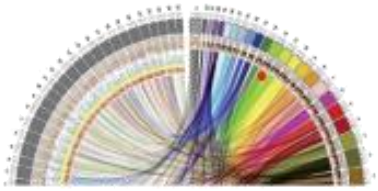


MSc in Bioinformatics

Universitat Autónoma de Barcelona

Machine learning based prediction of esterases' promiscuity

Supervised by

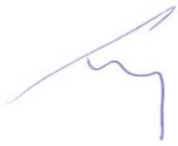
Víctor Guallar Tasies (BSC), director

Xavier Daura Ribera (UAB), tutor

Sign here:

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Sign here:



Universitat Autònoma de Barcelona

Index

Main structure

List of figures

List of tables

List of equations

Acknowledgements

I would like to thank Dr. Victor Guallar for giving me the opportunity to work in his laboratory and to his assistance on the ongoing research. His participation conjointly with Dr. Gerard Santiago has been of important help in setting up several of the objectives defined in this report. Furthermore, I would like to acknowledge Daniel Soler for his guidance during the development and execution of the project, without his expertise the research would have been more difficult and time consuming.

Abstract

Enzymes are of great interest for a vast variety of industries; however, the experimental characterization is very time consuming and expensive which has led to the development of plentiful of machine learning based platforms capable of predicting enzymes' function, *de novo*. Nonetheless, industrial enzymes need to adapt to nonbiological conditions while maintaining high activity, promiscuity and stereo-selectivity, properties that are not well covered, currently, by prediction technologies which means that their characterization still relies solely on experimentation.

This project has the intention of mitigating the problem by developing binary classifiers and multi-classifiers that can predict the promiscuity of esterases, one of the many industrially relevant enzymes.

The results are quite promising with all the developed classifiers achieving high metric scores and presumably without overfitting, however, the performance of the predictors will need to be confirmed with experimental data. If it is verified, the next step would be to extend the same process to other enzymes in demand, such as the phosphatases.

Introduction

For more than 90% of identified protein sequences, no characterization has been carried out experimentally due to the overwhelming speed at which new sequences are being added, meaning that most of them will be and have been annotated computationally[1].

An especially interesting category of proteins are enzymes, considering their important role in our daily lives and in a variety of industries such as food, detergent, agriculture, chemical, cosmetic and drugs.

Specifically, the demand of certain enzymes like lipases/esterases, proteases, hydrolases and polymerases that can work in industrial conditions has increased exponentially[2], thus requiring constant search for newer and better enzymes.

Knowing the cost and the difficulty for experimental function characterization, faster and more accurate in silico annotation is key to accommodate the vast amount of sequence data available[3].

Thankfully, a large number of methods have already been proposed and exists to identify and predict the class of enzymes, based broadly, on sequence, structure information or de novo predictions[4].

De novo predictions

De novo predictions have an undeniable advantage over other methods which is the fact that they don't rely on similarity to previously characterized enzymes but rather on properties or features that are shared by proteins with the same functions[4], which increases their applicability.

Within the *de novo* methods, machine learning algorithms seem to be the most extensively studied direction[5] especially for low similarity proteins[3,6].

Several types of algorithms have been applied with varied success: Linear regression, k nearest neighbor classifier (KNN), support vector machines (SVM), random forest (RF), artificial neural networks (ANN) and deep neural networks (DNN)[7,8], from which several

technologies have surfaced: ECPred (2018)[7], DEEPre (2017)[5], EzyPred[7] all of which are designed to predict enzyme EC numbers.

Nonetheless, biocatalysts in industrial settings are subject to nonbiological conditions, therefore, it is insufficient to know only their function but also if they will be able to withstand the harsh environment and industrial requirements. Basically, an ideal industrial enzyme should be thermostable, pH stable, halo stable (stable in high concentrations of salt) and promiscuous while keeping high chemo and stereo-selectivity[9].

In this case, fewer methods have been proposed to predict the enzyme's characteristics, and most of them focus on predicting thermostability, although some quite old studies exist about halo-stability[10] and pH stability[11]. Notably, most of these studies involve, again, machine learning except maybe for Scoop[12], which derives thermal stability curves with mathematical approximations.

Steps in machine learning projects

There are several steps to consider if one must carry out a machine learning project. The first and definitely one of the most important steps is the dataset and its processing.

