	1	5	6	59
	# of proteins:	28,857	28,686	25,919	25,668
	# of annotations:	28,924	28,753	25,986	25,668
Translocases	# of EC terms:	1	6	4	25
	# of proteins:	13,662	12,870	10,107	6,921
	# of annotations:	13,678	12,877	10,114	6,921

3.2.2 GO Term Datasets (GO-data)

ProFAB provides GO term datasets to be employed in protein function prediction methods. GO term annotations were obtained from three different releases of UniProKB/SwissProt (i.e., 2016-01, 2018-01 and 2020-01). Positive and negative training, validation and test datasets were prepared for each GO term. A positive training dataset is created by collecting annotations that are associated with the target GO term and its child terms based on the true path rule. A negative dataset consists of proteins annotated by the sibling terms and the children of the sibling terms of the target GO term. If the size of the positive dataset is less than 1,000, then the size of the negative dataset is adjusted to be twice the size of the positive dataset (by random selection from the overall negatives dataset). When the size of a positive dataset is higher than 1,000, no adjustments were made since the difference between sizes of the positive and negative datasets were usually close to each other. Positive and negative training dataset construction for each GO term is illustrated In Figure 4 on a toy GO DAG. The sizes of test datasets are determined based on the selected splitting strategy.

In random or similarity-based splitting, the sizes of the positive and negative test datasets correspond to 10% of the total datasets. For the temporal splitting method, the test dataset sizes vary among GO terms, since it depends on the number of proteins annotated in the corresponding year(s). The overall statistics for GO term datasets are given in Table 2.

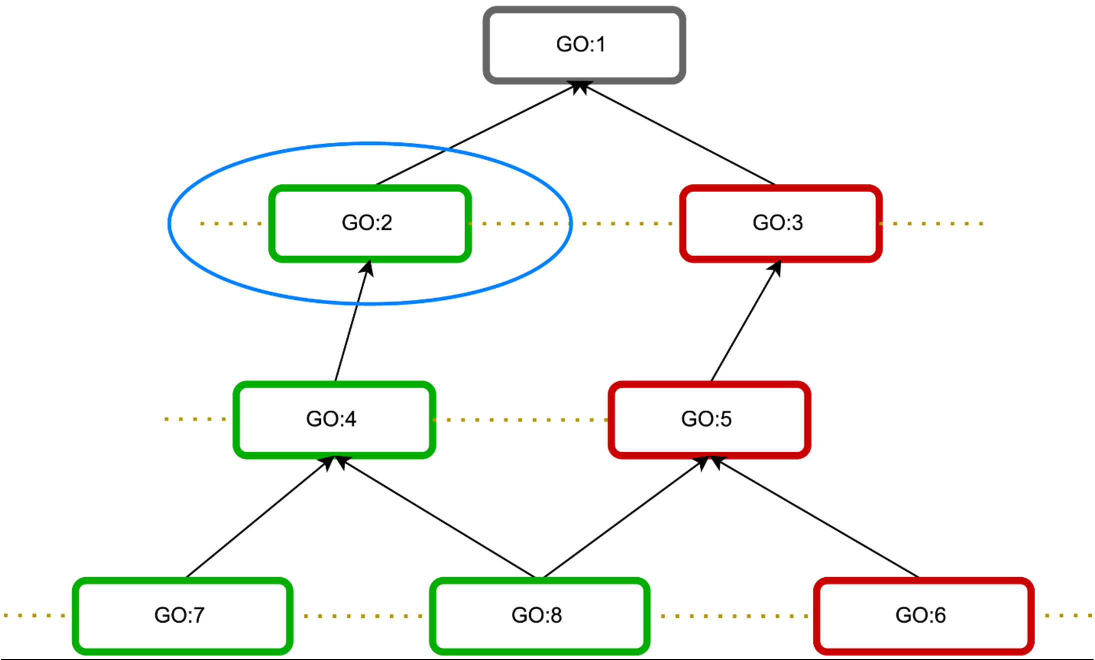

Figure 4: Positive and negative dataset construction for GO:2 on a toy GO DAG. To construct the positive dataset of GO:2, green-colored GO terms are used while GO terms colored with red are used to construct the negative dataset. GO:1 term is not used in dataset constructions. Dashed lines at the same level of GO terms indicate that these GO terms are siblings on GO DAG.

Table 2: Statistics of GO-data for three GO categories.

	Statistics		
GO Categories	**# of Proteins**	**# of GO Terms**	**# of Annotations**
Biological Process	112,839	4,763	509,788
Molecule Function	109,663	1,193	264,783
Cellular Component	110,062	722	305,772

3.3 Data Preprocessing

3.3.1 Splitting

To provide fair data training and to obtain generalized and comparable performance results, three different splitting methods are provided in ProFAB. These are random splitting, similarity-based splitting, and temporal splitting (see Figure 5).

Random split is a baseline splitting method in which overall training datasets are separated into training, validation, and test sets without considering a specific aim. Similarity-based splitting utilizes sequence similarities between proteins to create training/validation/test datasets. The main aim of similarity-based splitting is to avoid bias during training by preventing the model from memorizing the data and not producing predictions over highly similar sequences that fall into training, validation, and test datasets. To achieve this, we only used one sequence (the representative) from each UniRef50 cluster (2020-05 release). UniRef constitutes a group of datasets containing proteins clustered based on sequence similarity (i.e., in UniRef50, proteins that are more than 50% similar to each other are placed in the same cluster). As a third option, temporal split datasets were generated based on different time points. For this, we applied an annotation year-based partition of data (the January release of UniProtKB-SwissProt from each year is used) to generate the temporal split datasets. All GO annotations recorded until 2016 are used as training samples, annotations from the years 2017 and 2018 were placed into the validation dataset, and finally, annotations from 2019 and 2020 were considered as the test dataset.

The issue with the random split is that the test sequences are more likely to be similar to the training sequences since the sequences that are annotated in biological databases at different time points are all mixed. Therefore, the trained models might be biased towards the sequences that are more similar to each other. Temporal split provides a conservative way of evaluation by using the annotations that exist in the source database at a certain (pre-selected) date to train the model and the annotations created after that date to test the model; thus, it reflects that different protein families are annotated at different time points. In other words, it considers the fact that new annotations are done primarily for the less studied proteins in the literature [1]. Due to this reason, temporal split provides a more realistic evaluation.

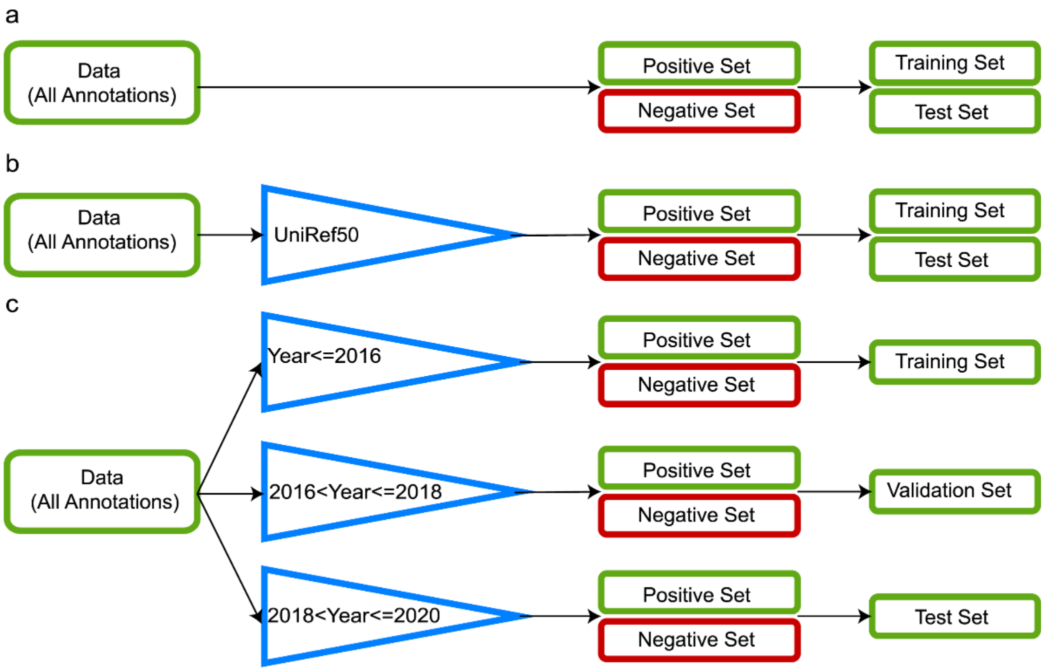

Figure 5: Illustration of splitting methods in ProFAB. All protein annotations for a GO term were considered a positive dataset, and then negative sets are generated after filtering processes. (a) Random split is a splitting method that separates data randomly as training, validation and test datasets. (b) Similarity-based split is a splitting method based on sequence similarity. Here, representative sequences from the UniRef50 dataset were used. (c) Temporal splitting is a splitting based on different time points. To generate training datasets, proteins found in 2016-SwissProt are used, whereas the proteins in 2018 and 2017-SwissProt (but not in 2016-SwissProt) are used to create a test set. Validation datasets include proteins found in 2020 and 2019-SwissProt but not in 2018-SwissProt.

3.3.2 Featurization

ProFAB provides various protein featurization methods to create numerical feature vectors (i.e., descriptors) to be used in the training of predictive models. For this, iFeature [37] and POSSUM [38] feature construction tools (in eight different featurization categories as shown in Figure 1) are integrated into ProFAB to be able to generate on-the-fly feature vectors for a set of query proteins. Using this module, a total of 39 different types of baseline features and two advanced transformer architecture-based features (i.e., ProtT5-XL-Uniref and ProtBert [39]) can be generated. Also, ProFAB provides datasets with pre-calculated and ready-to-use numerical features based on seven different featurization methods. These methods are amino acid composition (AAC), pseudo amino acid composition (PAAC), grouped amino acid composition (GAAC), CTD translation (CTDT), sequence order coupling number (SOCNumber), conjoint triad (CTriad), and k-separated-bigrams POSSUM (kPSSM). These descriptors are explained in detail in Table 3.

Table 3: Protein sequence descriptors provided in ProFAB.

Protein Descriptors	**Definitions**	**# of Dimensions**
Amino Acid Composition (aac)	normalized ratio of the number of amino acids [40]	20
Pseudo Amino Acid Composition (paac)	hydrophobicity & hydrophilicity value and side-chain mass of the amino acid [41]	50
Grouped Amino Acid Composition (gaac)	composition information of groups of amino acids [42]	5
CTD Translation (ctdt)	percentage definition of amino acids that show different properties [43]	39
Quasi-Sequence-Order Coupling Number (socnumber)	normalized ratio of the number of amino acids in 20-D and effect of sequence order in φ-D [44]	60
Conjoint Triad (ctriad)	abstracted features of protein pairs that are classified with respect to amino acids [45]	343
k-Separated-Bigrams POSSUM Vector (kpssm)	relationships among non-adjacent amino acids [46]	400

3.3.3 Scaling

This module is used to standardize or normalize the values in numerical feature vectors. Here, five different methods are provided to scale the vectors (in followings, x is original value of feature and z is scaled value of x):

(i) *standard scaler:* subtracts the mean and scales to unit variance,

$$z = \frac{x - \mu}{s}$$

where u is mean and s is standard deviation of the feature value. Standard scaler is used to scale the values of each feature independently so that mean and standard deviation of each feature are 0 and 1, respectively. Thus, the features have similar scales for each data point which making them suitable for machine learning algorithms.

(ii) *normalizer:* is the same as standard scaler but applied sample-wise,

$$z = \frac{x}{||x||}$$

where ||x|| is the norm of x. Normalizer scales rows to specific norms like L1 or L2. This technique can be useful while each row has its distinct representation which should be conserved.

(iii) *maxabs* scaler: sets the maximum value of each feature to 1.0,

$$z = \frac{x}{max(x)}$$

where max(x) is the maximum absolute value of the feature. This scaler is to preserve sign of data and ensure that relationship between non-zero values are maintained. This could be useful on sparse data.

(iv) *minmax scaler:* limits values of features into a determined range,

$$z = \frac{x - (x)}{(x) \; - (x)}$$

where min(x) is minimum while max(x) is maximum value of the feature. It's useful when to bring all values of data to a common scale within determined min and max values like 0 and 1.

(v) *robust scaler:* scales with respect to quantiles range after removing the median.

$$z = \frac{x - median(x)}{IQR(x)}$$

where median(x) is the median of feature and IQR(x) is interquartile range of the feature. This type of scaler can be considered while data have apparent outliers that may result in unproportional scaled values while applying min-max or standard scaling.

3.4 Model Training and Evaluation Metrics

3.4.1 Training Methods

ProFAB provides machine learning algorithms for constructing binary and multilabel classifiers. After the preprocessing steps (i.e., obtaining the dataset, featurization and scaling), classification methods can be used to train predictive models. At the end of training and hyperparameter optimization processes, it is possible to save the best-performing models and their performance results. The hyperparameters used in fine tuning are shown in Appendix A-Supplementary Tables Table A5.

