

# Machine Learning Models for Predictive Enzyme Classification

Malte Alexander Weyrich, Thanh Truc Bui, Jan Philipp Hummel and Sofiia Kozoriz

## 1 Abstract

The accurate prediction of enzyme commission (EC) numbers is not only crucial for the classification and understanding of newly discovered enzymes but also for completing the annotation of already known enzymes. Therefore, developing a reliable method for predicting EC numbers is very important.

However, due to insufficient data, enzyme function prediction using machine learning remains challenging. This paper proposes several methods for predicting enzymes in three different problem categories. Each category goes further into depth of the enzyme classification system, starting with a binary classification (level 0) of enzymes and non-enzymes, followed by a multi-class classification (level 1 & 2) of enzymes into main classes (e.g., EC X.-.-) and subclasses (e.g., EC X.X.-). Throughout developing our models, we used various input features and machine learning algorithms, of which the best will be thoroughly reviewed in this paper (see Table 1).

Level	Description	Model	F1	MCC
0	Binary classification	Random Forest	0.98	0.75
1	Main class prediction	Feedforward Neuronal Network	0.95	0.94
2	Subclass prediction	Feedforward Neuronal Network	0.85	0.83

**Table 1.** A table showing the best-performing models per level

## 2 Introduction

Enzymes are vital as biological catalysts, facilitating essential biochemical reactions in organisms. With enzyme catalysis, these reactions would occur slowly or practically impossible. Predominantly, proteins and enzymes are designated specific functions; for instance, oxidoreductases drive redox reactions, and isomerases convert molecules into their isomers. Therefore, a comprehensive understanding and a precise classification of enzymes are fundamental.

In the past, scientists attempted to categorize enzymes into groups and develop a logical rule set for naming. The efforts were not successful due to ambiguity. A significant milestone occurred in 1956 with the establishment of an official international commission on

enzyme classification (International Union of Biochemistry and Molecular Biology. Nomenclature Committee (1993)). This marked the initiation of the contemporary enzyme classification system, forming the basis for our understanding of enzymes today. In modern research, computational methods have become invaluable for enzyme classification; this shift has transformed the traditionally labor-intensive process. Machine learning and data-driven models, in particular, have assumed a leading role, holding the potential for enhanced accuracy and efficiency in annotating enzymes within vast genomic data. Several effective methods were developed to address this challenge. One notable example is DeepEC (Ryu *et al.* (2019)), a deep learning-based computational framework designed for the precise and high-throughput prediction of EC numbers for protein sequences. DeepEC employs three convolutional neural networks (CNNs) as a primary engine for EC number prediction and incorporates homology analysis for cases not classified by the CNNs.

Another noteworthy method is CLEAN (Yu *et al.* (2023)), which stands for “contrastive learning-enabled enzyme annotation”. CLEAN adopts a unique training objective to learn an embedding space for enzymes where the Euclidean distance reflects functional similarities. In CLEAN’s task, sequences sharing the same EC number exhibit a small Euclidean distance, while sequences with different EC numbers have a larger distance. The model was trained using contrastive losses with supervision for effective enzyme function prediction.

In our research, we developed several classification models for each level, using a variety of machine-learning algorithms and input features. The best-performing models per level will be thoroughly reviewed in this paper. The first is a binary classifier determining whether a given protein functions as an enzyme. The second and third models are multi-classifiers, categorizing enzymes into classes and subclasses based on their specific functionalities. Our primary objective is to outperform each category’s random baseline classifiers.

## 3 Methods

### 3.1 Binary Classification of Enzymes

In the course of assigning EC numbers to protein sequences, our initial step is to determine whether a given protein sequence is enzymatic or non-enzymatic (level 0). To identify the most effective approach for this task, we conducted training on the following binary classification models and compared their performance to select the optimal one:

- (1) k-Nearest-Neighbors

- (2) Random Forest
- (3) Support Vector Machine

The chosen method for binary classification will be detailed in the subsequent section, while section 6.1 will outline the methods that were not selected.