1. SAMPLE COLLECTION

Needless to say, there is no project without a proper dataset and the first question one should answer is if the size of the data is enough to solve the biological problem in hand, because the more you have the better you will be able to generalize[13].

Additionally, one should make sure to minimize the experimental errors at annotating the samples since it will have a direct influence on the results and its credibility.

Nonetheless, there are domains where obtaining more samples might be time consuming or/ and very expensive, such would be the case for the characterization of enzymes, then one should focus on what the minimum size would be to achieve a reasonable predictive performance. Unfortunately, there is no exact answer to this question but, from another study in the domain of neuroimaging, it has been shown to still achieve good performances while using only a median of 80 subjects or samples[14]. In spite of that, it is hard to tell if in other domains the same applies.

There are other factors or steps that influence the performance and the reliability of the results in a machine learning project that can be easier to control.

2. FEATURE EXTRACTION

For biological sequences where the properties are encoded into the sequence itself, it is in a format that machine learning algorithms cannot understand directly, therefore a preprocessing step or feature extraction is necessary to parse the biological data into vectors of numerical values[15].

There are numerous features available for proteins that have been used throughout the literature which can be categorized loosely into 2 families:

Evolutionary based information

Evolutionary-based information Is regarded as a highly informative feature[16] and as such it has been widely used in different studies with a variety of applications, such as enzyme EC number prediction[5], catalytic site prediction[17], protein-protein interaction[16] etc.

This information encoded in the form of PSSM (position specific scoring matrix) is variable in length as it depends on the primary structure of the protein, which is unsuited as input for machine learning algorithms, therefore numerical transformations are needed to keep the matrix size constant.

3 types of transformations have been conceived by different authors such as row transformations, column transformations, and mixture of row and column transformation that sum up to over 18 different PSSM based features[16].

Primary sequence-based features

Primary sequence-based features are the classical ones, used to represent physicochemical properties such as hydrophobicity, charge distribution, volume, aminoacidic composition, secondary structure, disorder content, etc. They are regarded also as essential for training machine learning models specially when combined with the evolutionary information.

3. DATA CLEANING, SHUFFLING, SPLITTING AND SCALING

It is hard to decide which features are more important when extracting them from the raw sequence, so generally all of them will be used.

As a result, too many features will be generated which means selection methods are needed in order to avoid the **high dimensionality curse**, when there are far more features than samples, causing overfitting, as well as to **reduce computation time** and possibly to **increase the performance** of the models by eliminating noise[18].

However, before selection, it is a good practice to perform some data cleaning, that is, discarding possible inconsistencies, inaccuracies and **outliers**[13]**,** observations that differs from the general distribution of the sample population[19], since they will, most likely, influence the selection results. In addition, some data rearrangements are preferred previous to selection to prevent other problems such as overconfidence in the results, specially, for small datasets.

Shuffle

First, shuffling the data removes possible trends related to the order of the samples[13] that might, otherwise, affect the results of the selection for methods that rely on machine learning scores.

Split

Second, according to a study[14], in order to get robust and unbiased performance estimates of a prediction model, it is highly advised not to use the whole dataset for feature selection because it might generate overoptimistic results. Furthermore, it proved that a train/test split strategy or a nested cross-validation (ncv) strategy, essentially a train/test split but with different train and test sets each time, are the most robust ones regardless of sample size. In consequence, the dataset should be separated into 2 subsets, one used for feature selection and training and the other for evaluating the model.

Fig. 1 | The 2 splitting strategies that yield robust performance estimates, the model development, in blue, includes feature selection and hyperparameters tuning. Adapted from *Vabalas, A. et al*[14]

Scale

Finally, scaling the feature vectors into a normalized range is also a very necessary step since the algorithms are quite sensible to the magnitude of the features and it might cloud the importance of smaller but relevant features otherwise[13].

4. FEATURE SELECTION

Once taken care of the previous step, several selection methods are available, and it is advisable to use all of them to have different subsets of features to compare with:

Filter methods

Filter methods doesn't rely on the scores of the machine algorithms to calculate feature importance but rather assess: I) the degree of dependence of individual variables with the labels in case of statistical-based filters[20]; II) how much information (negative of entropy) it provides on the distribution of the samples of each class in case of the information theory based filters[21].

