Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Subject Section

This is a title

Corresponding Author [1,*], Co-Author [2] and Co-Author [2,*]

[1]Department, Institution, City, Post Code, Country and
[2]Department, Institution, City, Post Code, Country.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation:
Results:
Availability:
Contact: name@bio.com
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Abstract

The accurate prediction of enzyme commission numbers (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 of great importance.

However, due to insufficient data, enzyme function prediction using machine learning is an ongoing challenge. In this paper, we propose several methods for predicting enzymes in three different problem categories (Table 1). Throughout the developing of our models, we used a variety of different input features and machine learning algorithms, of which the best will be thoroughly reviewed in this paper.

Table 1. Description of subproblem categories

Level	Description	Best performing method	F1 score
0	Binary classification	Random Forest	score
1	Main class classification	Feedforward neuronal network	score
2	Subclass classification	Feedforward neuronal network	score

2 Introduction

Enzymes serve as crucial biological catalysts, playing an essential role in numerous biochemical reactions within organisms. Without enzyme catalysis, those reactions would occur too slow or they would even be impossible. Most enzymes are proteins, and each enzyme is assigned a specific function. For example, Oxidoreductases catalyze redox reactions, while Isomerases convert a molecule into its isomer. Consequently, there is a fundamental need for a comprehensive understanding and precise classification of enzymes.

In the past, scientists attempted to categorize enzymes into groups and develop a logical rule set for naming. However, the efforts were hindered by ambiguity. A significant milestone occurred in 1956 with the establishment of an official international commission on enzyme classification (IUBMB). This marked the initiation of the contemporary enzyme classification system that forms the basis for our understanding of enzymes today.

In recent years, computational approaches have emerged as powerful tools for enzyme classification, offering efficient means to navigate through the vast landscape of protein sequences and structures. Enzyme classification, traditionally a labor-intensive task, has witnessed a transformative shift with the advent of computational techniques. This shift is driven by the exponential growth of available biological data, propelled by advancements in high-throughput technologies. Computational methods, particularly machine learning and data-driven models, now stand at the forefront of enzyme classification endeavors, promising to enhance accuracy and efficiency in the annotation of enzymes within large-scale genomic datasets.

3 Approach

4 Methods

The protein data was taken from the UniProt database and used in the CLEAN publication on Enzyme function prediction[1]. The mass table of common amino acids was taken from Bioinformatics Solutions Inc[2].

© The Author 2015. Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com

4.1 Machine learning algorithms used per level

Our initial goal is to categorize proteins as either enzymes or non-enzymes (level 0). To accomplish this, we've opted for the Random Forest, a straightforward yet powerful machine learning method. 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. We built a Random Forest Classifier using the scikit-learn library 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.

For ...

For ...

4.2 Data preprocessing in general

Unwanted Sequences Removal We excluded sequences that contain the amino acids "O" and "U" from our dataset.

4.3 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 both the protein sequence and the protein embeddings.

From the amino acid mass table we computed the mass of protein sequences by adding up the individual masses of their amino acid components. The protein embeddings, each represented by a 2560-dimensional vector, underwent statistical analysis. We computed the median, standard deviation, and vector magnitude by aggregating values across all 2560 dimensions for each protein. To simplify the embeddings, we applied Principal Component Analysis (PCA)[3] separately to enzyme and non-enzyme datasets, retaining 90% of the variance. This process yielded reduced dimensions of 397 for enzymes and 369 for non-enzymes. Therefore, we reduced the dimensions to 397, providing a streamlined representation of the protein embeddings while retaining crucial information for both enzyme (see figure 1) and non-enzyme (see figure 2) datasets.

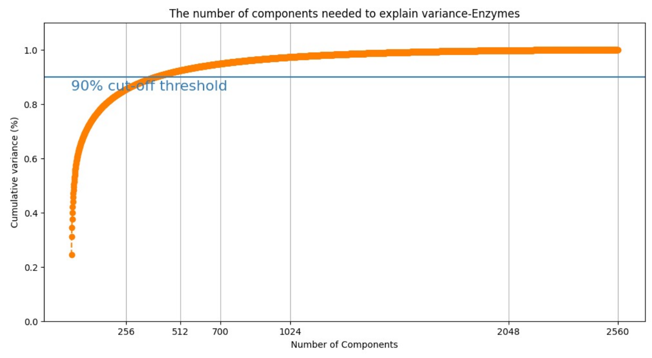

Fig. 1. The number of components needed to explain the variance in the enzyme dataset

By combining the information from the proteins' sequences, masses, and embeddings, we created a Pandas DataFrame 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[4] from the scikit-learn library to select features based on importance weights. The refined set of input features contains 'Mass,' 'Emb median,' 'Emb std,' 'Emb