### 3.2 Multi-class Classification of Enzymes

We used a feedforward neural network (FNN) for levels 1 and 2 to classify the enzymes into their main classes and subclasses. The FNN is an artificial neural network (ANN) that uses a series of layers to extract features from the input data and classify it into different categories, thus making it a suitable choice for our multi-class classification task.

### 3.3 Training Data

The protein data sets were taken from the UniProt database and used in the CLEAN publication on Enzyme function prediction (see Yu *et al.* (2023)). The complete data pool of enzymes (called Split100 in Yu *et al.* (2023)) contains 224,693 sequences. These sequences are separated into several smaller subsets (SplitX,  $X \in \{10, 30, 50, 70\}$ ), where  $X$  represents the amount of similarity allowed between the sequences. For reference, Split30 only contains sequences that share at most 30% similarity. Since we want to reduce family overrepresentation, we used the Split30 dataset for our research, which contains 9,186 enzymes. This redundancy reduction ensures that the model is not remarkably biased towards a specific family of enzymes. This subset was combined with a non-enzyme dataset holding 37,347 sequences, resulting in 46,533 sequences.

### 3.4 Data Preprocessing

We removed all sequences containing an ambiguous amino acid, such as 'U' or 'O', from the dataset, as well as sequences with a length longer than 1,022 amino acids. This has to do with the fact that our enzyme dataset only contains sequences with a maximum length of 1,022 amino acids. The non-enzyme dataset, however, contained many sequences with a length of up to 10,000 amino acids. By removing these sequences, we ensured that the model was not biased towards the non-enzyme dataset.

Additionally we made use of the esm2 embeddings (see Lin *et al.* (2023)) and ProtT5 embeddings (see Elnaggar *et al.* (2021)). Large language models for protein sequences have learned the patterns of amino acid sequences by being trained on many protein sequences. This allows us to infer embeddings for our amino acid sequences, which are numerical vectors that represent the amino acid sequences. For each sequence we inferred the esm2 embeddings (see Lin *et al.* (2023)) and ProtT5 embeddings (see Elnaggar *et al.* (2021)) using the esm2 model (see Lin *et al.* (2023)) and the ProtT5 model (see Elnaggar *et al.* (2021)) respectively.

#### 3.4.1 Data Preprocessing for Level 0

As a random forest model relies on multiple decision trees and each decision tree requires various features, we opted to extract additional information from the protein sequence and the esm2 embeddings Lin *et al.* (2023).

From the protein sequences, we computed the mass by summing the individual masses of its amino acid components. Meanwhile, the esm2 embeddings, each represented by a 2,560-dimensional vector, underwent statistical analysis. We calculated the median, standard deviation, and vector magnitude by aggregating values across all 2,560 dimensions for each protein. For clarity, the mass of protein sequences is presented in atomic mass unit (AMU), and the term 'vector magnitude' refers to the square root of the sum of the squares of all vector dimensions. We computed

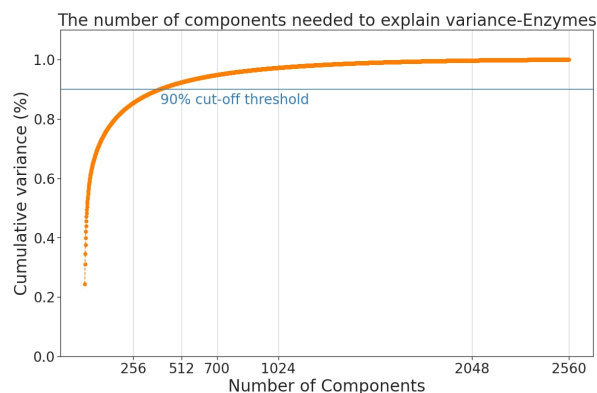


Fig. 1. The number of components needed to explain the variance in the enzyme dataset

the median, standard deviation, and vector magnitude by aggregating values across all 2,560 dimensions for each protein.