In ProFAB, eight different machine learning algorithms are provided:

(i) logistic regression: similar to linear regression however it is used to predict categorical values thanks to sigmoid function at the output layer. Its main hyperparameter is “C” which is used to regularize the model between overfitting and underfitting [47]. Although, logistic regression curve is not linear, since its boundary layer is line fixed at 0.5, it is accepted as linear algorithm as seen in Figure 6.

(ii) support vector machine (SVM): is a boundary based learning algorithm. Both linear and non-linear learning can be achieved through its kernel functions. The main hyperparameters: “C” which is regularization term and “kernel” which is a linear or nonlinear function to draw boundaries between samples [48]. Linear based SVM is exemplified in 2-D in Figure 6. A line is separating instances by maximizing the margin between instances and support vector.

(iii) decision tree: is a tree based method. It learns by information gain between different features in hierarchical order. Its main hyperparameter is “max_depth” of trees. As its value is higher up, the model can capture more information but this decreases generalization and causes overfitting [49]. Its demonstration is shown in Figure 7. A hierarchical model that uses input features to make binary (1 or 0) decisions at each node. The hierarchy is constructed based on information gain, which assesses the importance of features and arranges them in order of relevance.

(iv) random forest: is a collection of different decision trees. It is prone to generate more generalizable models [50]. It creates different trees and ensemble them to generate an optimal solution as seen in Figure 7

(v) k-nearest neighbor (k-NN): where each data point is assigned according to nearest points. Query point is assigned a class where representation of nearest neighbors is likely to be found. The main hyperparameter: “k” which is the number of nearest neighbors that is highly data dependable [51].

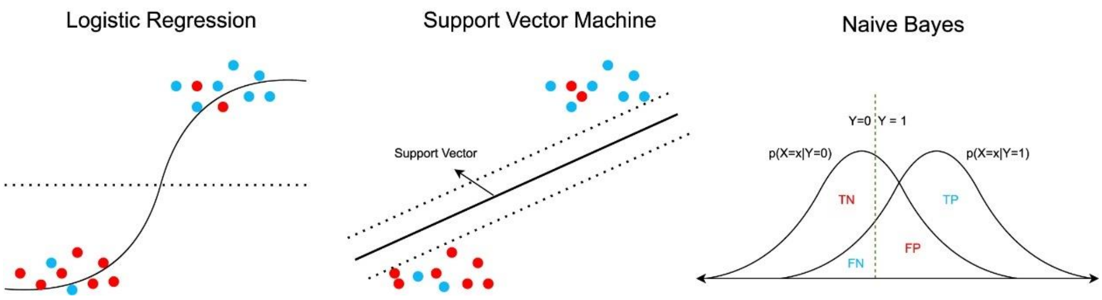

Figure 6: Demonstration of logistic regression, support vector machine and naive bayes learning algorithms in 2-D.

(vi) naive bayes (NB): is a Bayesian method to classify samples according to their likelihood. There isn't any hyperparameter for this method [52]. In Figure 6, distribution of posterior probability and predictions are shown.

(vii) gradient boosting (GBoost): is a method that aims to ensemble weak models to generate a strong one. The model is optimized via stochastic gradient descent. “learning_rate” which is to define the rate of change, “max_depth” of trees. As its value is higher up, the model can capture more information but this decreases generalization and causes overfitting. “loss” is an optimization function [53]. In Figure 9, the working technique of GBoost is given. It creates a tree which is weak at the beginning, then GBoost uses it to generate a strong one, iteratively. It descends the gradient by introducing a new tree.

(viii) multilayer perceptron (MLP): is a linear feed-forward neural network algorithm. The main hyperparameters are “learning_rate” which is to define rate of change and “activation” that is a nonlinear activation function to the output layer [54]. Its network is illustrated in Figure 8. It linearly updates all features to new dimensions until the last layer. The model finally updates parameters by loss function which calculates difference between predicted and true values.

(x) extreme gradient boosting (XGBoost): is the improved version of GBoost. It is more efficient in time and space complexity, and by using early stopping and depth pruning methods, it prevents overfitting. Its main parameters: “learning_rate” which is to define the rate of change, “max_depth” of trees [55].

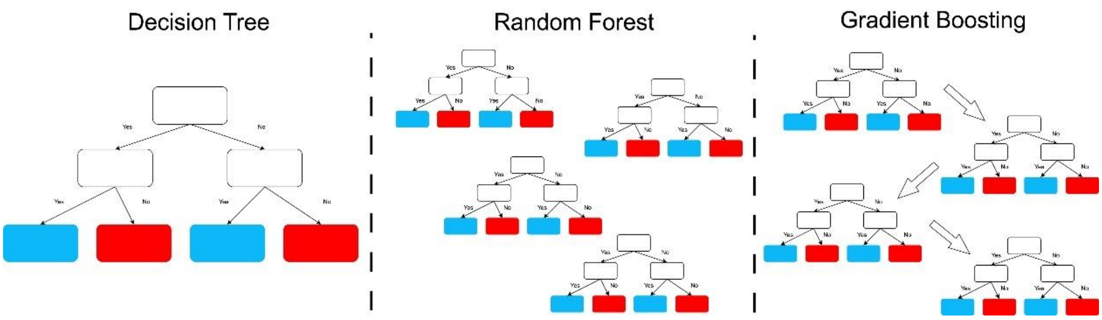

Figure 7: Demonstration of tree-shaped learning algorithms.

(xi) light gradient boosting machine (LightGBM): is another improved gradient boosting algorithm like XGBoost. The difference from XGBoost comes from the growth of leaf nodes. While XGBoost is using breadth-first tree growth which expands all trees at the same, LightGBM expands leaves according to their loss values (leaf-wise method). This enables the model to be more aggressive and faster. Its main parameters: “learning_rate” which is to define the rate of change, “max_depth” of trees [56].

(xii) recurrent neural network (RNN): is a natural language processing algorithm that uses previous information as a context for current information to improve parameters. The main parameters: “hidden_size” that is the number of features in hidden layers, “learning_rate” which is to define the rate of change and, “bidirectional” that is to decide whether RNN is bidirectional, “dropout” which is a ratio to drop some neurons to prevent overfitting [57]. In Figure 8, an example of RNN is given. Input is processed through hidden layer and output layer and decoded data is reintroduced to the hidden layer to improve gradient descent.

(xii) convolutional neural network (CNN): is a convolutional method to make predictions. It maps (filters) the input (or kernel) by sliding over input features (called convolution), and then to prevent complexity and overfitting, filtered features are downsampled by taking average or choosing max value of mapped features. Finally, the outcomes of the pooling layer are flattened to 1-D vectors and they are introduced to fully connected neural networks to learn relationships and to perform classification. The main parameters: “learning_rate” which is to define the rate of change, “kernel_size” that is size of filter for mapping, “stride” which is size of filter striding, “padding” which is number of noneffective features adding both start of input, “pooling” that is down sampling method, “dilation” which is spacing between filter elements, “dropout” which is a ratio to drop some neurons to prevent overfitting [58]. As seen in Figure 8, input is scanned with a kernel that is defined by stride and kernel size values. Resulting feature map is given to the pooling layer to summarize the information and pass the most important feature to the next layer. It achieves this by selecting max value found within a small window or it averages the values. Then, linear layers and an activation function are used to end the process.

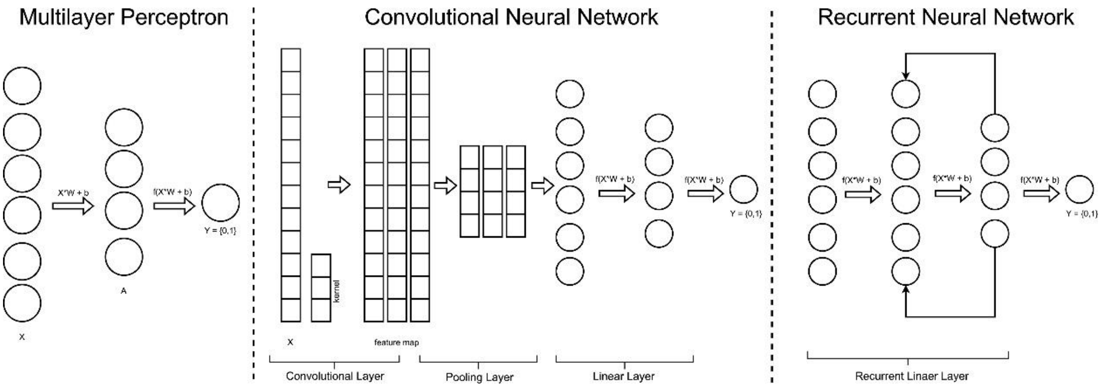

Figure 8: Demonstration of neural network based learning algorithms.

3.4.2 Evaluation Metrics

ProFAB provides several metrics to assess the performance of trained models from different perspectives. These evaluation metrics and their equations are given in Table 4 where the symbol of integral defines area under the curves, TPR is true positive rate (recall) and FPR is false positive rate (the proportion of negative instances that are incorrectly predicted as positive). These metrics are calculated based confusion matrix which is a 2x2 matrix that consists of 4 values:

- true positive (TP) is the number of positive predictions that are actually positive
- true negative (TN) is the number of negative predictions that are actually negative
- false negative (FN) is the number of negative predictions that are actually positive
- false positive (FP) is the number of positive predictions that are actually negative

For multilabel classification, there are multiple confusion matrices, with each matrix representing the classification performance for a different class. Because of that issue, in multilabel classification, all metrics are calculated by *macro* and *micro* ways. While *macro* score is the average of scores from each confusion matrix, *micro* score is calculated from element-wise summation of confusion matrices.

The list of evaluation metrics:

(i) recall: which is also called sensitivity, is the ratio of true positives to total positives, in other words, a measure of a model's ability to remember relevant instances in a dataset. Recall is crucial when false negatives are highly costly. If recall is low, many positive instances will be missed and false negative rate increases.

(ii) precision: is the ratio of true positives to total positives to predicted positives, in other words, proportion of actual relevant data points which the model identified relevant. It is crucial when false positives are highly costly such as in medical diagnosis scenarios that positively predicted values must indicate disease.

(iii) accuracy: is the ratio of the number of correctly predicted values to the total number of instances in a dataset, in other words, a measure to observe how successful the model is at predicting all classes. It is often suitable when the class distribution is balanced. It can be too high if one class dominance is observed in the dataset.

(iv) f-score: which is the harmonic mean of precision and recall. It looks for a balance between these two. It is highly effective for imbalanced data. If β is 1, it is

called F1-score where weights of recall and precision are the same. If high precision is more important than high recall, β values can be lowered to 0.5.

(v) MCC: seeks out quality of data by considering all four results of classification method: TP, TN, FN and FP. It is preferred when class imbalance exists especially when the negative class is dominant. It ranges between -1 and 1 . -1 means negative correlation where opposite predictions are true while 1 indicates positive correlation where values are truly predicted. 0 value indicates random prediction.

(vi) AUPRC: is the value of area under precision-recall curve (PRC). PRC is a curve which indicates trade-off between precision and recall for different probability thresholds. Area under this curve calculates overall performance of the model along these different thresholds. Higher AUPRC means as precision increases, recall keeps increasing too while low values indicate the model is weak to maintain these values at high.

(vii) AUROC: is the value of area under receiver-operating curve (ROC). ROC is a curve which indicates trade-off between true positive rate (recall) and false positive rate for different probability thresholds. It is a remarkable metric when data is balanced or when false positive and false negative have equal importance. If ROC curve is high at the left-top corner, it means high true positive rate low false positive rate. If it is a diagonal line, then the model performs poorly.

Table 4 Performance evaluation metrics provided in ProFAB.

Metric name	**Equation**
Recall	$\frac{TP}{TP + FN}$
Precision	$\frac{TP}{TP + FP}$
Accuracy	$\frac{TP + FP}{FN + TN + TP + FP}$
F1-Score	$\frac{2 * Precision * Recall}{Precision + Recall}$
F0.5-Score	$\frac{(1 + \beta^2) * Precision * Recall}{Precision + Recall}$
MCC	$\frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) + (TN + FP)(TN + FN)}}$
AUPRC	$\int Precision\ d(Recall)$
AUROC	$\int TPR\ d(FPR), FPR = \frac{FP}{FP + TN}$

3.5 ProFAB Structure and Example Runs (Use Cases)

ProFAB was designed for easy and simple application of various workflows. Although ProFAB is regarded as a benchmarking platform, it is constructed upon a framework of adaptable and customizable technology. Each module is designed with its unique purpose, enabling the outputs to be integrated into other tools by only adjusting the module parameters. Selective functions in ProFAB allow the user to implement splitting, scaling, featurization, training and evaluation in a few lines of code. This innovative capability is exemplified through various use cases.