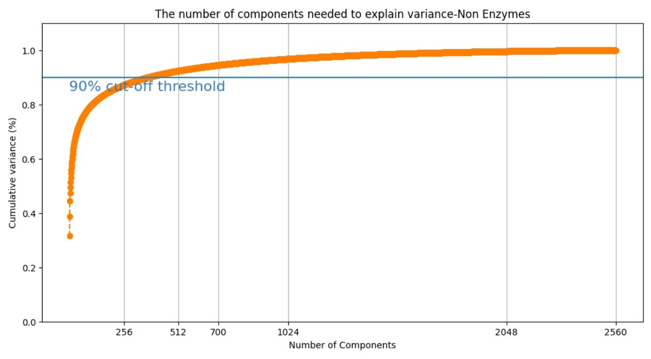

Fig. 2. The number of components needed to explain the variance in the non enzyme dataset

magnitude,' and 'PCA 1' through 'PCA 47' 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).

4.4 Training procedure

For all our models we divided the data table into two parts: 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 then assess its performance on a different subset to ensure its generalization to new, unseen data.

4.4.1 Random Forest

To address the imbalance between the number of non-enzymes and enzymes, where non-enzymes outnumber enzymes approximately fourfold, we duplicated 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.

4.4.2 Level 1 FNN

4.4.3 Level 2 FNN

4.5 Validation on test dataset

4.5.1 Scoring metrics

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

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

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

The *MCC* score serves as an indicator for 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.

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

5 Results

5.1 Random Forest Level 0

Using the Random Forest model on the "new" 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 9876 non-enzymes.

5.1.1 Support Vector Machine

5.2 Level 1 performance

5.3 Level 2 performance

6 Discussion

7 Conclusion

8 Supplementary Information

8.1 K-nearest neighbors algorithm using ncd vectors

A less popular approach of transforming string like input features into numerical values is the normalized compression distance (ncd) algorithm. The ncd algorithm is based on the idea that the similarity of two strings can be measured by the amount of information needed to describe one string given the other string. The ncd of two strings x and y is defined as follows:

$$\text{ncd}(x, y) = \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))} \tag{5}$$

where $C(x)$ is the length of the compressed string x and $C(xy)$ is the length of the concatenated string xy.

We implemented this algorithm in python using *gzip*, which is a loss less compression algorithm based on a combination of LZ77 and Huffman encoding. Rigler *et al.* (2007)

The ncd algorithm was used to transform the amino acid sequences into numerical vectors by comparing each sequence to all other sequences in the training dataset. This resulted in a n-dimensional numerical vector for each sequence, where n is the amount of sequences in the training dataset where each position in the vector represents the ncd of the sequence to the corresponding sequence in the training dataset. These vectors were then used as input for the k-nearest neighbors algorithm. Due to the exponential computational complexity of the ncd algorithm, we had to under sample the non enzyme dataset to match the amount of samples in our enzyme dataset, meaning the positiv in the train dataset were balanced.

When inferring unseen data, the ncd input vector was calculated by comparing it to all sequences in the training data set, thus also resulting in a n-dimensional numerical vector. This means that the performance on new data is largely dependent on the training data set.

The performance of the k-nearest neighbors algorithm using ncd vectors compared to a random baseline is shown in figure 3. Although the mean F1 score of the k-nearest neighbors algorithm using ncd vectors lies at 0.728, it did not perform better than the random baseline, which had a F1 score of 0.843. This indicates that the ncd approach has a worse precision and recall than the random baseline. At the same time both classifiers have a low MCC score of 0.2 and 0.01 respectively, which indicates that both classifiers are not better than random guessing.

The reason for the poor performance of the k-nearest neighbors algorithm using ncd vectors is most likely due to the ncd algorithm not being suited for protein sequences as shown in Matsumoto *et al.* (2000) as well as the test dataset not being balanced, while the training dataset was.

References

Matsumoto, T., Sadakane, K., and Imai, H. (2000). Biological sequence compression algorithms. *Genome Informatics*, **11**, 50–51.

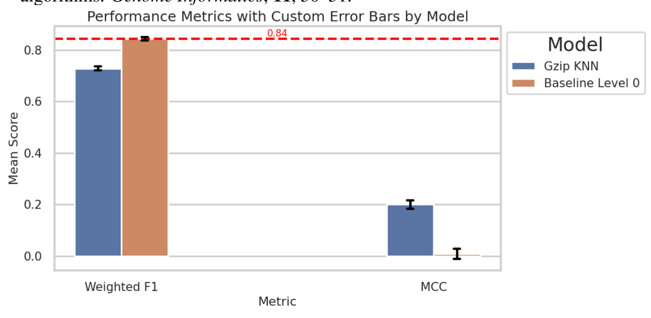

Fig. 3. Performance on test dataset compared to random baseline

Rigler, S., Bishop, W., and Kennings, A. (2007). Fpga-based lossless data compression using huffman and lz77 algorithms. In *2007 Canadian Conference on Electrical and Computer Engineering*, pages 1235–1238.