To simplify the embeddings, we applied Principal Component Analysis (Pedregosa *et al.* (2011)) separately to enzyme and non-enzyme datasets, retaining 90% of the variance. This process yielded reduced dimensions of 397 (see Figure 1) for enzymes and 369 for non-enzymes. By standardizing both datasets to a dimensionality of 397, we achieved a streamlined representation of protein embeddings while preserving essential information. This constant dimensionality of 397 was also employed to reduce the protein embeddings of unseen datasets during evaluation.

By combining the information from the proteins' sequences, masses, and embeddings, we created a data frame that concatenates enzymes and non-enzymes, including 401 features: mass, embeddings median, embeddings standard deviation (std), embeddings magnitude, and dimension 1 to 397 of the reduced embeddings.

We also used the method `SelectFromModel` from the scikit-learn library (see Pedregosa *et al.* (2011)) to select features based on importance weights. The refined set of input features contains 'Mass,' 'Emb median,' 'Emb std,' 'Emb magnitude', and 'PCA 1' through 'PCA 47' (see table 2) excluding 'PCA 29', 'PCA 31', 'PCA 35', 'PCA 36', 'PCA 39' to 'PCA 42', 'PCA 44' and 'PCA 46' (where 'Emb' represents embeddings and 'PCA X' signifies the X-th dimension of the reduced embeddings).

Mass	Emb median	Emb std.	Emb magnitude	PCA 1	...	PCA 47
60153,711	-0,002189	0,227337	11,502516	0,450330	...	-0,050497
81547,542	-0,002620	0,240143	12,150464	-0,087801	...	0,220343
...	...	...	...	...	...	...

Table 2. Input features for the Random Forest model

#### 3.4.2 Data Preprocessing for Level 1

For the level 1 prediction, we labeled each sequence with its respective main class, resulting in 7 classes (see Figure 2). While the second main class is the most prevalent, the seventh main class is the least prevalent, with only containing 190 representatives.

#### 3.4.3 Data Preprocessing for Level 2

For the level 2 prediction, we labeled each sequence with its respective subclass (see section 7 Figure 7). The problem of imbalanced data and underrepresentation of some labels is even more pronounced when labeling

the sequences with their respective subclass. Some enzyme families had less than ten representatives in the dataset, making it difficult for the model to learn the patterns of these families. To address this issue, we combined underrepresented families of the same main class into a single class (e.g. **EC 2.2.-.-**, **EC 2.9.-.-** and **EC 2.19.-.-** were merged into the label **EC 2.2|9|19.-.-**). This way, we guaranteed that each subclass had at least ten representatives in the dataset. This procedure had to be applied to each main class separately, as the subclasses are unique within their respective main class.

### 3.5 Training Procedure

We divided the data table into two parts for all our models: a training set and a validation set, using a random state of 42 for consistency. The training set contains 70% of the original data, while the validation set holds the remaining 30%. This separation allows us to train our models on a subset of the data and assess its performance on a different subset to ensure its generalization to new, unseen data.

#### 3.5.1 Level 0 - Random Forest

A straightforward yet powerful machine learning method is the Random Forest algorithm. The choice of Random Forest is driven by its effectiveness in handling classification tasks, making it a well-suited option for our specific protein classification objective.

For the Random Forest classifier, we addressed the imbalance between the number of non-enzymes and enzymes, where non-enzymes outnumber enzymes approximately fourfold by duplicating the enzyme data in the training set four times. This duplication ensures a more balanced dataset, allowing the model to be trained on an equal representation of both classes.

We built a Random Forest Classifier using the scikit-learn library (see Pedregosa *et al.* (2011)) with specific parameters: The classifier consists of a total of 200 decision trees; each tree has a maximum depth of 16 and a node is designated as a leaf only if it has a minimum of 8 samples. The random state parameter is set to 42, ensuring reproducibility in the model's construction.

The classifier consists of a total of 200 decision trees. Each tree has a maximum depth of 16, and a node is designated as a leaf only if it has a minimum of 8 samples. The random state parameter is set to 42, ensuring reproducibility in the model's construction.