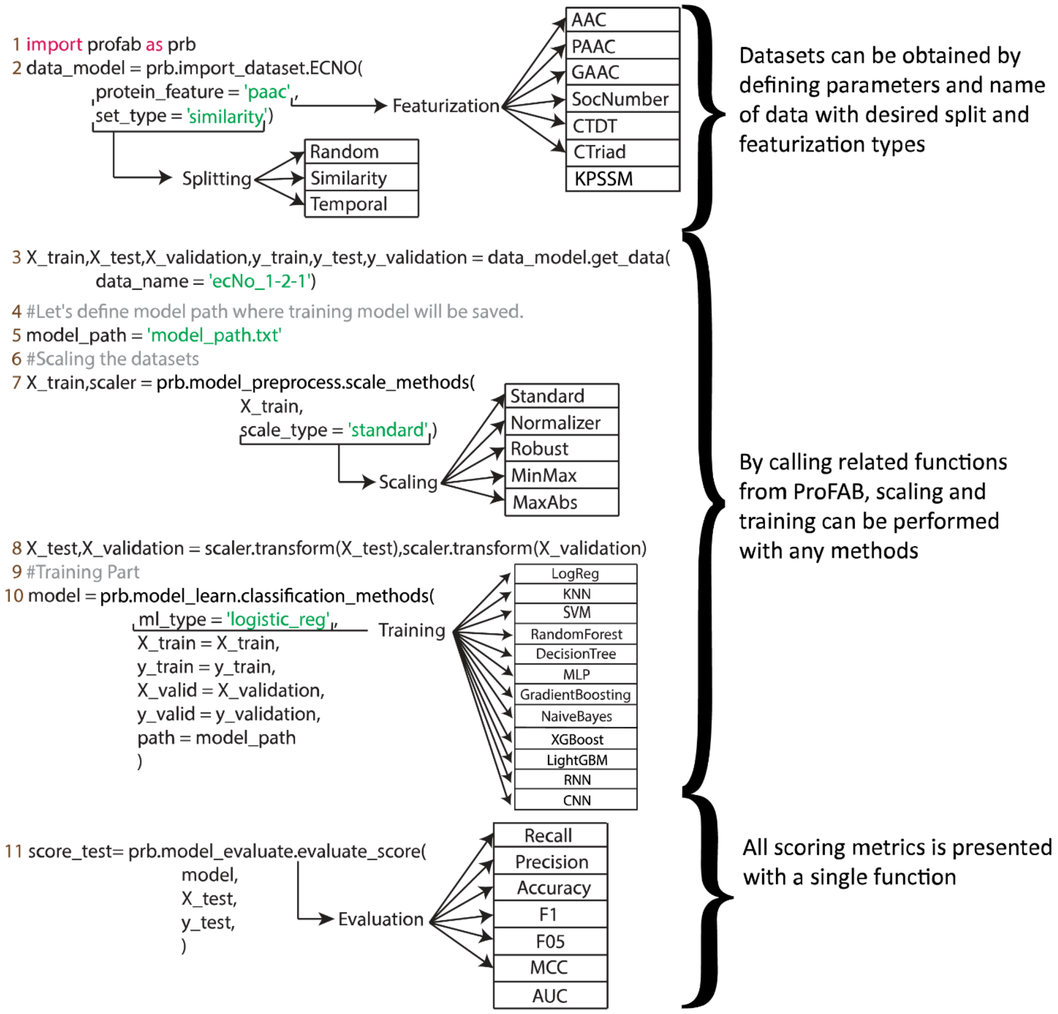

Figure 9: An example code to carry out benchmarking experiments with ProFAB. Multiple methods for dataset preprocessing, and model training/evaluation are provided.

In Figure 9, the usage of ProFAB with possible options to select at different levels of the process was shown step by step. At first, protein feature and splitting method are assigned with function “ECNO” (line number: 2) and then, by providing dataset name, (“ecNo_1-2-1” in this case), train and test sets can be obtained (line number: 3). After receiving data, both train and test can be scaled with ProFAB “scale_methods” function by specifying method name (line numbers: 7-8). To train a model with these datasets, a function “classification_methods” comes into play. By specifying machine learning algorithms with “ml_type” parameter, the model can be trained with that trainer (line number: 10) and then scoring metrics can be evaluated by running function “evaluate_score” (line number: 11).

```
1 import profab as prb
2 data_model = prb.import_dataset.ECNO(protein_feature = 'paac' , set_type = 'similarity')
3 X_train, X_test, X_validation, y_train, y_test, y_validation = data_model.get_data(data_name = 'ecNo_1-2-1')
4 model_path = 'model_path.txt'
5 X_train, scaler = prb.model_preprocess.scale_methods(X_train, scale_type = 'standard')
6 X_test, X_validation = scaler.transform(X_test).scaler.transform(X_validation)
7 model = prb.model_learn.classification_methods(ml_type = 'logistic_reg', X_train = X_train, y_train = y_train,
                                                X_valid = X_validation, y_valid = y_validation, path = model_path)
8 score_test = prb.model_evaluate.evaluate_score(model, X_test, y_test)
9 print(score_test)
{'Precision': 0.804, 'Recall': 0.822, 'F1-Score': 0.813, 'F05-Score': 0.807, 'Accuracy': 0.872, 'MCC': 0.716, 'AUC': 0.860,
'AUPRC': 0.843, 'TP': 255, 'FP': 62, 'TN': 547, 'FN': 55}
```

Figure 10: Complete script of a benchmarking use case in which EC number 1.2.1.

Figure 10 displays the complete script of a benchmarking use case in which EC number "1.2.1.-" is predicted using the PAAC descriptor (line number: 2-3) and the logistic regression algorithm (line number: 7), via similarity-based (line number: 2) dataset splitting and standard scaling (line number: 5). The performance results of the model are printed on the screen and the trained model is saved in the provided path.

In Figure 11, a featurization method is shown. A fasta file, "sample" is introduced to function and a protein descriptor "edb" is given as a parameter. The function converts the fasta file to a numerical descriptor and writes to an output file in text format (line 2).

```
1 from profab.model_preprocess import extract_protein_feature
2 extract_protein_feature('edp', 1,
                          'directory_folder_input_file',
                          'sample')
```

Figure 11: A script of use case that extracts numerical protein features from "sample.fasta" file.

Another prepared use case shown in Figure 12 involves converting protein sequence data into numerical representation, followed by preprocessing steps and the utilization of machine learning algorithms for evaluation. The protein sequence data consists of positive and negative datasets and utilizes transformer NLP model "ProtT5" as a numerical feature. A parameter called "max_len" is set to 100 amino acids, and to ensure dimension consistency, the average is computed by setting "take_avg" to True (line numbers: 2-3) for positive and negative datasets. In the subsequent section, each dataset is loaded using the Python package "numpy," (line numbers: 4-5) and both datasets are then passed to the "ttv_splitter" to divide them into train and test sets (line number: 6). Since the positive and negative data are known, labels are generated automatically. Subsequently, a CNN (Convolutional Neural Network) is employed as the learning algorithm, and the resulting model is saved to the file "model.txt" (line number: 9). Finally, the model and the test set are used for evaluation (line number: 11).

A user can also utilize ProFAB to manage dataset construction as well. Figure 13 illustrates a use case where only the necessary data from the ProFAB database is obtained for a selected protein function. If the user possesses a positive dataset, they can choose to exclusively load the negative dataset from the ProFAB database, if it is available. In this particular scenario, the protein function "EC 1.2.7.-" is employed (line number: 2). By utilizing the ProFAB SelfGet() function, users can load their own dataset (line number: 3) and conveniently download negative labeled samples directly from the ProFAB database. In SelfGet() function, the user can adjust delimiter type and label & name columns with respect to data structure.

```
from profab.model_preprocess import extract_protein_feature
pos_feat = extract_protein_feature('T5XL', 0,
                'example',
                'positive_data',
                 take_avg = True,
                 max_len = 100)
neg_feat = extract_protein_feature('T5XL', 0,
                'example',
                'negative_data',
                take_avg = True,
                max_len = 100)
X_pos = np.loadtxt(pos_feat)
X_neg = np.loadtxt(neg_feat)

from profab.model_preprocess.splitter import ttv_split
X_train,X_test,y_train,y_test = ttc_split(X_pos = X_pos,X_neg = X_neg)

from profab.model_learn.classifications import classification_methods
model = classification_methods(X_train,y_train, ml_type = 'CNN', path = 'model.txt')

from profab.model_evaluate.evaluation_metrics import evaluate_score
test_score = evaluate_score(model,X_test,y_test,isDeep = True)
```

Figure 12: A script that uses transformer model ProtT5 XL Uniref to extract numerical protein features from positive and negative protein sequence data, and then uses this data to train and evaluate a model.

```
1 from profab.import_dataset  import SelfGet, ECNO
2 negative_set = ECNO(label = 'negative').get_data('ecNo_1-2-7')
3 positive_set = SelfGet(delimiter = '\t', name = False, label = False ).get_data('users_1-2-7_positive_set.txt')
```

Figure 13: A script that utilizes ProFAB to obtain negative corresponding protein function data, EC 1.2.7.-. The positive set is loaded with another ProFAB function SelfGet() by defining certain parameters.

CHAPTER 4

RESULTS

This chapter presents and examines the ProFAB benchmark tests, along with a use case. The results are analyzed separately for binary classification and multilabel classification. For binary classification, randomly chosen GO terms are used, whereas for multilabel classification, the predetermined GO-terms based on previous study [59] were utilized as the test cases for analysis. In the following sections, the detailed discussion and visual representations of the results are provided for various cases involving different protein descriptors and learning algorithms. The findings and outcomes of these cases are thoroughly analyzed and explained.

4.1 Binary Classification

Datasets from both EC-data and GO-data were used in benchmark tests. For EC-data, the datasets of oxidoreductase (EC number: 1), hydrolase (EC number: 3) and lyase (EC number: 4) enzyme classes are used. For GO-data, datasets of 15 randomly selected molecular function (Table A3 in Appendix A-Supplementary Tables), cellular component (Table A2 in Appendix A-Supplementary Tables), and biological process terms are used (Table A1 Appendix A-Supplementary Tables). The total number of samples for the GO benchmark tests is 18,620. The tests were carried out using 8 different machine learning algorithms and 6 different featurization methods and 3 different splitting strategies. To scale the datasets, standard scaling was applied. To finetune the learning, randomized search was used. For each algorithm, the hyperparameters and their value ranges are given in Supplementary Table A5 in Appendix A-Supplementary Tables.

Performance results are investigated under three groups based on the utilized protein descriptors. The first group took into account traditional ones; AAC, PAAC, SOCNUMBER and CTDT (trained with MLP, SVM, NB, GBoost, XGBoost, LightGBM algorithms). The second group was evaluated with the features extracted from pre-trained advanced transformer-based models ProtBert and ProtT5-XL-UniRef50 and (trained with RNN, CNN, XGBoost and LightGBM algorithms). In the final group, one hot encoding and label encoding featurization of protein sequences was trained with RNN, CNN, LightGBM, and XGBoost algorithms. In figures, each cell indicates a value of F1 score for that machine learning algorithm (row) and that protein descriptor (column).