Wrapper and embedded methods

Wrapper and embedded methods, unlike the previous one, apply machine learning algorithms to search for those features that increase its performance, but they differ in their mechanism.

In embedded methods, the machine learning itself produces a relevance score of the features during the training, it is intrinsic to the algorithm such as random forest or XGBoost, however in wrapper methods, the scoring and the machine learning are 2 independent parts that are combined, for example RFE (recursive feature elimination)[18].

5. CHOOSING AND TUNING LEARNING ALGORITHMS

At this point, the dataset has been properly cleaned, rearranged, split and the features selected which means it is time to choose the algorithm and tune its hyperparameters.

The algorithm of choice will greatly depend on the biological problem and the nature of the dataset and, in this case, it involves classification tasks, that is, we want to identify the class, or the category of a sample, being binary, or multiclass.

The algorithms employed in this study are explained below:

Support vector machines (SVM)

SVMs are a type of supervised learning algorithm that performs binary classifications and are widely used due to its high accuracy and the ability to deal with high dimensional data[22].

To understand its essence, one needs to grasp its 4 basic components:

- **The separating hyperplane**: The data is represented as points in the space and the separation between the 2 (binary classifications) classes of data is performed through a line if we are dealing with 2 dimensions, a plane if is in 3D and a hyperplane when we deal with high dimensional data[23].
- **The maximum margin hyperplane**: The hyperplane method is not unique to SVM but, unlike other classifiers, it tries to maximize the distance with the nearest data point (also called as margin)[23].

- **The soft margin**: Many data sets cannot be separated cleanly, few data points from one class may be similar to the other class, to handle these cases SVM algorithms adds a soft margin that allows some points to be in the other class without modifying the results. However, we must control how many misclassifications are allowed and how far from the hyperplane they can be[23].
- **The kernel function**: The data will not always be separable with a straight line (if we are in 2D), so sometimes kernel functions, a mathematical trick that projects data from a low dimensional space to a higher dimensional space, can be applied since it is proven that for any given data set there exist a kernel function that will allow the data to be lineally separated[23].

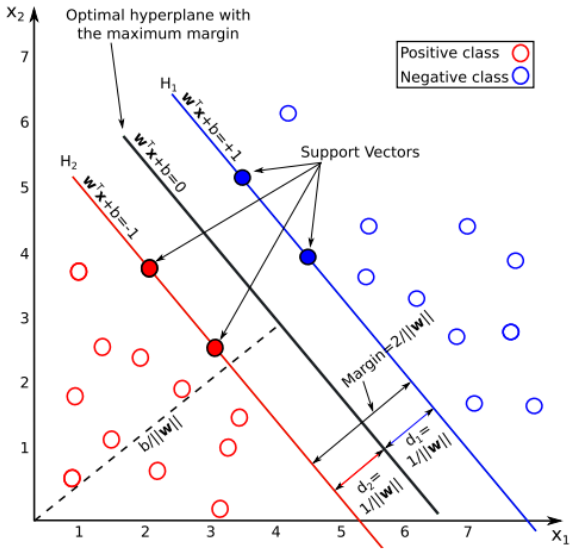


Fig.2 | A graphical representation of the workings of an SVM. By Alaa Tharwat[24]

After the explanation it is easy to understand that there are hyperparameters that needs to be searched and optimized by try and error such as the size of the soft margin, that is the penalization of misclassifications, named C, the type of kernel function and a kernel parameter gamma[24].

K-Nearest Neighbours (KNN)

KNN is one of the simplest and most common classifiers, yet its performance competes with the most complex classifiers in the literature and it is placed among the top 10 methods in data mining.

The core of this method depends mainly on measuring the distance between the test examples and the training examples and the classification is produced based on how many data points of each class are closest to the test sample. Therefore, the performance will depend greatly on the type of distance measurement used[25]. KNN is simple but proved to be highly efficient and effective algorithm

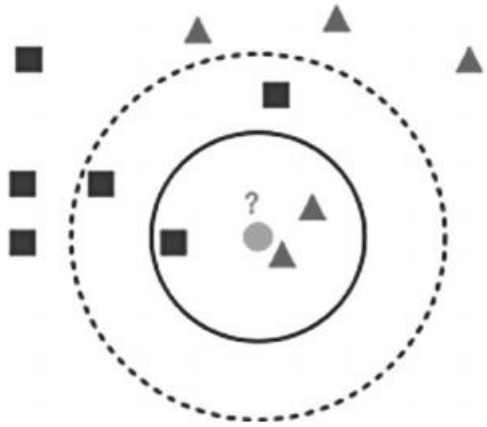


Fig.3 | How the KNN classifies, the triangles and the squares represent different classes. In this case the new data point will probably be classified as triangle, since more points are closer. By Abu Alfeilat et al[25].