#### 3.5.2 Level 1 - FNN

The feedforward neural network for level 1, using ESM2 embeddings, involves a sequential arrangement of linear layers, each accompanied by LeakyReLU activation functions. This choice of activation function, LeakyReLU, is particularly remarkable for its capability of dealing with the “dying ReLU” problem by allowing a small gradient when the unit is inactive. The architecture of the model consists of layers sized at 2,560, the input layer, hidden layers being 2,048, 768, 256, and finally, 7 units as an output layer. It is worth pointing out that the architecture contains batch

normalization layers (BatchNorm1d) intended for normalizing the input layer through adjustment and scaling activations to stabilize and accelerate the training process.

#### 3.5.3 Level 2 - FNN

For the level 2 prediction, we used a feedforward neural network (FNN) to classify the enzymes into their respective subclasses. We implemented this architecture using TensorFlow, an open-source library for machine learning (see Abadi *et al.* (2015)). The architecture comprises one input layer, two hidden layers, and one output layer. In between the hidden layers, we used dropout layers to prevent overfitting. The layers are fully connected, and the activation function used is the rectified linear unit (ReLU). The number of perceptrons used per layer and the dropout rate were optimized using the Optuna library (see Akiba *et al.* (2019)). This resulted in 262 perceptrons in the first hidden layer and 121 in the second. The dropout rate was set to 0.23. While the input layer had the same dimension as the esm2b embeddings (2,560), the output layer had a dimension of 51, since there are 51 subclasses in the training dataset (see Figure 7). To further address the imbalance in the dataset, we used class weights to penalize the model more for misclassifying the minority classes while being more lenient towards misclassifying the majority classes. During training, we used early stopping with a patience of 15 epochs to prevent overfitting.

### 3.6 Evaluation on Test Dataset

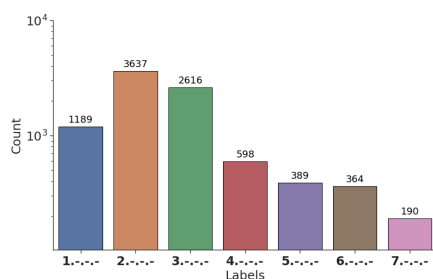
The enzymatic dataset we used for evaluation is called “New-392” and was used as one of the benchmark datasets in the CLEAN publication (see Yu *et al.* (2023)). For the level 0 prediction, we concatenated the “New-392” dataset with the non-enzymes test dataset. We also filtered for ambiguous amino acids and sequences longer than 1,022 amino acids. The refined non-enzyme dataset contains 9,336 non-enzymes, resulting in 9,728 sequences for the level 0 prediction. For the level 1 and level 2 predictions, we removed multifunctional enzymes that differ in their main class.

#### 3.6.1 Scoring Metrics

To assess the performance of our models, we emphasize the F1 and MCC scores. Given the imbalance in the test set, we also considered the weighted F1 score for the multiclassification models to ensure a more equitable evaluation of the model's effectiveness.

The F1 score is the harmonic mean of precision and recall, with values ranging from 0 to 1. A score of 1 signifies a flawless prediction, while 0 indicates an utterly inaccurate prediction.

The MCC score is a correlation coefficient that indicates the accuracy of binary classifications, with values ranging from -1 to 1. A score of 1 signifies a flawless prediction, -1 indicates a completely inaccurate prediction, and 0 suggests predictions at random. This scoring metric takes into account the entirety of the confusion matrix ( $TP$ ,  $TN$ ,  $FP$ ,  $FN$ ),



**Fig. 2.** Main class distribution of Split30

making it a valid metric for measuring the performance across all classes. It can also be applied to multiclassification models by calculating the MCC score for each class and then averaging the scores.

$$\text{precision} = \frac{TP}{TP + FP} \quad (1)$$

$$\text{recall} = \frac{TP}{TP + FN} \quad (2)$$

$$\text{f1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (3)$$

$$\text{MCC score} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \quad (4)$$