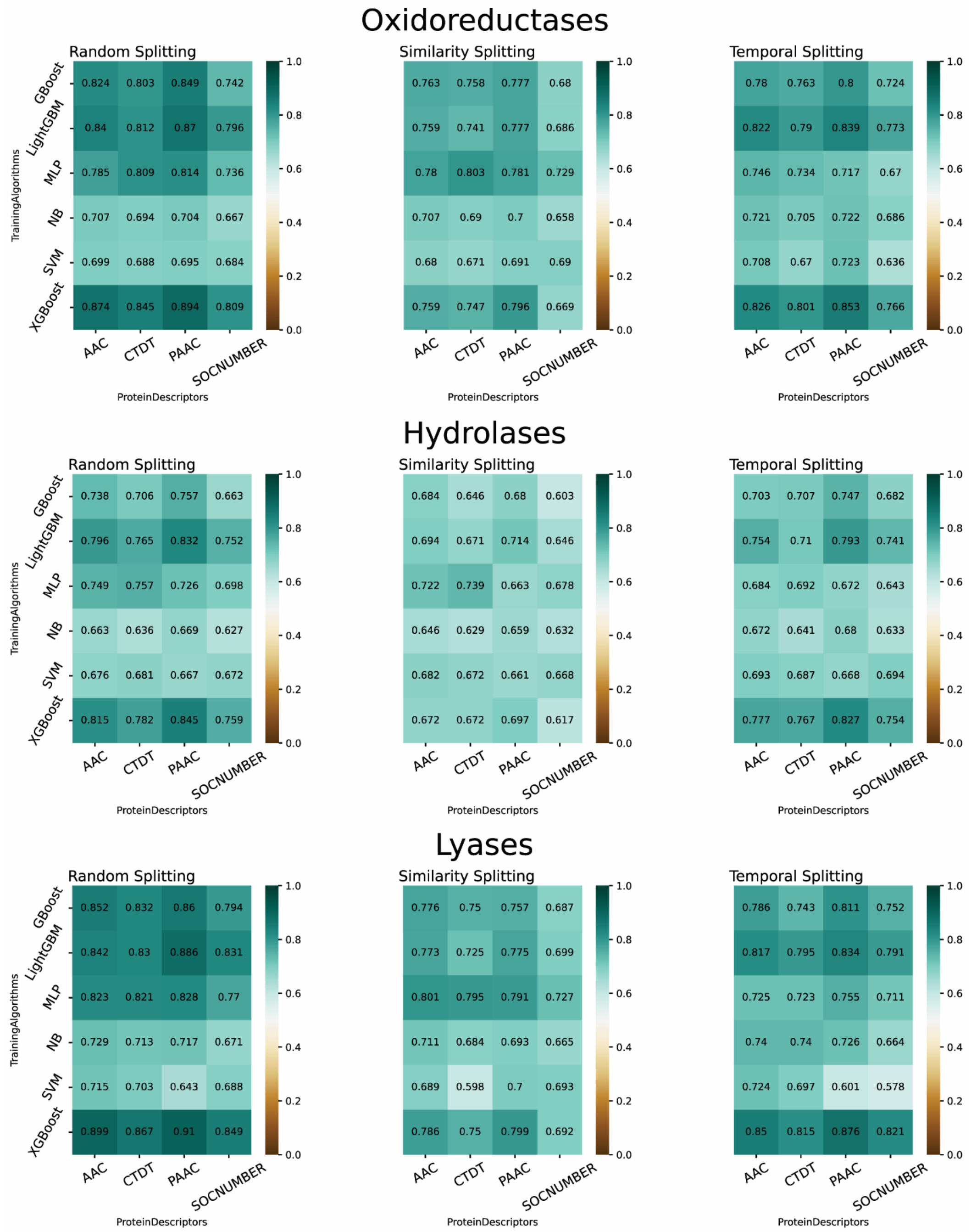

Figure 14: Average F1-score results of the predictive models of main EC classes 1 (Oxidoreductases), 3 (Hydrolases) and 4 (Lyases) based on AAC, PAAC, SOCNumber and CTDT.

According to the performance results of the first two groups (first group Figure 14, Figure 15, and Figure S1 in Appendix B-Additional Results and second group Figure 16, Figure 17 and Figure S2 in Appendix B-Additional Results), the overall F1-score for enzyme class prediction is higher than GO term prediction (0.755 and 0.612, for the first group, 0.887 and 0.785 for the second group, respectively). GO term prediction can generally be considered a more difficult problem compared to enzyme main class prediction since GO terms are hierarchically related to each other with complex relationships, which results in the absence of clear differences between predicted classes. Average performance comparison of GO categories shows that estimating molecule function is easier compared to cellular component and biological process, independent of the selected learning algorithm or protein descriptor (F1-score: 0.672 for MF and 0.582 for BP and CC for the first group), which is in accordance with the literature [60]. The molecular function of a protein is completely dependent on its sequence and structure, whereas there are other critical factors for localization and biological process, such as the protein-protein interactions.

A similar performance difference has been observed only between a few of the main enzyme classes (e.g., ligase F1-score: 0.803, hydrolase: 0.701 for the first group). In terms of featurization methods (i.e., protein descriptors), ProtBert and ProtT5-XL-UniRef50 outperformed traditional descriptors. The overall average F1 score of XGBoost and LightGBM is 0.799 and 0.887 for advanced descriptors and 0.776 and 0.756 for traditional ones, respectively. The performance results of the XGBoost and LightGBM models trained with one-hot encoding and label encoding were calculated as 0.639 and 0.728, respectively. When we investigated the performance of each group, we observed that the performances of the models trained with SOCNUMBER features were the lowest for both EC and GO (GO F1-score: 0.566, EC F1-score: 0.725, respectively), whereas advanced descriptors performed the best (ProtBert GO F1-score: 0.825, ProtBert EC F1-score: 0.916, ProtT5-XL-UniRef50 GO F1-score: 0.745, ProtT5-XL-UniRef50 EC F1-score: 0.857). The detailed results are available in Appendix B-Additional Results in Tables B1-B4.

The average performance comparison in terms of splitting methods shows that random splitting yields overoptimistic performances, most probably due to memorization during model training (GO F1-score: 0.647), whereas similarity and temporal-based splits provide more realistic results (GO F1-score: 0.558 and 0.531, respectively Figure 12 and 14) for the first two groups. Considering the performances in the third group, some of the models failed to learn temporal-based splitting for GO terms (F1-score: 0.459) as these models tend to assign predictions to the majority classes based on their incomplete knowledge learned from imbalanced training data.

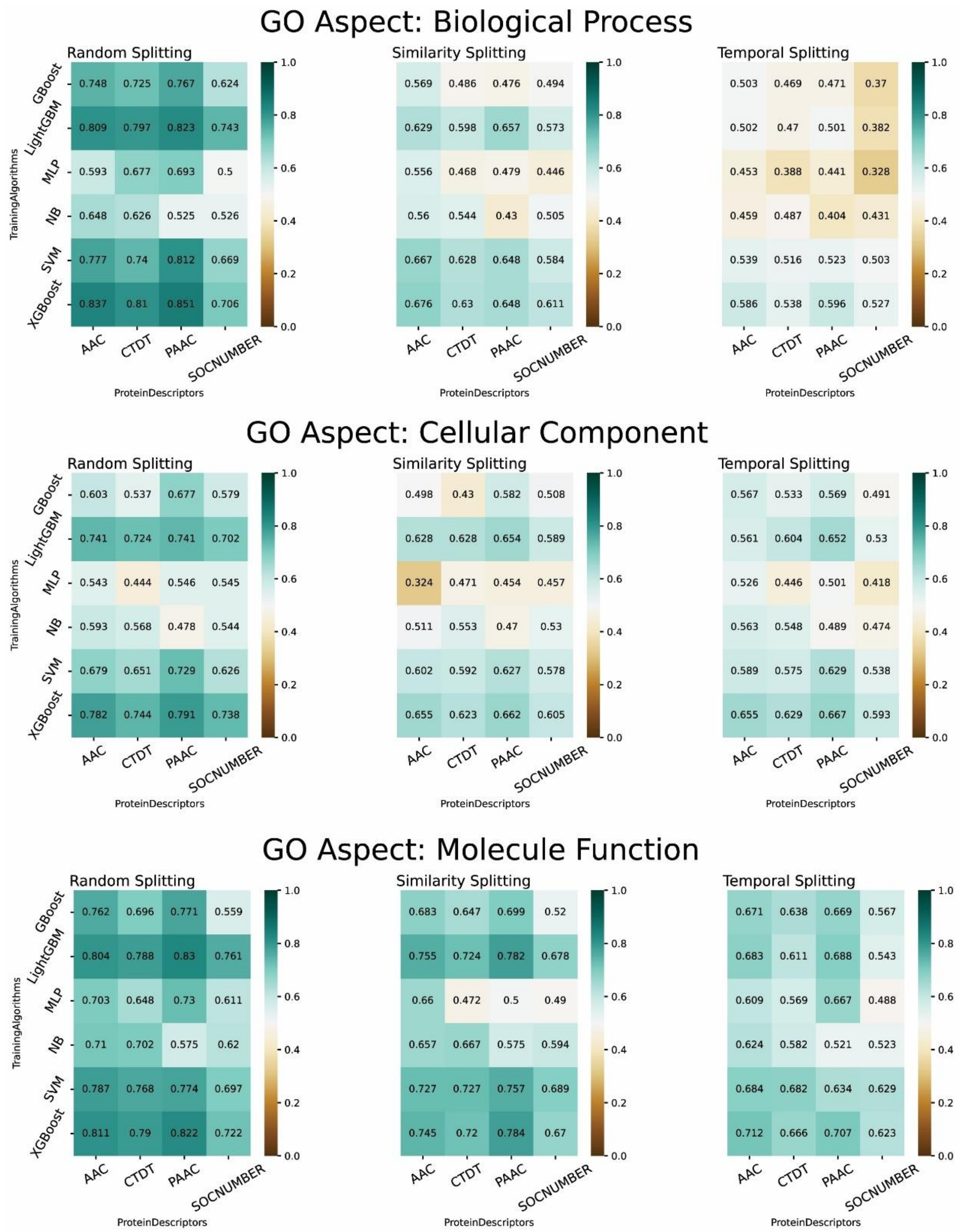

Figure 15: Average F1-score results of the predictive models of 15 randomly selected GO terms from MF, BP and CC GO categories based on AAC, PAAC, SOCNUMBER and CTDT.

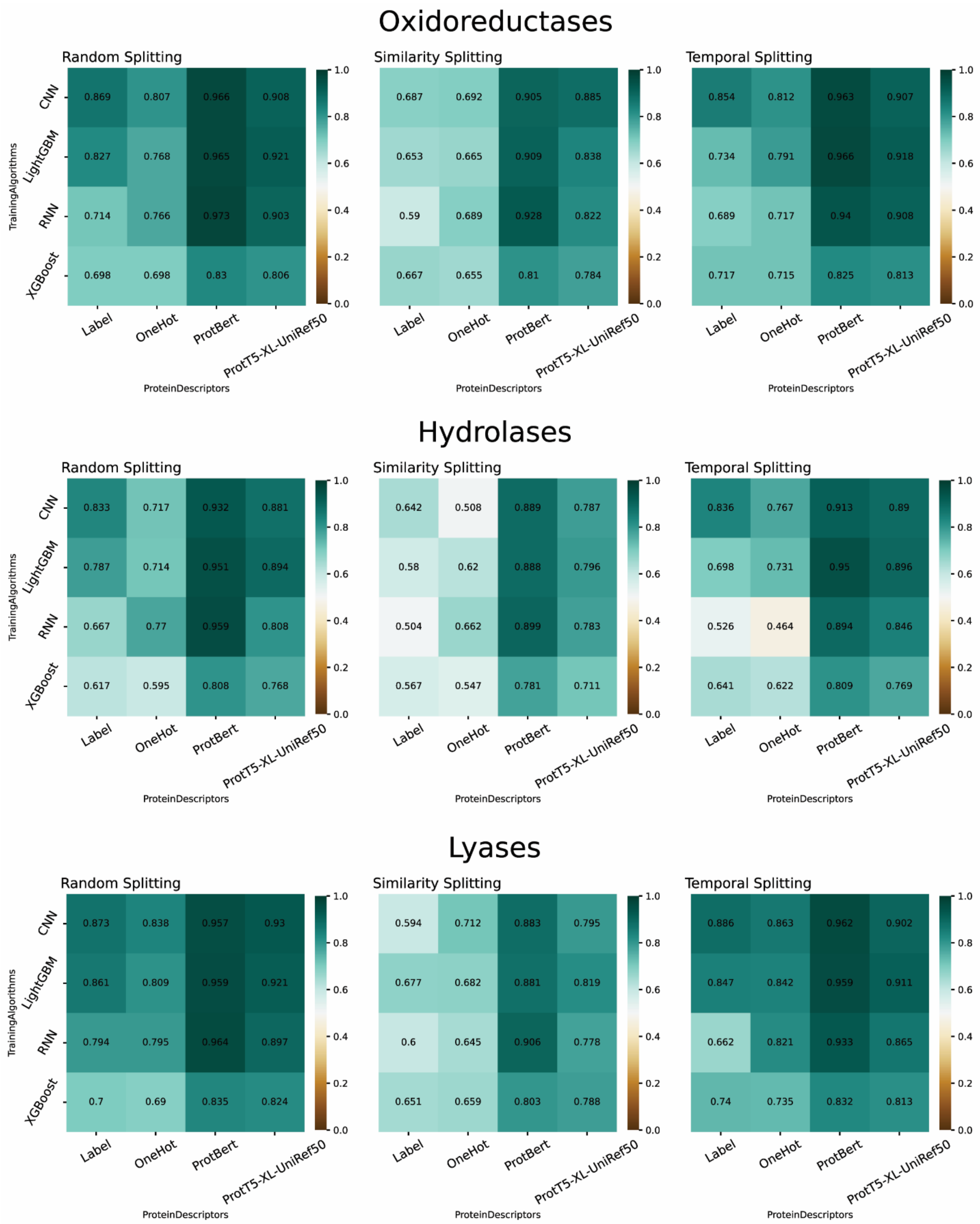

Figure 16: Average F1-score results of the predictive models of main EC classes 1 (Oxidoreductases), 3 (Hydrolases), and 4 (Lyases) based on one-hot encoding, label encoding, ProtBert and ProtT5-XL-Uniref50 protein descriptors.