In KNN, the hyperparameters to be tuned would be the distance metric, for instance the Minkowski distance, a very used and generalized metric defined as[25]:

$$D(x,y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} \qquad (Eq.1)$$

where P is a positive number and also a hyperparameter, or the Canberra distance presented as[25]:

$$Can(x,y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|} \qquad (Eq.2)$$

and finally, the number of neighbours, K.

Linear models

Linear models are also one of the simplest algorithms and the concept behind them is very easy to understand. It is the linear relationship between one or more features (X) and one dependent label[26] (Y) and although most of the time they are used for regression tasks, there are variants such as the implementation of Ridge Classifier from Scikit-learn that can be applied in classification tasks by transforming the binary classes into {-1,1} and treating it as a regression problem[27].

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_i X_i \qquad (Eq. 3)$$

The hyperparameter to be tuned, in Ridge Classifier is α, which penalizes the size of the coefficients β_i.

Again, there are no clear answers on which algorithm is the best, so usually, several different models are trained simultaneously and compared or even combined.

6. PERFORMANCE EVALUATION

As a result of the previous step, several models will be trained but the final goal is to be able to predict the class of future incoming samples as accurately as possible, so logically metrics to measure, compare and identify the best performing models are essential.

Let's start by defining the 4 fundamental quantities in classification that are the basis for all the metrics.

- True negatives (TN): The number of predicted negatives that are truly negative
- True positives (TP): The number of predicted positives that are true
- False positives (FP): The number of predicted positives that are negative
- False negatives (FN): The number of predicted negatives that actually are positive

From the above quantities we can define the metrics most used in the literature such as:

- Accuracy: the proportion of correctly classified simples[28]:

$$acc = \frac{TP + TN}{FP + FN + TP + TN} \qquad (Eq. 4)$$

Accuracy: worst value= 0, best value = 1

- Precision: The proportion of true positives in all the predicted positives[28]

$$Pr = \frac{TP}{FP + TP} \qquad (Eq.5)$$

Precision: worst value= 0, best value = 1

- Recall: The probability of detecting true positives[28]

$$Re = \frac{TP}{FN + TP} \qquad (Eq.6)$$

Recall: worst value= 0, best value = 1

- F1: a combination of precision and recall[28]

$$F1 = \frac{2 * Pr + Re}{\mathrm{Pr} + Re} \qquad (Eq.7)$$

F1: worst value= 0, best value = 1

- Matthews correlation coefficient (MCC): It is only high when negative and positive classes are well identified so it serves as a general measure of the quality of a model[13].

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \qquad (Eq.8)$$

MCC: worst value= -1, best value = 1

Apart from comparing the performance of the predictions on testing samples, the trained models should also predict on samples used for training because if the metric scores are not equal or higher than those of the training samples, it would mean that the classifier has overfitted and cannot perform as well for samples other than the training.

7. APLICABILITY DOMAIN

Now that the best models have been chosen because of their high scores in different metrics, a new platform or a new technology can be built around them. However, the reliability of the predictions is limited. There is a restriction on the applicability of the model to those samples that are similar enough to the training samples, because otherwise it

would be predicting something that it has not seen and fitted before so logically its prediction on that data would not be trustworthy.

This limitation is the applicability domain[29] and as a consequence, any dissimilar data should be filtered out before applying the new technology.

The focus of this project

Now, let's focus on the research we are aiming at here. This specific study can be seen as a continuation of another project that the group was involved in where the goal was to find the determinants of esterase promiscuity.

Esterases, which catalyze the hydrolysis of ester bonds, are very relevant industrially and are involved in many applications such as the synthesis of chiral drugs for pharmaceuticals or the production of various acids that are widely used in food, beverage, perfumes industries and so on[30]. Thus, the identification of newer and more promiscuous ester hydrolyses might have a positive impact on society.