### 3.6.2 Bootstrapping Procedure

Bootstrapping describes randomly drawing  $n$  samples with replacement from the same data pool  $B$  times. We used this method to further evaluate our model scores, as it allows us to estimate the mean, standard error, and confidence interval of the scores. We chose  $B$  as 10,000 and  $n$  as the predicted samples. This means that we drew 10,000 pairs of actual labels and predicted labels from the test set with replacement, allowing us to calculate the scores for each bootstrapped dataset and then estimate the performance's mean, standard error, and confidence interval. Each score has been rounded to significant digits using the standard error. We calculated the error bars using an  $\alpha$  of 0.05 and the standard error of the scores. The performance plots show the mean score and the confidence intervals as error bars.

## 4 Results

### 4.1 Binary Classification

Using the Random Forest model on the “new” dataset aggregated with the non-enzymes test set, we achieved accurate predictions for about 90% of positive cases (enzymes) and 98% of negative cases (non-enzymes), even with an imbalanced test set comprising 392 enzymes and 9,876 non-enzymes. As a reference, we provided the confusion matrix in Figure 3. Additionally, we attained an Accuracy of 97.6%, a weighted f1-score of 97.8%, and an MCC-score of 74.1%. Figure 6 visually demonstrates the significant outperformance of our Random Forest model compared to the random baseline, which predicted labels based on the training set's proportions.

Actual	Non-Enzyme	9669 97.90%	207 2.10%
	Enzyme	40 10.20%	352 89.80%
		Non-Enzyme	Enzyme
		Predicted	

Fig. 3. Confusion Matrix of the Random Forest Model on the “new” dataset

### 4.2 Main Class Prediction

After testing the FNN, we observed that the model achieved an F1 score of 0.95 and an MCC score of 0.94. The confusion matrix presented in

Actual	1-.-.-	83 (96.51%)	0 (0.00%)	0 (0.00%)	1 (1.16%)	2 (2.33%)	0 (0.00%)	0 (0.00%)
	2-.-.-	1 (0.91%)	102 (92.73%)	5 (4.55%)	2 (1.82%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
	3-.-.-	0 (0.00%)	4 (4.71%)	80 (94.12%)	1 (1.18%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
	4-.-.-	0 (0.00%)	2 (2.25%)	0 (0.00%)	87 (97.75%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
	5-.-.-	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	8 (100.00%)	0 (0.00%)	0 (0.00%)
	6-.-.-	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	6 (100.00%)	0 (0.00%)
	7-.-.-	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	4 (100.00%)
	1-.-.-	2-.-.-	3-.-.-	4-.-.-	5-.-.-	6-.-.-	7-.-.-	
	Predicted							

Fig. 4. Normalized confusion matrix of level 1 prediction.

Figure 4 demonstrates the model's robust predictive capability across seven classes, highlighting its high accuracy level. Most predictions align with the diagonal, and the percentages of accurate predictions stand out, ranging remarkably from 92.73% to 100%.

### 4.3 Subclass Prediction

In order to test our FNN model for the subclass prediction, we labeled the sequences of the new dataset with their respective subclasses. This yielded a subset of the labels in the training dataset, containing 28 subclasses out of the 51 in the training dataset. Our model achieved a weighted F1 score of 0.85 and an MCC score of 0.83 on the new dataset. We also calculated the confusion matrix of the model on the new dataset (see Figure 5). The missing rows in the confusion matrix represent the 23 remaining subclasses that were not present in the new dataset. The reasonably low noise per row indicates that the model can distinguish between the present subclasses of the new dataset.

