*Bioinformatics*, YYYY, 0–0

doi: 10.1093/bioinformatics/xxxxx

Advance Access Publication Date: DD Month YYYY

Manuscript Category

|  |
| --- |
| Subject Section  Manuscript Title  Corresponding Author1,\*, Co-author2 and Co-Author2  1Department of XXXXXXX, Address XXXX etc., 2Department of XXXXXXX, Address XXXX etc.  \*To whom correspondence should be addressed.  Associate Editor: XXXXXXX  Received on XXXXX; revised on XXXXX; accepted on XXXXX  Abstract  **Motivation:** The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.  **Results:** The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.  **Availability:** The quick brown fox jumps over the lazy dog.  **Contact:** example@example.org  **Supplementary information:** Supplementary data are available at *Bioinformatics* online. |

# Introduction

# Methods

# Methods

//

## Binary Classification

In the course of assigning EC numbers to protein sequences, our initial step is to determine whether a given protein sequence is enzymatic or a non-enzymatic.

To identify the most effective approach for this task, we conducted training the following binary classification models and compared their performance to select the optimal one:

k-Nearest-Neighbors

1. Random Forest
2. Support Vector Machine

The chosen method will be detailed in the subsequent section, while Section 7.1 will outline the methods that were not selected.

## Binary Classification

As we selected the Random Forest model to determine whether a given protein sequence is enzymatic or a non-enzymatic, below we describe each approach and outline its performance metrics.

### 7.1.1 Binary Classification: k-Nearest Neighbors (kNN)

kNN is a non-parametric classification method, which assigns a new object to the most common class amongst the most similar k objects in the data set. [1]. We implemented three kNN models in Python using scikit-learns Nearest Neighbors library and these models using three distinct types of input features:

1. Protein embedding vectors encoded by ProtT5 [2]
2. Protein embedding vectors encoded by ESM2 [3]
3. Normalized compression distance vectors

### 7.1.1.1 kNN using ProtT5 encoded embedding vectors

Each protein sequence, represented as a numerical vector of 1024 dimensions, is placed as a data point within a 1024-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 1024-dimensional space.

Figure \ref{fig:03} 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 in contrast to the random baseline. This models Mathews Correlation Coefficient of 0.728 indicates that this approach preforms significantly better than random guessing.

### 7.1.1.2 kNN using ESM2 encoded embedding vectors

Each protein sequence, represented as a numerical vector of 2560 dimensions, is placed as a data point within a 2560-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 2560-dimensional space.

Figure \ref{fig:03} 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.

### 7.1.1.3 kNN using normalized compression distance vectors

The normalized compression distance (ncd) algorithm transforms string like input features into numerical values and is based on the concept of measuring the similarity of two strings by the amount of information needed to describe the one string given the other string. Given two strings x and y, thein cd is defined as follows:

//ncd formula//

We implemented the ncd algorithm in Python utilizing gzip, a lossless compression algorithm based on a combination of LZ77 and Huffman encoding [4].

The ncd algorithm transformed amino acid sequences into numerical vectors by comparing each sequence with all others in the training dataset. This transformation yielded an n-dimensional numerical vector for each sequence, where 'n' represents the number of sequences in the training dataset. Each position in the vector signifies the ncd of the sequence concerning the corresponding sequence in the training dataset.

Due to the exponential computational complexity of the ncd algorithm, we used under sampling of the non-enzyme dataset to match the sample size of our enzyme dataset, ensuring balance in the positive instances within the training dataset.

When inferring unseen data, the ncd input vector was computed by comparing it to all sequences in the training dataset, resulting in an n-dimensional numerical vector. Consequently, the performance on new data heavily relies on the characteristics of the training dataset.

Figure \ref{fig:03} illustrates the performance comparison between the kNN algorithm using ncd vectors versus a random baseline.

Despite the mean F1 score of the kNN algorithm using ncd vectors resting at 0.728, it did not outperform the random baseline, which achieved an F1 score of 0.843, this observation suggests that the ncd approach exhibits inferior precision and recall compared to the random baseline. Additionally, both classifiers exhibit a low MCC score of 0.2 and 0.01, respectively, indicating that neither classifier performs better than random guessing.

### 7.1.2 Binary Classification: Support Vector Machine (SVM)

SVM is an algorithm that assigns labels to objects by learning from examples. [5] 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 [2]
2. Protein embedding vectors encoded by ESM2 [3]
3. One hot encoded protein sequence vectors [6]

### 7.1.2.1 SVM using ProtT5 encoded embedding vectors

//content

### 7.1.2.2  SVM using ESM2 encoded embedding vectors

//content

### 7.1.2.3  SVM using one hot encoded protein sequence vectors

//content