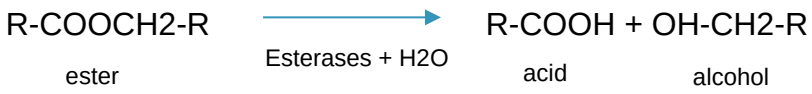


Fig. 4 | The hydrolysis of an ester to an acid and alcohol is catalyzed by an esterase

To assess the substrate range of the 145 diverse esterases[31] generated during the previous project, a customized library of 96 different esters, consisting of mostly aromatic and alkyl esters, were used.

After correlating the specificity of the enzymes with structural analysis of their catalytic center, it was found that the active site's effective volume, which is the active site cavity volume corrected by the relative solvent accessible area (cavity volume/SASA), was able to identify with a 94% accuracy those esterases with activity for 20 or more substrates[31]. Nevertheless, this method requires 3D structures that are difficult to obtain which greatly reduces its applicability. As an alternative, it was proposed to generate a model that could predict promiscuity based solely on sequence information since it is known that the primary structure of a protein greatly determines its 3D structure.

Objectives

The main goal of this Master Thesis is:

- ✓ To generate a machine-learning based prediction model of esterases promiscuity based on sequence information alone by using the experimental data produced in the other project[31], while alleviating the scarcity problem of developed technologies that can predict enzyme's industrial fitness.

    - A simple binary classifier will demonstrate that it is possible to generate a model using the mentioned dataset and the features extracted.

    - A multi-classifier will provide finer predictions and might be preferred over the binary classifier.

Methodology

1. DATASET

As stated before, this project can be treated as a continuation of a previous study on esterase promiscuity, thus the dataset employed is the same and it is formed by 145 phylogenetically, environmentally and structurally diverse microbial ester hydrolases, plus 2 commercially available esterases with an average pairwise sequence identity of 13.7%[31].

- For the classification, the dataset was separated into 2 classes, taking as threshold 20 substrates; promiscuous if it was able to hydrolyze 20 esters or more and non-promiscuous if less, yielding 67 promiscuous or positives and 80 negatives.
- For the multi-classification, the dataset was split into 3 classes, 30 substrates or more was considered highly promiscuous, less than 30 but more than 10 substrate as moderately promiscuous and less than 10 as non-promiscuous according to the same definitions in the study[31], as a result 32, 71 and 44 samples ended up in each class respectively.

2. FEATURE EXTRACTION

2 webservers Possum[16] and ifeature[32] were used to extract evolutionary information and physicochemical properties, respectively, from the protein sequences.

iFeature

iFeature is capable of generating 53 different types of descriptors, from which 32 features types were extracted resulting in a total of 4.662 number of features or dimensions, the rest of feature types were discarded because they can only be applied to sequences of the same length.

Table 1 | The features extracted from iFeature[32] used in this project

iFeaure descriptors	definition
Grouped amminoacid composition	aa composition and its variants are the % of each aa, dipeptide, tripeptide in the sequence[33].
Grouped dipeptide, tripeptide composition	
K-spaced aa group pairs composition	
Moran Autocorrelation	It describes the correlation between 2 aa in terms of 8 aa properties[33]
Geary Autocorrelation	
Normalized Moreau-Broto Autocorrelation	
Composition	The 20 aas are divided into 3 groups for each of the 7 properties considered and then it looks the class composition, how frequent are the transitions to other classes and the overall distribution along the sequences[33].
Transition	
Distribution	
Conjoint triad	It clusters 20 aa into 7 classes, it treats tripeptides as an unit and counts the frequency of each unit in the sequence[34].
K-spaced conjoint triad	
Sequence-order coupling number	It can be used to represent aa distribution patterns of a specific physicochemical property along a protein[33]
Quasi sequence order	
Pseudo-aa-composition	It counts the composition of the aa in the sequence, but it adds a coupling term to not lose order information of the aa[35].
Amphiphilic Pseudo aa composition	
Pseudo K-tuple reduced aa composition type1 to 16	

Possum

Possum is a server that generates features based on the PSSM profiles of each protein by applying different matrix transformations to make it length independent. 18 features, all the default features, were extracted and 4 more by varying the default values resulting in 18.730 dimensions.