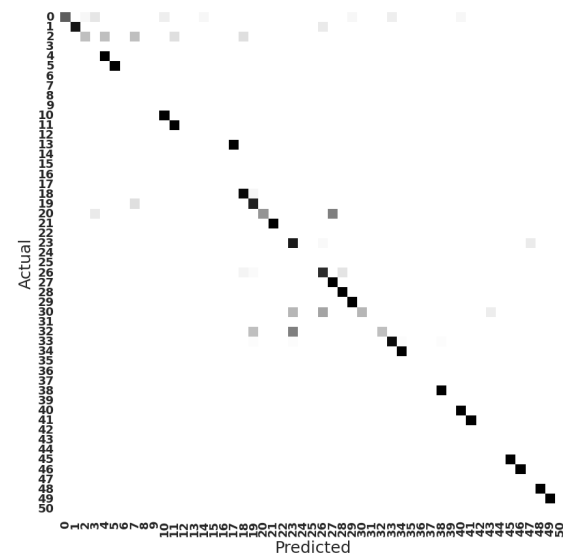


Fig. 5. Normalized confusion matrix of level 2 prediction.

## 5 Discussion

The two main approaches we employed for the problem, namely Random Forest and Feedforward Neural Network, delivered promising results, surpassing the random baseline models. However, there is room for improvement due to the dataset's imbalance. One avenue for enhancement involves exploring additional features, such as incorporating information from protein 3D structures and protein interactions. These aspects could provide valuable insights and contribute to further refining the accuracy of our models.

An idea for further enhancement is to create a comprehensive pipeline by combining all three models to predict proteins from level 0 to level 2. In this hypothetical pipeline, the Random Forest model would predict level 0. If the protein is identified as an enzyme, the Feedforward Neural Network (FNN) at level 1 could predict the main class. Following this, the enzyme could undergo prediction by the FNN at level 2. Since the FNN at level 2 implicitly predicts the main class, we could align the predictions of the two models. If the main class predicted by the FNN at level 2 matches the main class predicted by the FNN at level 1, we would consider the entire result from the FNN at level 2. Otherwise, we would discard the prediction of the FNN at level 2.

In conclusion, the three models represent promising tools for supporting researchers in classifying proteins. While each model demonstrates efficacy individually, the envisioned pipeline integrating all three could enhance protein classification's overall accuracy and reliability. Further exploration and implementation of these models may contribute valuable insights to the field of protein classification.

## 6 Supplementary Information

### 6.1 Additional Binary Classification Models

As we thoroughly reviewed the Random Forest model in section 4.1, we will outline the other binary classification models we considered for this task.

#### 6.1.1 k-Nearest-Neighbors (kNN)

kNN is a non-parametric classification method that assigns a new object to the most common class amongst the most similar  $k$  objects in the data set (Hand (2007)). We implemented three kNN models in Python using scikit-learns Nearest Neighbors library, and these models used three distinct types of input features:

- (1) Protein embedding vectors encoded by ProtT5 (see Elnaggar *et al.* (2021))
- (2) Protein embedding vectors encoded by ESM2 (see Lin *et al.* (2023))

(1) *kNN using ProtT5 encoded embedding vectors*: Each protein sequence, represented as a numerical vector of 1.024 dimensions, is placed as a data point within a 1.024-dimensional space. The kNN classifier, using  $k=7$ , categorizes each data point to the class that is most common among its  $k$  nearest neighbors in the 1.024-dimensional space. Figure 6 illustrates the performance comparison between the kNN algorithm using ProtT5 encoded embedding vectors versus a random baseline. We observe that the mean F1 score for this model demonstrates a slight improvement over the random baseline, recording 0.857 compared to 0.843. This implies that this approach showcases superior precision and recall compared to the random baseline. This model's Mathews Correlation Coefficient of 0.728 indicates that this approach performs significantly better than random guessing.

(2) *kNN using ESM2 encoded embedding vectors*: Each protein sequence, represented as a numerical vector of 2.560 dimensions, is placed as a data point within a 2.560-dimensional space. The kNN classifier, using  $k=7$ ,

categorizes each data point to the class that is most common among its  $k$  nearest neighbors in the 2.560-dimensional space. Figure 6 illustrates the performance comparison between the kNN algorithm using ESM2 encoded embedding vectors versus a random baseline. Unlike above, we observe that the mean F1 score for this model demonstrates a slight regression over the random baseline, recording 0.833 compared to 0.843. This implies that this approach showcases inferior precision and recall in contrast to the random baseline. Despite being lower than the previous case, this models Mathews Correlation Coefficient of 0.689 still indicates that this approach preforms better than random guessing.