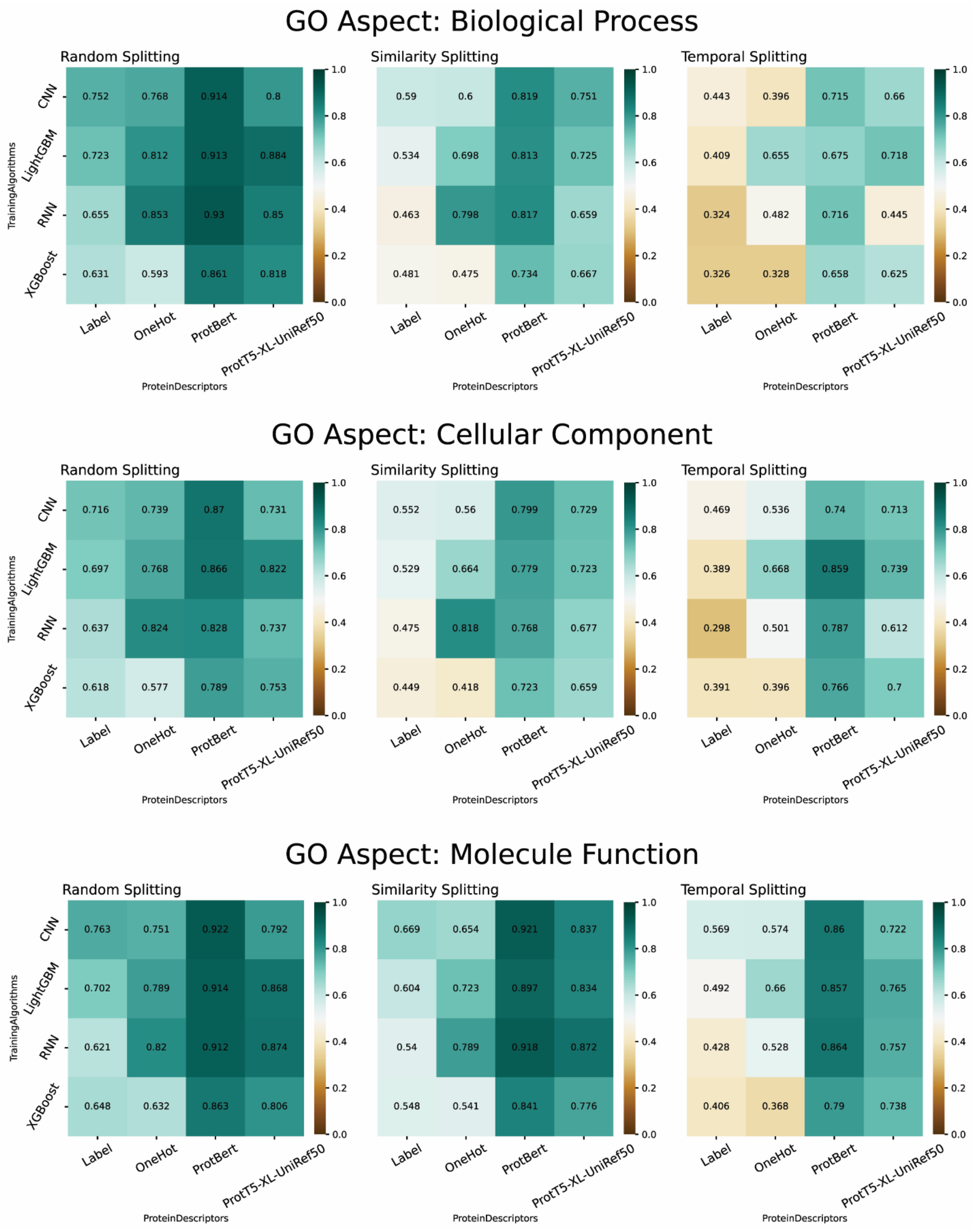

Figure 17: Average F1-score results of the predictive models of 15 randomly selected GO terms from MF, BP and CC GO categories based on one-hot encoding, label encoding, ProtBert and ProtT5-XL-Uniref50 protein descriptors.

When we consider ML algorithms in the first descriptor group, the models trained with XGBoost classifier scored the best in both GO and EC prediction compared to the models trained by the other five algorithms (e.g., GO prediction temporal split-based average F1-score of XGBoost: 0.625, the average of other models: 0.537; EC prediction split-based average F1-score of XGboost: 0.815, the average of other models: 0.734). LightGBM provided the highest results for both EC and GO prediction for the second and third groups (Figure 16 and Figure 17).

4.2 Multilabel Classification

In the context of multilabel classification, the predetermined GO-terms based on previous study [59] were utilized as the test cases for analysis. The statistics for these 11 cases are shown in Table 5. In total, 48 GO-terms and 40,549 proteins found in ProFAB database are used to perform multilabel classification. Positive and negative data were included for each class, and each data point was annotated with at least one class (i.e., each protein is definitely associated with at least one GO term and there is no all-zero label).

Table 5: Statistics for datasets used in multilabel classification benchmarking.

	Statistics	
Datasets	**# of GO Terms**	**# of Proteins**
BP Middle Shallow	5	1,876
BP Middle Specific	5	2,198
BP High Specific	5	8,933
BP High Shallow	3	12,027
CC Low Specific	5	359
CC Middle Normal	2	2,116
CC Middle Shallow	3	2,213
MF High Normal	5	8,584
MF Middle Specific	5	1,553
MF Middle Normal	5	1,766
MF High Shallow	5	17,759
Total	48	40,509

The datasets were generated under three conditions: specific, shallow and normal. Specific datasets include highly specialized GO-terms representing specific functions while shallow datasets consist of GO-terms encompass a wide range of functions, and normal datasets are found in between these two extremes incorporating a moderate level of GO-terms. Low, middle and high in dataset descriptors indicate the size of data from smaller to larger.

The results were derived from protein descriptions using ProtBert and TransT5, along with four distinct learning methods: RNN, XGBoost and LightGBM. These specific choices were made based on their performance in binary classification. CNN was not involved since its failure. Additionally, the datasets were benchmarked using the protein descriptor AAC and machine learning algorithms SVM, LightGBM and XGBoost to compare the test results from [53]. For evaluation, both macro and micro results of F1, precision and recall scores were calculated for each 11 datasets with ProtBert and ProtT5 for each machine learning algorithm and shown in Table 6 (RNN), Table 7 (LightGBM) and Table 8 (XGBoost). In all evaluations, datasets were randomly split.

According to multilabel classification evaluations, the behavior of models is similar to binary classification in terms of performance comparison of GO aspects. GO-terms found in MF category are the easiest to predict while BP category is the most the difficult one (average Macro F1 scores are 0.609, 0.748 and 0.835 for BP, CC and MF, respectively), regardless of machine learning algorithm and protein description methods.

In terms of protein descriptions, ProtT5 gave higher scores than ProtBert, unlike the results of binary classification. (avg Macro F1 score 0.723 for ProtBert and 0.735 for ProtT5). Since both are NLP models, they are more dependent on sequence structure than other physicochemical or compositional methods like PAAC. On the other hand, a conventional method, AAC gives better results than binary classification. The performance improvements may be due to the well-tuned hyperparameters (average macro F1 score 0.742 for AAC, 0.781 for Bert 0.804 for ProtT5 for the LightGBM machine learning algorithm).

Table 6 Precision, recall and F1 scores of RNN with two protein descriptors, ProtBert and ProtT5.

Protein Descriptor	Datasets	Macro Precision	Micro Precision	Macro F1	Micro F1	Macro Recall	Micro Recall
ProtBert	BP Middle Shallow	0.489	0.519	0.484	0.517	0.496	0.515
	BP Middle Specific	0.602	0.596	0.534	0.588	0.520	0.579
	BP High Specific	0.534	0.517	0.494	0.512	0.534	0.508
	BP High Shallow	NaN	0.507	0.400	0.505	0.413	0.503
	CC Low Specific	0.837	0.886	0.823	0.876	0.814	0.867
	CC Middle Normal	0.593	0.793	0.590	0.793	0.601	0.793
	CC Middle Shallow	0.686	0.729	0.684	0.725	0.703	0.721
	MF High Normal	0.858	0.859	0.858	0.859	0.859	0.859
	MF Middle Specific	0.554	0.672	0.557	0.672	0.571	0.672
	MF Middle Normal	0.812	0.785	0.776	0.784	0.769	0.782
	MF High Shallow	0.667	0.643	0.640	0.639	0.645	0.635
ProtT5 XL Uniref 50	BP Middle Shallow	0.457	0.493	0.453	0.481	0.481	0.469
	BP Middle Specific	0.524	0.554	0.502	0.546	0.490	0.540
	BP High Specific	0.494	0.487	0.481	0.481	0.495	0.476
	BP High Shallow	NaN	0.441	0.345	0.438	0.352	0.435
	CC Low Specific	0.736	0.733	0.686	0.733	0.677	0.733
	CC Middle Normal	0.615	0.767	0.556	0.767	0.542	0.767
	CC Middle Shallow	0.589	0.596	0.579	0.595	0.630	0.594
	MF High Normal	0.886	0.878	0.875	0.878	0.871	0.878
	MF Middle Specific	0.695	0.781	0.703	0.781	0.716	0.781
	MF Middle Normal	0.902	0.877	0.890	0.875	0.883	0.873
	MF High Shallow	0.717	0.693	0.692	0.690	0.684	0.687

When considering performances of ProFAB and PROBE based on protein descriptor AAC and machine learning algorithm SVM, ProFAB outperforms PROBE in these cases (Table 9). The average macro F1 and precision scores for ProFAB are 0.787 and 0.805, whereas for PROBE, they are 0.258 and 0.393 respectively. This outstanding performance was observed thanks to ProFAB's modifiable structure and its well-designed positive set construction (see Section 3.2.2). When considering data size, ProFAB is better at middle-sized due to lower difference in positive and negative sets. Predicting high-sized datasets is the most challenging, possibly negative instances may overwhelm positive ones.

Table 7: Precision, recall and F1 scores of LightGBM with two protein descriptors, ProtBert and ProtT5.

Protein Descriptor	Datasets	Macro Precision	Micro Precision	Macro F1	Micro F1	Macro Recall	Micro Recall
ProtBert	BP Middle Shallow	0.807	0.846	0.671	0.732	0.589	0.644
	BP Middle Specific	0.922	0.906	0.798	0.828	0.721	0.763
	BP High Specific	0.575	0.560	0.572	0.558	0.569	0.555
	BP High Shallow	0.794	0.836	0.671	0.723	0.584	0.637
	CC Low Specific	0.871	0.925	0.801	0.871	0.757	0.822
	CC Middle Normal	0.821	0.938	0.742	0.896	0.691	0.857
	CC Middle Shallow	0.972	0.959	0.933	0.941	0.900	0.924
	MF High Normal	0.919	0.919	0.915	0.916	0.912	0.913
	MF Middle Specific	0.895	0.895	0.800	0.846	0.729	0.802
	MF Middle Normal	0.964	0.965	0.916	0.916	0.874	0.873
	MF High Shallow	0.847	0.835	0.768	0.761	0.704	0.699
ProtT5	BP Middle Shallow	0.834	0.860	0.710	0.769	0.634	0.695
	BP Middle Specific	0.908	0.914	0.828	0.857	0.769	0.806
	BP High Specific	0.571	0.554	0.559	0.545	0.548	0.535
	BP High Shallow	0.809	0.855	0.723	0.780	0.656	0.717
	CC Low Specific	0.821	0.868	0.715	0.795	0.667	0.733
	CC Middle Normal	0.745	0.936	0.726	0.907	0.709	0.880
	CC Middle Shallow	0.970	0.970	0.937	0.948	0.909	0.928
	MF High Normal	0.941	0.941	0.942	0.942	0.942	0.943
	MF Middle Specific	0.918	0.943	0.917	0.943	0.917	0.943
	MF Middle Normal	0.981	0.976	0.959	0.958	0.940	0.941
	MF High Shallow	0.873	0.860	0.829	0.820	0.790	0.784

In the view of machine learning methods, CNN showed notable performance in binary classification while CNN failed to predict all classes logically for multilabel classification, and it is generally lower than RNN and LightGBM results for both protein description models (avg. ProtBert and ProtT5 macro F1 score 0.781 and 0.804 for LightGBM and 0.622 and 0.615 for RNN). Despite these results, CNN can demonstrate its power by changing certain parameters such as learning rate, kernel size, hidden layer size. XGBoost remains the worst option among these algorithms because it is the slowest one (with GPU) and due to its lower performance in both binary and multilabel classification.

Table 8: Precision, recall and F1 scores of XGBoost with two protein descriptors, ProtBert and ProtT5.