Table 2 | Features extracted from possum[36]

Mathematical transformations applied	Possum descriptors
Row transformations	AAC-PSSM
	D-FPSSM
	S-FPSSM
	AB-PSSM
	RPM-PSSM
	smoothed-PSSM window size (5)
	smoothed-PSSM window size (7)
	smoothed-PSSM window size (9)
	PSSM-composition

Column transformations	DPC-PSSM
	k-separated-bigrams-PSSM
	tri-gram-PSSM
	EEDP
	TPC
Mixed of row and column transformations	EDP
	Pse-PSSM (ξ = 1)
	Pse-PSSM (ξ = 2)
	Pse-PSSM (ξ = 3)
	DP-PSSM
	PSSM-AC
	PSSM-CC
	RPSSM

3. DATA CLEANING, SHUFFLING, SPLITING AND SCALING

Cleaning

After generating the features, some cleaning was needed because many columns had many zeros or identical values in most of the rows which carried little information.

```
to_remove = []
for col in all_data.columns:
    if all_data[col].astype(bool).sum(axis=0) < 140:
        to_remove.append(col)
all_data.drop(to_remove, axis=1, inplace=True)
nunique = all_data.apply(pd.Series.nunique)
cols_to_drop = nunique[nunique < 137].index
all_data.drop(cols_to_drop, axis=1, inplace=True)
```

Code 1

As a result, 4.662 iFeature features and 18.730 possum features were reduced to 2.572 and 14.601 features respectively.

Shuffling, splitting and scaling

The dataset was shuffled and split together using the *train_test_split (TTS)* class in **scikit-learn**, the python's default machine learning library and the features scaled in the range of [0,1] using *MinMaxScaler* class of the same library.

Notably, the shuffle and split strategies were different for binary and multi-classification. For the former only one major split using the mentioned class was performed and slightly different train/test sets generated by changing the random state variable within the function (Fig 1. A). For the multi-classification, however, besides the previous approach, 2 more systems were tested:

a) By combining *TTS* class and the *Kfolds* class in scikit-learn, the 2 schemes depicted in Fig 1, A and B, were merged. The dataset was first split using *TTS*, with 1 random state, and then the resulting training set was further split into 6 train and non-overlapping test sets with *Kfolds*.
b) In approach B, *TTS* was omitted, and the dataset was first shuffled using the *Shuffle* class in **scikit-learn**, then split with *Kfolds*, essentially it would be a nested cross-validation (Fig 1. B).

In regard to outlier detection, **PyoD**[19] was the library used to recognize the abnormal samples from the distribution of the features. This library contains a comprehensive set of more than 20 different algorithms, some based on machine learning, from which 6 were selected arbitrarily and the results summed together to find those predicted to be outliers most of the time.

Table 3 | Definitions of the different outlier detection algorithms used[37]

Outlier models	Definitions
KNN (K nearest neighbours)	For a sample its distance to its kth nearest neighbor could be viewed as the outlying score
ABOD (Angle based outlier detection)	The variance of its weighted cosine scores to all neighbors could be viewed as the outlying score
CBLOF (Clustering Based Local Outlier Factor)	It generates clusters and decides if it is an outlier based on the size of the cluster the point belongs to as well as the distance to the nearest large cluster
LOF (Local Outlier Factor)	It measures how isolated the sample is with respect to the surrounding neighborhood
IForest (IsolationForest)	It has a tree structure; it detects anomalies by noticing those samples that are easily split at the beginning of the root of the tree.
HBOS (Histogram-based outlier detection)	It detects anomalies by building histograms around the samples and those out of the boundary are anomalies.
Feature Bagging	It is a meta estimator that fits a number of base detectors (LOF) on various sub-samples of the dataset and use averaging methods to improve the predictive accuracy

4. FEATURE SELECTION AND MODEL TRAINING

Even with the cleaning, the number of original features remained exceedingly high, a total of 17.173 dimensions, therefore selection was needed to avoid overfitting.

5 libraries were used **ITMO_FS**(1)[38], **Boruta**(2)[39], **Scikit-feature**(3)[40] , **Scikit-learn**(4) and **XGBoost**(5)[41] to implement 10 different feature selection methods.