#### 6.1.2 Support Vector Machine (SVM)

SVM is an algorithm that assigns labels to objects by learning from examples (Boser *et al.* (1992)). We implemented three SVM models in Python using scikit-learns C-Support Vector classification library and these models using three distinct types of input features:

- (1) Protein embedding vectors encoded by ProtT5 (Elnaggar *et al.* (2021))
- (2) Protein embedding vectors encoded by ESM2 (Lin *et al.* (2023))

(1) *SVM using ProtT5 encoded embedding vectors*:

(2) *SVM using ESM2 encoded embedding vectors*:

## 7 Supplementary Figures

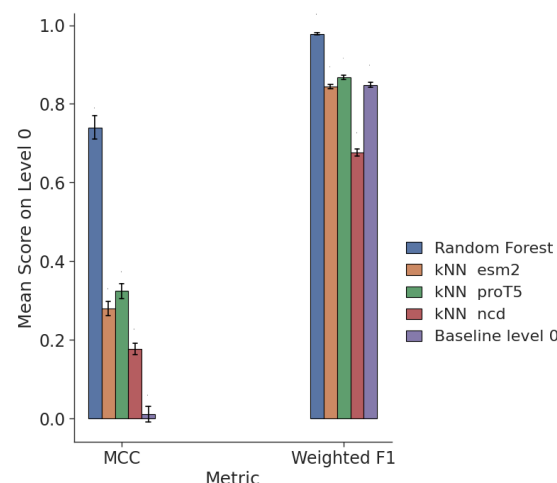


Fig. 6. Comparison of all level 0 models

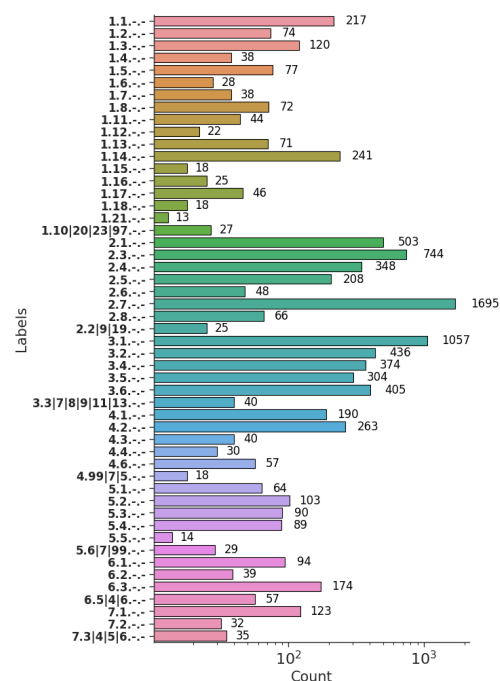


Fig. 7. Subclass distribution in Split30 (logarithmic scale)

## References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
- Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.
- Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., et al. (2021). Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, **44**(10), 7112–7127.
- Hand, D. J. (2007). Principles of data mining. *Drug safety*, **30**, 621–622.
- International Union of Biochemistry and Molecular Biology. Nomenclature Committee (1993). *Enzyme nomenclature 1992*. Academic Press, San Diego, CA.
- Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., and Rives, A. (2023). Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, **379**(6637), 1123–1130.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.
- Ryu, J. Y., Kim, H. U., and Lee, S. Y. (2019). Deep learning enables high-quality and high-throughput prediction of enzyme commission numbers. *Proceedings of the National Academy of Sciences*, **116**(28), 13996–14001.
- Yu, T., Cui, H., Li, J. C., Luo, Y., Jiang, G., and Zhao, H. (2023). Enzyme function prediction using contrastive learning. *Science*, **379**(6639), 1358–1363.