Protein Descriptor	Datasets	Macro Precision	Micro Precision	Macro F1	Micro F1	Macro Recall	Micro Recall
ProtBert	BP Middle Shallow	0.796	0.829	0.683	0.741	0.609	0.669
	BP Middle Specific	0.902	0.882	0.779	0.797	0.698	0.727
	BP High Specific	0.556	0.537	0.550	0.535	0.545	0.534
	BP High Shallow	0.774	0.813	0.659	0.710	0.576	0.630
	CC Low Specific	0.843	0.892	0.745	0.805	0.678	0.733
	CC Middle Normal	0.768	0.942	0.705	0.895	0.659	0.853
	CC Middle Shallow	0.964	0.958	0.921	0.931	0.883	0.906
	MF High Normal	0.911	0.911	0.907	0.908	0.903	0.905
	MF Middle Specific	0.895	0.893	0.812	0.859	0.750	0.828
	MF Middle Normal	0.962	0.955	0.917	0.915	0.878	0.877
	MF High Shallow	0.833	0.819	0.752	0.745	0.686	0.684
ProtT5	BP Middle Shallow	0.815	0.852	0.674	0.736	0.588	0.649
	BP Middle Specific	0.894	0.900	0.805	0.834	0.739	0.777
	BP High Specific	0.553	0.537	0.545	0.530	0.537	0.525
	BP High Shallow	0.783	0.834	0.693	0.759	0.627	0.696
	CC Low Specific	0.807	0.842	0.690	0.771	0.638	0.711
	CC Middle Normal	NaN	0.934	0.698	0.888	0.661	0.846
	CC Middle Shallow	0.962	0.959	0.932	0.941	0.905	0.924
	MF High Normal	0.935	0.935	0.937	0.938	0.939	0.940
	MF Middle Specific	0.947	0.962	0.911	0.936	0.880	0.911
	MF Middle Normal	0.979	0.976	0.948	0.949	0.920	0.923
	MF High Shallow	0.859	0.841	0.811	0.800	0.769	0.763

As for the final comparison, types of datasets can be discussed. According to benchmarks, normal datasets tend to give higher precision scores than others. The reason behind might be that since specific GO-terms are more prone to memorization, test prediction could underfit. On the other hand, shallow datasets cover a high range of functions where proteins are less similar to each other. This consequent caused the model to be underfitting with the training sets.

Table 9: Precision, recall and F1 scores of two studies: ProFAB and PROBE with machine learning algorithm, SVM and protein descriptor, AAC.

Protein Descriptor	Datasets	Macro Precision	Micro Precision	Macro F1	Micro F1	Macro Recall	Micro Recall
ProFAB	BP Middle Shallow	0.719	0.767	0.682	0.734	0.649	0.703
	BP Middle Specific	0.787	0.805	0.764	0.787	0.746	0.770
	BP High Specific	0.566	0.557	0.591	0.581	0.619	0.608
	BP High Shallow	0.687	0.720	0.659	0.689	0.634	0.660
	CC Low Specific	0.792	0.804	0.772	0.813	0.768	0.822
	CC Middle Normal	0.828	0.918	0.754	0.898	0.727	0.880
	CC Middle Shallow	0.915	0.915	0.899	0.909	0.889	0.902
	MF High Normal	0.920	0.919	0.920	0.921	0.921	0.924
	MF Middle Specific	0.884	0.916	0.881	0.911	0.879	0.906
	MF Middle Normal	0.962	0.963	0.963	0.959	0.965	0.955
	MF High Shallow	0.793	0.784	0.771	0.762	0.751	0.742
PROBE	BP Middle Shallow	0.188	0.635	0.143	0.369	0.149	0.306
	BP Middle Specific	0.471	0.645	0.185	0.274	0.156	0.203
	BP High Specific	0.702	0.675	0.340	0.356	0.258	0.249
	BP High Shallow	0.130	0.652	0.041	0.101	0.039	0.068
	CC Low Specific	0.075	0.290	0.081	0.138	0.180	0.145
	CC Middle Normal	0.172	0.661	0.130	0.453	0.146	0.391
	CC Middle Shallow	0.252	0.559	0.116	0.179	0.105	0.136
	MF High Normal	0.869	0.869	0.835	0.841	0.811	0.815
	MF Middle Specific	0.486	0.679	0.333	0.390	0.371	0.360
	MF Middle Normal	0.496	0.475	0.389	0.446	0.468	0.458
	MF High Shallow	0.481	0.840	0.241	0.353	0.194	0.224

Table 10: Precision, recall and F1 scores of with ml types: LightGBM and XGBoost and protein descriptor, AAC.

Protein Descriptor	Datasets	Macro Precision	Micro Precision	Macro F1	Micro F1	Macro Recall	Micro Recall
LightGBM	BP Middle Shallow	0.796	0.821	0.691	0.740	0.621	0.674
	BP Middle Specific	0.867	0.889	0.781	0.813	0.712	0.748
	BP High Specific	0.313	0.435	0.292	0.368	0.287	0.319
	BP High Shallow	0.743	0.769	0.605	0.638	0.512	0.545
	CC Low Specific	0.794	0.829	0.724	0.791	0.696	0.756
	CC Middle Normal	0.732	0.913	0.712	0.888	0.694	0.865
	CC Middle Shallow	0.963	0.964	0.901	0.914	0.847	0.870
	MF High Normal	0.906	0.905	0.905	0.907	0.904	0.908
	MF Middle Specific	0.897	0.920	0.878	0.911	0.862	0.901
	MF Middle Normal	0.958	0.957	0.935	0.932	0.914	0.909
	MF High Shallow	0.815	0.808	0.742	0.736	0.681	0.676
XGBoost	BP Middle Shallow	0.780	0.808	0.672	0.722	0.598	0.653
	BP Middle Specific	0.882	0.886	0.801	0.813	0.734	0.752
	BP High Specific	0.518	0.504	0.510	0.497	0.501	0.489
	BP High Shallow	0.685	0.706	0.578	0.616	0.505	0.546
	CC Low Specific	0.798	0.838	0.685	0.756	0.626	0.689
	CC Middle Normal	NaN	0.915	0.696	0.881	0.677	0.850
	CC Middle Shallow	0.938	0.940	0.878	0.891	0.826	0.848
	MF High Normal	0.902	0.902	0.902	0.903	0.901	0.903
	MF Middle Specific	0.911	0.934	0.877	0.909	0.848	0.885
	MF Middle Normal	0.966	0.966	0.933	0.932	0.903	0.900
	MF High Shallow	0.782	0.773	0.716	0.710	0.662	0.657

CHAPTER 5

DISCUSSIONS

In this section, binary and multilabel classification benchmarking results are discussed in terms of the data task type, splitting method and training algorithms.

For binary classification, splitting methods have crucial effects on GO-terms prediction while EC number prediction seems not to be influenced. The reason for this consequence can be because of data size and annotation of data instances. As mentioned in Chapter 4, for each splitting methods, data size of EC number (~30,000) is much higher than each 15 GO-term datasets (~2000). Therefore, learning models are more sensitive to individual differences in dataset when approaching GO-terms prediction. The same reason is valid for protein descriptor methods. As seen in Figure 14 and Figure 15, protein descriptors have more influence on the model performances for GO-term datasets than EC number datasets. This is because models may be sensitive to weaknesses of numerical features due to lower data size. Though, this difference is closed when transformer models come in play. It is because transformer models are better at capturing both positional and attention information among amino acids. Hence, training algorithms can find a sharper solution to predict protein functions. In terms of splitting methods on GO-terms, sequence similarity and temporal splitting have demonstrated their effectiveness. Sequence similarity means all proteins are at most 50% similar to each other, therefore the model tries to understand more complex data than random split method, and it is hard especially when data size is low, such as around 1000. In terms of temporal splitting, as annotations of proteins in training and test sets were made at different timelines, models try to make predictions for unseen proteins beforehand. The other reason why EC number training is not affected so much for temporal splitting can be that most of the annotations of proteins have been done earlier For example, # of proteins annotated for Oxidoreductase before 2016 ~32000, while it is ~1000 between 2018 and 2020. Consequently, complexity among GO terminology and difference between dataset sizes might have affected the performance of the models between EC and GO-terms with respect to splitting methods and protein descriptors. In terms of machine learning methods, advanced methods which are XGBoost, LightGBM, CNN and RNN achieved better performance than others. Notably, XGBoost and LightGBM (new versions of tree based algorithms) have surpassed other traditional methods thanks to their pruning and advanced leaf expanding strategies. On the other hand, CNN and RNN which are neural network models are better at capturing information than the conventional learning methods.

For the results of multilabel classification, it can be indicated that models are better at catching higher precision scores than recall. The reason why precision is important is given at Section 3.4.2. It means that models can predict actual true positives by not increasing false positive rates, therefore protein functions were predicted truly.

Another point to discuss is "NaN" values in the macro precision score. "NaN" signifies that at least one class has either zero true positives (TP) or zero true negatives (TN) in its predictions. In other words, there is no correct prediction for that particular class. This may be caused by overfitting or class imbalance. Since GO-terms were selected with balanced data, overfitting remains to be the only problem. This problem can be solved with decreasing the number of leaves and depths of XGBoost which may simplify the model and prevent overfitting. For RNN, batch number and dropout level can be changed which are modifiable in ProFAB open-source code. On the other hand, some datasets may not be suitable for some machine learning algorithms. For example, CNN failed to learn in all multilabel classification datasets (gave micro MCC score nearly zero), that is why ProFAB provides many different training algorithms. A user can find the algorithm that best suits her/his data and problem.

CHAPTER 6

CONCLUSION

ProFAB is an open-access, comprehensive, and flexible benchmarking platform for protein function prediction, based on GO terms and EC numbers. It provides ready-to-use datasets for training, validation, and fair testing of prediction methods. In ProFAB, a total of 7,656 datasets are provided for 6,678 GO terms and 978 EC numbers. Available featurization methods can be used to construct numerical feature vectors for protein sequences for a selected dataset.

ProFAB aims to provide a platform where it is possible to examine the performance results of the baseline protein function prediction models. Moreover, users may utilize the provided datasets and splitting settings to evaluate their own methods. We hope ProFAB will be useful for researchers in our field, where they train their models using the datasets and splits in our framework and compare results with our baselines of different varieties, in terms of featurization and classification algorithms. In this way, fair evaluation can be achieved in protein function prediction, which is the ultimate aim of ProFAB.

ProFAB is an extendable project open to further development. In this context, one of the planned developments is the addition of drug-target protein interaction prediction as a new task. Another extension will be the addition of protein structure-based features and protein embeddings (i.e., features learned by deep representation learning models [59]) to the platform, along with the existing sequence-based features.

The key points of this study are as follows:

- There is a lack of benchmarking platforms in protein function prediction area based on Enzyme Commission (EC) numbers and Gene Ontology (GO) terms.
- ProFAB is a benchmarking platform that provides positive and negative datasets for each EC number and GO term based on three data splitting strategies which are random, similarity-based, and annotation-time-based.
- ProFAB includes four main functionalities (i) providing ready-to-use training datasets with predefined splitting strategies (ii) dataset featurization & scaling where the goal is to create sequence-based numerical features and to scale them to fit in a specific range, (iii) predictive model training for GO term and EC number prediction by providing several classifiers and wrapper functions to

train models using the pre-calculated feature vectors, and (iv) performance evaluation of models by means of various metrics.

- ProFAB is designed as a fair and open-access benchmarking platform where researchers can make use of training, validation, and test datasets based on predefined settings, and train and evaluate their predictive models.
- ProFAB provides performance results based on different settings to be used for comparison between datasets, preprocessing approaches (i.e., splits, featurization and scaling), and algorithms in the context of protein function prediction.