- **Statistical-based filters**: (1) Chi square, (4) fisher score
- **Information theory-based filters**: (1) Information gain (IG), (3) MRMR (minimum redundancy maximum relevancy), CIFE (conditional infomax feature extraction), (4) mutual information
- **Wrapper methods**: (4) RFE (recursive feature elimination) combined with a linear model Ridgeclassifier or SVM
- **Embedded methods**: (2) Boruta, (4) Random forest (RF), (5) XGBoost

Binary classifier

For binary classifiers, each of the stated methods was performed independently for the 2 families of features and combined afterwards. The total number of dimensions was reduced to 70 features, 30 iFeatures and 40 possum, which is less than a ½ of the number of samples, a proportion that was proven to decrease overfitting[14].

The reason why possum features were given more importance was that many authors[16,42,43] seemed to claim that evolutionary information was more informative. In addition, the study; where the dataset came from, proved that phylogeny could be an indicator of promiscuity[31].

Moreover, with the intent of decreasing even further the complexity of the models, new feature subsets were constructed by subtracting 10 features (5 iFeature and 5 possum) each time from the initial feature set, until 20, resulting in 10 (filters) X 6 (numbers) total feature sets.

Each of these sets were then used to train each of the 3 learning algorithms with 5 different random states in the case of Ridge Classifier and SVM while in the case of KNN, a preprocessing step of the scaled features using Neighbourhood Component Analysis

(NCA) function in **scikit-learn** was added or omitted to compare their difference[44], yielding a total of 720 trained and tuned models.

$$(2\ svm\ and\ ridge \cdot 5\ random\ states + 2\ knn\ and\ NCA) \cdot 60\ sets = 720\ models \quad (Eq.9)$$

Multi-classifier

For multi-classifiers, besides the previous selection strategy that generates 720 models, 2 more splitting and selection methods were tested as mentioned before:

a) For the splitting approach A, features from the 2 families were first combined and then selection was applied afterwards, and again new feature subsets were constructed using the same idea from the first strategy. However, this time the process was repeated for each of the splits generated by *Kfolds*, returning a total of 6 (folds) X 9 (without Boruta selection) X 6 feature sets.

   The subsets were used to train each of the 3 algorithms and an additional preprocessing step for KNN which generates a total of 1296 models

$$4\ algorithms + NCA\ \cdot 324\ sets = 1296\ models \qquad (Eq.10)$$

b) For approach B, TTS were omitted but the rest remains the same with the difference that more data were available for the train/test sets of each fold. Again, at each of the 6 folds, 54 features subsets were generated and then used to train the algorithms yielding a total of 1296 models.

A remark, that is unrelated but relevant, should be made about the features used to detect outliers. It was possible to utilize the original features but, in this case, the selection strategy in binary classifications was applied to generate 60 different feature sets for both the binary and multi classifications.

Then they were used for outlier detection and once the outliers were eliminated, the selection strategies were applied again on the original features since their presence can affect the selection.

As a result, 5 outliers were eliminated during the training of binary classifiers and 4 outliers for the multi-classifiers.