REFERENCES

1. Zhou N, Jiang Y, Bergquist TR, et al. The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. Genome Biol. 2019; 20:

2. Rifaioglu AS, Doğan T, Saraç ÖS, et al. Large-scale automated function prediction of protein sequences and an experimental case study validation on PTEN transcript variants. Proteins 2018; 86:135–151

3. Dalkiran A, Rifaioglu AS, Martin MJ, et al. ECPred: a tool for the prediction of the enzymatic functions of protein sequences based on the EC nomenclature. BMC Bioinformatics 2018; 19:

4. Kotera M, Okuno Y, Hattori M, et al. Computational Assignment of the EC Numbers for Genomic-Scale Analysis of Enzymatic Reactions. J. Am. Chem. Soc. 2004; 126:16487–16498

5. Tipton K, Boyce S. History of the enzyme nomenclature system. Bioinformatics 2000; 16:34–40

6. Carbon S, Douglass E, Dunn N, et al. The Gene Ontology Resource: 20 years and still GOing strong. Nucleic Acids Res. 2019; 47:D330–D338

7. Ashburner M, Ball CA, Blake JA, et al. Gene Ontology: tool for the unification of biology. Nat. Genet. 2000; 25:

8. Thomas PD. The Gene Ontology and the Meaning of Biological Function. Gene Ontol. Handb. 2017; 19–20

9. Gaudet P, Škunca N, Hu JC, et al. Primer on the gene ontology. Methods Mol. Biol. 2017; 1446:25–37

10. Qiu J-D, Luo S-H, Huang J-H, et al. Using support vector machines to distinguish enzymes: Approached by incorporating wavelet transform. J. Theor. Biol. 2009; 256:625–631

11. Davidson NJ, Wang X. Non-Alignment Features Based Enzyme/Non-Enzyme Classification Using an Ensemble Method. 2010 Ninth Int. Conf. Mach. Learn. Appl. 2010; 546–551

12. Hu Q-N, Zhu H, Li X, et al. Assignment of EC Numbers to Enzymatic Reactions with Reaction Difference Fingerprints. PLoS ONE 2012; 7:e52901

13. Matsuta Y, Ito M, Tohsato Y. ECOH: An Enzyme Commission number predictor using mutual information and a support vector machine. Bioinformatics 2013; 29:365–372

14. Volpato V, Adelfio A, Pollastri G. Accurate prediction of protein enzymatic class by N-to-1 Neural Networks. BMC Bioinformatics 2013; 14:S11

15. Amidi A, Amidi S, Vlachakis D, et al. EnzyNet: enzyme classification using 3D convolutional neural networks on spatial representation. PeerJ 2018; 6:e4750

16. Che Y, Ju Y, Xuan P, et al. Identification of Multi-Functional Enzyme with Multi-Label Classifier. PLOS ONE 2016; 11:e0153503

17. Sureyya Rifaioglu A, Doğan T, Jesus Martin M, et al. DEEPred: Automated Protein Function Prediction with Multi-task Feed-forward Deep Neural Networks. Sci. Rep. 2019; 9:

18. You R, Zhang Z, Xiong Y, et al. Data and text mining GOLabeler: improving sequence-based large-scale protein function prediction by learning to rank. Bioinformatics 2018; 34:2465–2473

19. You R, Yao S, Xiong Y, et al. NetGO: improving large-scale protein function prediction with massive network information. Nucleic Acids Res. 2019; 47:379–387

20. Kulmanov M, Khan MA, Hoehndorf R. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. 2017; 34:660–668

21. Doğan T, MacDougall A, Saidi R, et al. UniProt-DAAC: domain architecture alignment and classification, a new method for automatic functional annotation in UniProtKB. Bioinformatics 2016; 32:2264–2271

22. Bileschi ML, Belanger D, Bryant DH, et al. Using deep learning to annotate the protein universe. Nat. Biotechnol. 2022; 40:932–937

23. Bonetta R, Valentino G. Machine learning techniques for protein function prediction. Proteins Struct. Funct. Bioinforma. 2020; 88:397–413

24. Jumper J, Evans R, Pritzel A, et al. Highly accurate protein structure prediction with AlphaFold. Nature 2021; 596:583–589

25. Huang K, Fu T, Gao W, et al. Therapeutics Data Commons: Machine Learning Datasets and Tasks for Drug Discovery and Development. 2021;

26. Wu Z, Ramsundar B, Feinberg EN, et al. MoleculeNet: A benchmark for molecular machine learning. Chem. Sci. 2018; 9:513–530

27. Haghighatlari M, Vishwakarma G, Altarawy D, et al. ChemML: A machine learning and informatics program package for the analysis, mining, and modeling of chemical and materials data. Wiley Interdiscip. Rev. Comput. Mol. Sci. 2020; 10:

28. Hu W, Fey M, Zitnik M, et al. Open Graph Benchmark: Datasets for Machine Learning on Graphs. Adv. Neural Inf. Process. Syst. 2020; 33:22118–22133

29. Korshunova M, Ginsburg B, Tropsha A, et al. OpenChem: A Deep Learning Toolkit for Computational Chemistry and Drug Design. J. Chem. Inf. Model. 2021; 61:7–13

30. Rao R, Bhattacharya N, Thomas N, et al. Evaluating Protein Transfer Learning with TAPE. 2019;

31. Huang K, Fu T, Glass LM, et al. DeepPurpose: a deep learning library for drug–target interaction prediction. Bioinformatics 2021; 36:5545–5547

32. Hanussek M, Bartusch F, Krüger J. BOOTABLE: Bioinformatics benchmark tool suite for applications and hardware. Future Gener. Comput. Syst. 2020; 102:1016–1026

33. Bankevich A, Nurk S, Antipov D, et al. SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. J. Comput. Biol. 2012; 19:455–477

34. Benson DA, Cavanaugh M, Clark K, et al. GenBank. Nucleic Acids Res. 2012; 41:D36–D42

35. The 1000 Genomes Project Consortium, Corresponding authors, Auton A, et al. A global reference for human genetic variation. Nature 2015; 526:68–74

36. Suzek BE, Wang Y, Huang H, et al. UniRef clusters: A comprehensive and scalable alternative for improving sequence similarity searches. Bioinformatics 2015; 31:926–932

37. Chen Z, Zhao P, Li F, et al. iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics 2018; 34:2499–2502

38. Wang J, Yang B, Revote J, et al. POSSUM: A bioinformatics toolkit for generating numerical sequence feature descriptors based on PSSM profiles. Bioinformatics 2017; 33:2756–2758

39. Elnaggar A, Heinzinger M, Dallago C, et al. Transactions on Pattern Analysis and Machine Intelligence ProtTrans: Toward Understanding the Language of Life

Through Self-Supervised Learning. IEEE TRANS PATTERN Anal. Mach. Intell. 2021; 14:

40. Bhasin M, Raghava GPS. Classification of Nuclear Receptors Based on Amino Acid Composition and Dipeptide Composition. J. Biol. Chem. 2004; 279:

41. Chou KC. Prediction of protein cellular attributes using pseudo-amino acid composition. Proteins Struct. Funct. Genet. 2001; 43:246–255

42. Zhou C, Wang C, Liu H, et al. Identification and analysis of adenine N6-methylation sites in the rice genome. Nat. Plants 2018; 4:

43. Cai CZ. SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence. Nucleic Acids Res. 2003; 31:

44. Chou K-C. Prediction of Protein Subcellular Locations by Incorporating Quasi-Sequence-Order Effect. Biochem. Biophys. Res. Commun. 2000; 278:

45. Shen J, Zhang J, Luo X, et al. Predicting protein-protein interactions based only on sequences information. Proc. Natl. Acad. Sci. 2007; 104:

46. Saini H, Raicar G, Lal S, et al. Protein Fold Recognition Using Genetic Algorithm Optimized Voting Scheme and Profile Bigram. J. Softw. 2016; 11:756–767

47. Sperandei S. Understanding logistic regression analysis. Biochem. Medica 2014; 24:

48. Cortes C, Vapnik V. Support-vector networks. Mach. Learn. 1995; 20:273–297

49. Murthy SK. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. Data Min. Knowl. Discov. 1998; 2:345–389

50. Breiman L. Random forests. Mach. Learn. 2001; 45:5–32

51. Zhang Z. Introduction to machine learning: k-nearest neighbors. Ann. Transl. Med. 2016; 4:

52. Friedman N, Geiger D, Goldszmidt M. Bayesian Network Classifiers. Mach. Learn. 1997; 29:

53. Friedman JH. Greedy function approximation: A gradient boosting machine. Ann. Stat. 2001; 29:1189–1232

54. Bebis G, Georgiopoulos M. Feed-forward neural networks. IEEE Potentials 1994; 13:27–31

55. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. 2016; 785–794

56. Ke G, Meng Q, Finley T, et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Adv. Neural Inf. Process. Syst. 2017; 30:

57. Sherstinsky A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Phys. Nonlinear Phenom. 2020; 404:132306

58. Indolia S, Goswami AK, Mishra SP, et al. Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. Procedia Comput. Sci. 2018; 132:679–688

59. Unsal S, Atas H, Albayrak M, et al. Learning functional properties of proteins with language models. Nat. Mach. Intell. 2022 43 2022; 4:227–245

60. Zhao Y, Wang J, Guo M, et al. A Literature Review of Gene Function Prediction by Modeling Gene Ontology. Front. Genet. Wwwfrontiersinorg 2020; 1:400

Appendix

Appendix A

Supplementary Tables

In this section, Table A1 – Table A5 are presented. Table A1 to Table A3 shows GO-terms and their corresponding names for each aspect. In Table A4, types of species and the number of proteins that belong to them are displayed. The ranges of values for each hyperparameters used in machine learning process were presented in Table A5

Table A1: List of selected Biological Process GO terms for the benchmarking experiments.

Aspects	GO ID	GO Name
Biological Process	GO:0006325	Chromatin Organization
	GO:0007281	Germ Cell Development
	GO:0008037	Cell Recognition
	GO:0010977	Negative Regulation of Neuro Projection Development
	GO:0015849	Organic Acid Transport
	GO:0018107	Peptidyl-Threonine Phosphorylation
	GO:0022618	Ribonucleoprotein Complex Assembly
	GO:0031334	positive regulation of protein-containing complex assembly
	GO:0033674	positive regulation of kinase activity
	GO:0043408	Regulation of MAPK cascade
	GO:0043409	negative regulation of MAPK cascade
	GO:0043902	obsolete positive regulation of multi-organism process
	GO:0045787	positive regulation of cell cycle
	GO:0045861	negative regulation of proteolysis
	GO:0050792	regulation of viral process

Table A2: List of selected Cellular Component GO terms for the benchmarking experiments.

Aspects	GO ID	GO Name
Cellular Component	GO:0005770	late endosome
	GO:0005789	endoplasmic reticulum membrane
	GO:0005794	Golgi apparatus
	GO:0005798	Golgi-associated vesicle
	GO:0005887	integral component of plasma membrane
	GO:0008021	synaptic vesicle
	GO:0015629	actin cytoskeleton
	GO:0016021	integral component of membrane
	GO:0019867	outer membrane
	GO:0019898	extrinsic component of membrane
	GO:0030424	axon
	GO:0030659	cytoplasmic vesicle membrane
	GO:0031226	intrinsic component of plasma membrane
	GO:0031227	intrinsic component of endoplasmic reticulum membrane
	GO:0031305	integral component of mitochondrial inner membrane

Table A3: List of selected Biological Process GO terms for the benchmarking experiments.

Aspects	GO ID	GO Name
Molecule Function	GO:0000981	DNA-binding transcription factor activity, RNA polymerase II-specific
	GO:0000987	cis-regulatory region sequence-specific DNA binding
	GO:0003712	transcription coregulator activity
	GO:0003723	RNA binding
	GO:0004175	endopeptidase activity
	GO:0004519	endonuclease activity
	GO:0008175	tRNA methyltransferase activity
	GO:0008270	zinc ion binding
	GO:0015108	chloride transmembrane transporter activity
	GO:0015291	secondary active transmembrane transporter activity
	GO:0016810	hydrolase activity, acting on carbon-nitrogen (but not peptide) bonds
	GO:0019207	kinase regulator activity
	GO:0019900	kinase binding
	GO:0046914	transition metal ion binding
	GO:0048037	obsolete cofactor binding

Table A4: Species names of the proteins in ProFAB dataset and their statistics.

Types of Species	Sets			
	EC Data	GO-Biological Function	GO-Cellular Component	GO-Molecular Function
Mus Musculus	4,183	13,778	14,207	12,738
Homo Sapiens	4,361	14,324	15,248	14,628
Rattus Norvegicus	2,269	7,025	6,915	6,444
Arabidopsis Thaliana	5,777	10,171	10,764	10,182
Saccharomyces Cerevisiae	2,001	5,114	5,411	4,750
Staphylococcus aureus	5,673	371	256	458
Schizosaccharomyces pombe	1,507	4,358	4,862	3,864
Escherichia coli	5,077	18	5	4
Others	237,323	53,568	48,145	56,572

Table A5: Hyperparameters and their value ranges for each algorithm.

Algorithm	Hyperparameters
MLP	hidden_layer_sizes = [100,300] activation = {'logistic', 'tanh', 'relu'}, solver = {'sgd','adam'}, batch_size = [100,150], learning_rate = {'constant', 'adaptive']} max_iter = 500 learning_rate_init = [10e-3,0.001]
GBoost	loss = {'deviance','exponential'} learning_rate = [0.1,5] criterion = {'mse','friedman_mse'} max_features = ['sqrt','log2']
SVM	kernel = {'linear','poly', 'rbf', 'sigmoid'} C = [0.5,50] gamma = [10e-3,1]
XGBoost	learning_rate = [10e-3.5, 5.0] tree_method = {'exact', 'gpu_hist'} max_leaves = [10,100] eval_metric = 'logloss'
LightGBM	boosting_type = {'gbdt','dart','goss'} learning_rate = [0.001,5.0] n_estimators = [100,300] min_split_gain = [10e-5,10e5] reg_alpha = [10e-4.7, 10e-0.5] reg_lambda = [10e-4.5, 10e-0.7] objective = 'binary'
CNN	epochs = [110,250], learning_rate = [7.5*10e-4,9*10**-4] batch_size = [30,100] out_size = {128,256} kernel_size_1 = [2,5] kernel_size_2 = [2,5] stride = 1 hid_size = 128 p = 0.3
RNN	epochs = [110,250], learning_rate = [7.5*10e-4,9*10**-4] batch_size = [100,500] embedding_size = {64,128,256} num_layers = [3,5] p = 0.3

Appendix B

Additional Results

In this chapter, additional benchmarking results in Table B1-B4 (MCC scores) and Figure B1 and Figure B2 (F1 scores) are presented in both machine learning and protein descriptor perspectives for binary classification method. In Figure B1 and Figure B2, each cell presents values of F1 of the model with machine learning algorithm (row) and protein descriptor (column).

Table B1: Average MCC scores of machine learning algorithms in the prediction of main enzyme classes based on different splitting techniques.

Main Enzyme Classes	Splitting Method	Learning Algorithms					
		GBoost	MLP	NB	SVM	XGBoost	LightGBM
Oxidoreductases	Random	0.625	0.548	0.285	0.232	0.725	0.652
	Similarity	0.520	0.519	0.278	0.220	0.520	0.457
	Temporal	0.496	0.326	0.280	0.225	0.600	0.572
Transferases	Random	0.560	0.502	0.236	0.179	0.682	0.590
	Similarity	0.388	0.381	0.191	0.118	0.438	0.350
	Temporal	0.391	0.267	0.204	0.104	0.542	0.461
Hydrolases	Random	0.444	0.416	0.151	0.100	0.615	0.570
	Similarity	0.321	0.354	0.132	0.096	0.365	0.334
	Temporal	0.369	0.210	0.115	0.069	0.534	0.450
Lyases	Random	0.688	0.621	0.322	0.242	0.777	0.691
	Similarity	0.524	0.553	0.276	0.244	0.549	0.460
	Temporal	0.369	0.210	0.115	0.069	0.534	0.450
Isomerases	Random	0.674	0.626	0.319	0.225	0.763	0.678
	Similarity	0.506	0.581	0.271	0.385	0.546	0.461
	Temporal	0.532	0.430	0.319	0.212	0.581	0.513
Ligases	Random	0.710	0.523	0.456	0.317	0.847	0.786
	Similarity	0.684	0.484	0.393	0.525	0.702	0.606
	Temporal	0.514	0.477	0.439	0.407	0.714	0.686
Translocases	Random	0.782	0.616	0.326	0.386	0.863	0.862
	Similarity	0.628	0.590	0.314	0.718	0.669	0.666

Table B2: Average MCC scores of protein descriptors in the prediction of main enzyme classes based on different splitting techniques.

Main Enzyme Classes	Splitting Method	Protein Descriptor			
		AAC	CTDT	PAAC	SOCNUMBER
Oxidoreductases	Random	0.535	0.515	0.571	0.423
	Similarity	0.448	0.438	0.472	0.319
	Temporal	0.441	0.405	0.474	0.345
Transferases	Random	0.482	0.446	0.500	0.404
	Similarity	0.349	0.313	0.330	0.252
	Temporal	0.340	0.314	0.377	0.282
Hydrolases	Random	0.419	0.379	0.421	0.311
	Similarity	0.308	0.273	0.287	0.200
	Temporal	0.292	0.272	0.346	0.254
Lyases	Random	0.594	0.567	0.572	0.494
	Similarity	0.489	0.417	0.487	0.343
	Temporal	0.470	0.423	0.485	0.366
Isomerases	Random	0.595	0.533	0.578	0.484
	Similarity	0.507	0.439	0.518	0.369
	Temporal	0.428	0.416	0.488	0.392
Ligases	Random	0.617	0.598	0.634	0.577
	Similarity	0.572	0.582	0.609	0.500
	Temporal	0.537	0.540	0.596	0.485
Translocases	Random	0.611	0.650	0.698	0.596
	Similarity	0.616	0.570	0.644	0.561

Table B3: Average MCC scores of machine learning algorithms in the prediction of GO term based on different splitting techniques.

GO Aspect	Splitting Method	Learning Algorithms					
		GBoost	MLP	NB	SVM	XGBoost	LightGBM
Biological Process	Random	0.563	0.441	0.352	0.561	0.676	0.660
	Similarity	0.355	0.266	0.252	0.346	0.443	0.395
	Temporal	0.314	0.216	0.186	0.213	0.378	0.241
Molecule Function	Random	0.479	0.407	0.369	0.489	0.586	0.604
	Similarity	0.439	0.338	0.341	0.485	0.526	0.542
	Temporal	0.448	0.386	0.286	0.372	0.490	0.409
Cellular Component	Random	0.415	0.276	0.226	0.350	0.592	0.544
	Similarity	0.367	0.267	0.267	0.326	0.446	0.422
	Temporal	0.362	0.297	0.249	0.256	0.432	0.362

Table B4: Average MCC scores of protein descriptors in the prediction of GO term based on different splitting techniques.

GO Aspect	Splitting Method	Protein Descriptor			
		AAC	CTDT	PAAC	SOCNUMBER
Biological Process	Random	0.586	0.559	0.600	0.424
	Similarity	0.401	0.331	0.356	0.284
	Temporal	0.315	0.244	0.282	0.191
Molecule Function	Random	0.549	0.500	0.525	0.382
	Similarity	0.496	0.446	0.481	0.357
	Temporal	0.469	0.409	0.408	0.308
Cellular Component	Random	0.428	0.380	0.439	0.354
	Similarity	0.356	0.342	0.388	0.311
	Temporal	0.346	0.326	0.386	0.247

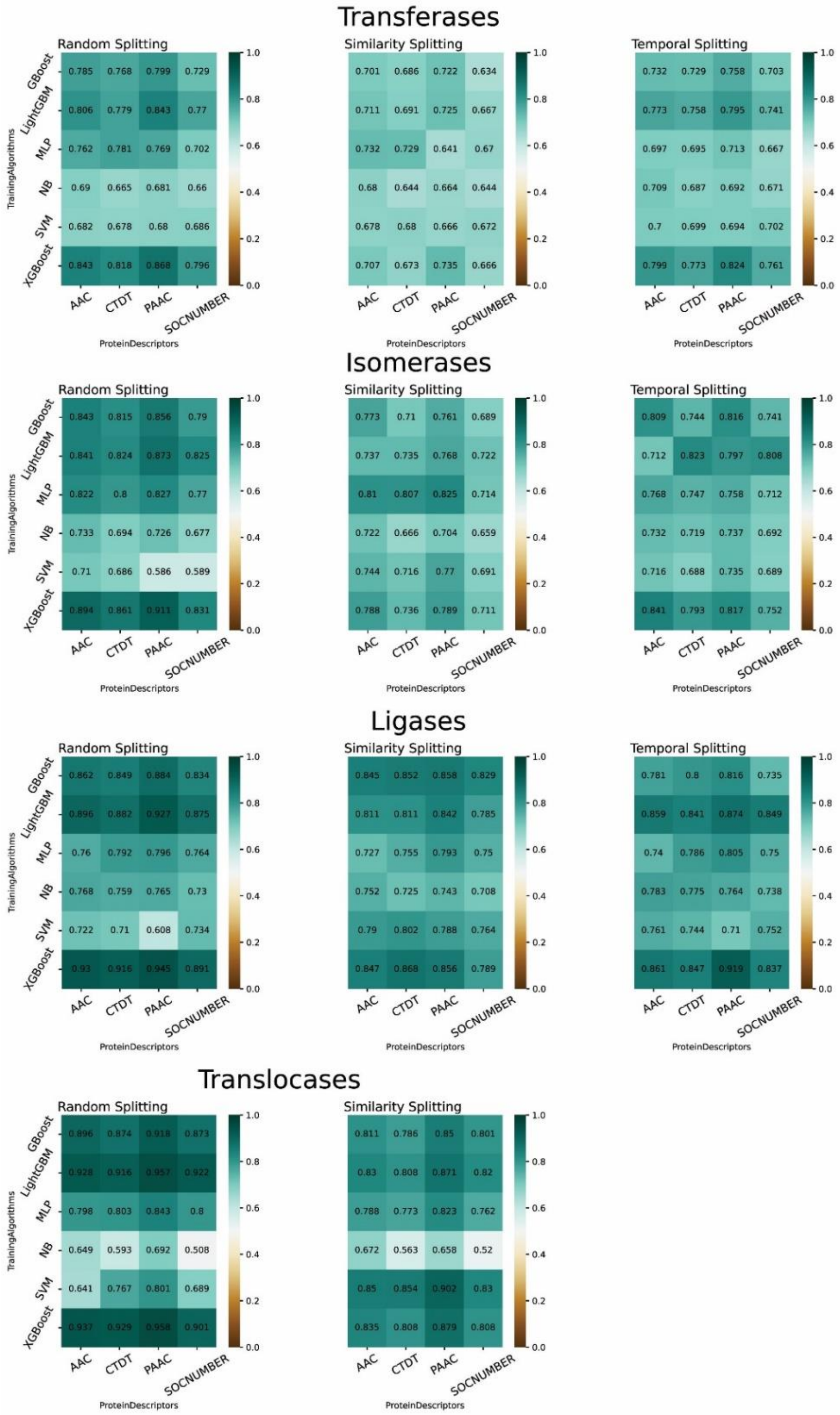

Figure B1: Performance results (F1-score) for EC main class prediction for transferase, isomerases, ligases and translocase datasets (there are no temporal splitting results for translocases class since previous years' annotations do not exist due to the fact that this class has recently been generated and annotated).

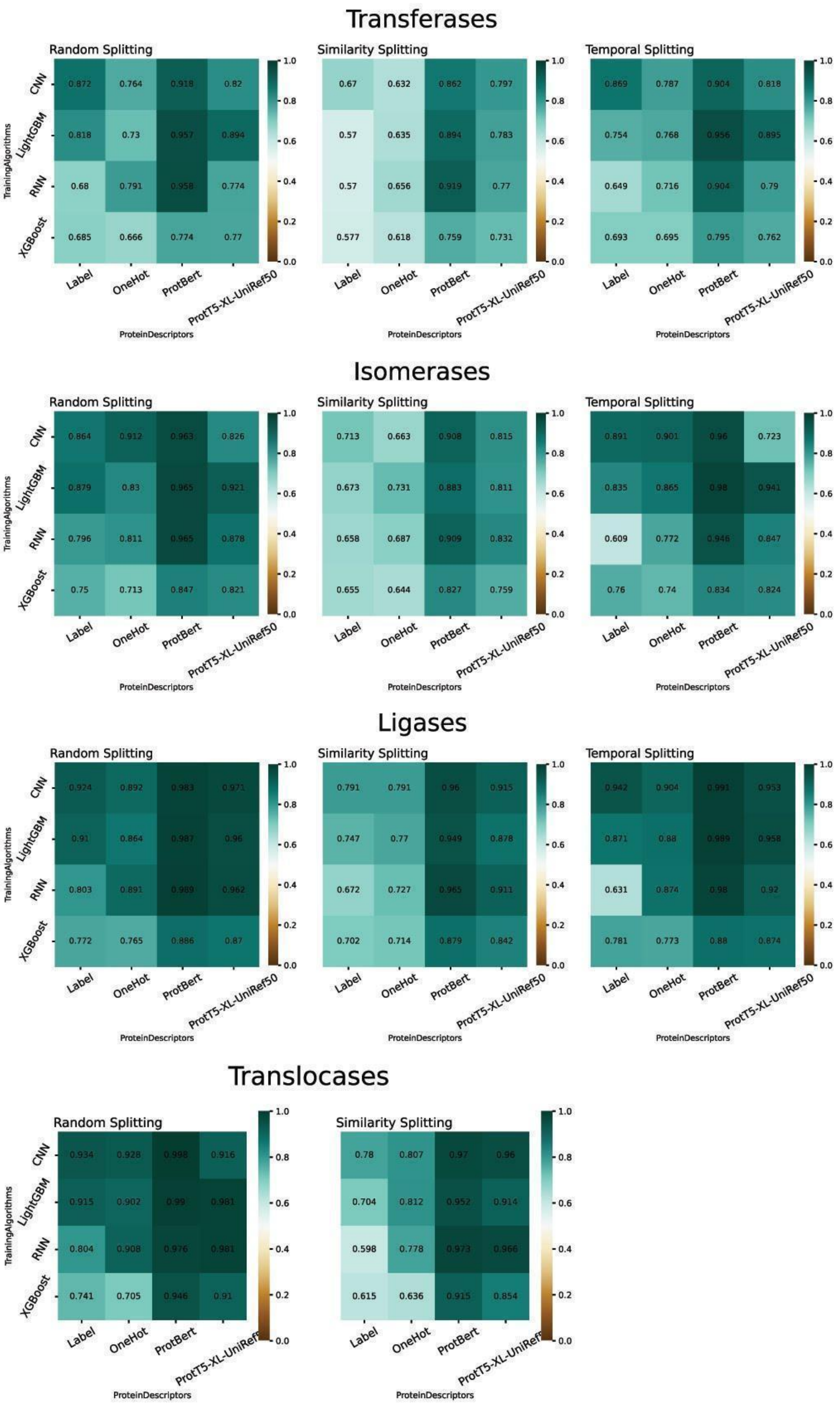

Figure B2: Performance results (F1-score) for EC main class prediction for transferase, isomerases, ligases and translocase datasets. The models are trained with transformed-based features, one-hot encoding and label encoding.