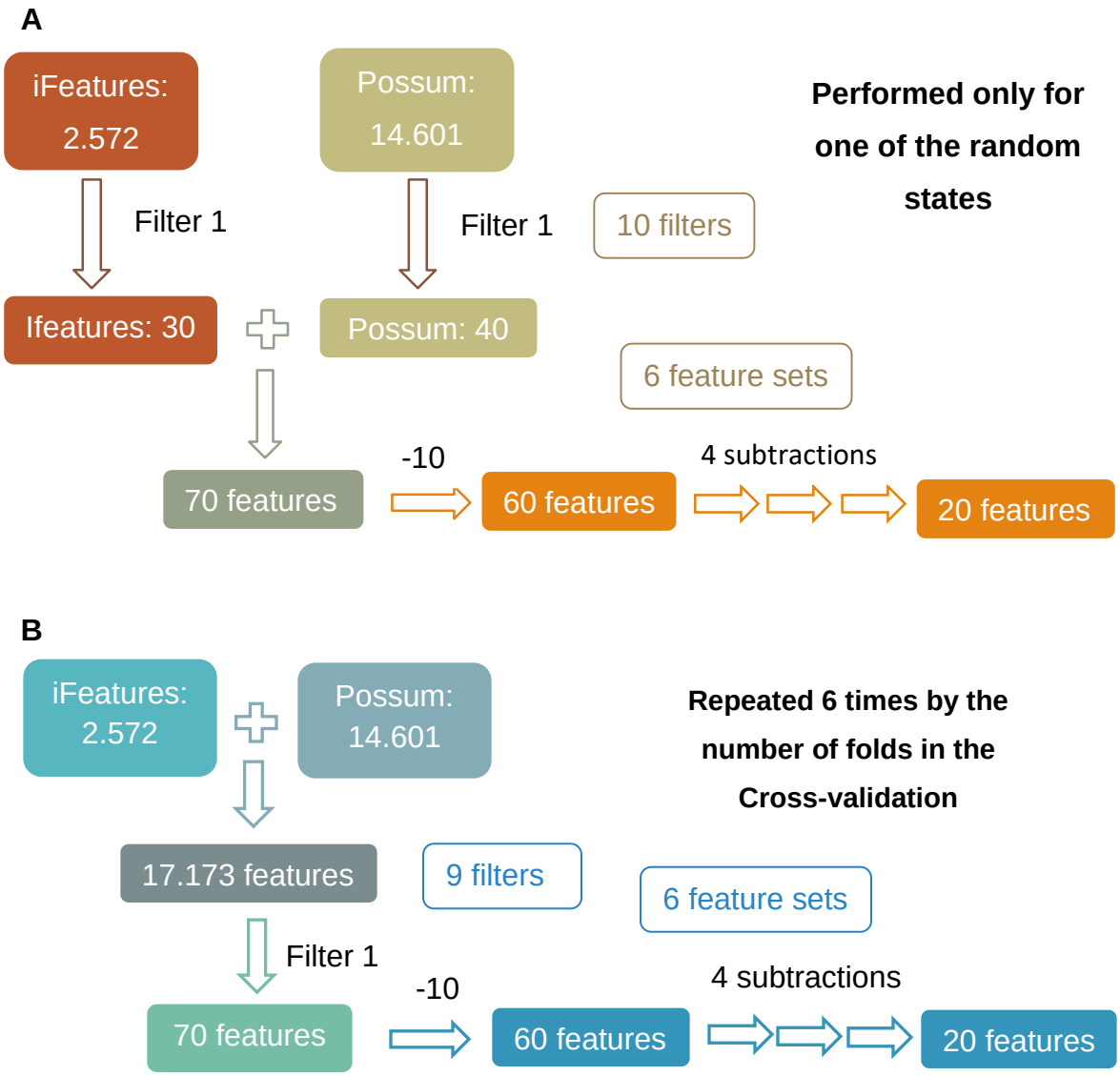


Fig 5 | Feature selection strategies. **A)** represents the selection approach for binary classification, a total of 60 feature sets are generated on 1 random state, in the other states the models are trained using the same features. **B)** represents the feature selection strategy for the multi-classification, it is the same for both splitting approaches but, the difference is that there are more samples available for feature selection in the splitting approach b).

Results and discussion

BINARY CLASSIFICATION

Following the split, feature selection and training strategies explained in the methodology section, 60 feature sets and 720 binary models were generated and compared using all the metrics described in step 6 of the introduction to find the subset that produced the best performing models for each algorithm.

The following criteria were used to identify the optimal feature sets: first the overall MCC and the number of dimensions, the fewer the better as it reduces the complexity of the models, and then by the precision score of class 1, those considered promiscuous.

Only these 2 metrics are presented because they are the most informative for this project. The overall-MCC represents how well the TN and TP are identified, so a general measure of the model's performance.

It is preferred over F1 because it takes into consideration the effects of class imbalances[13] in the dataset. Additionally, class 1 precision assesses how reliable are the predictions, allowing to identify promiscuous new enzymes, the ultimate goal of this project.

Surprisingly, there was a single feature set, filtered using Chi squared and further purged into 20 dimensions (chi-20), that performed the best in SVM and Ridge, independently of the random states as displayed in **table 4 and 5**.

Unlike the others, KNN had the best precision scores while training with **Random Forest (RF)** features with 30 dimensions as seen later in **table 6**.

SVM

Table 4 | Overall MCC and precision scores at different random states using chi-20 as input to an SVM, in addition to its hyperparameters

	Hyperparameters			Overall		Class 1	
Random states	**kernel**	**C**	**gamma**	**Test MCC**	**Train MCC**	**Train Precision**	**Test Precision**
20	RBF	0,31	0,91	0,57	0,50	0,79	0,77
40	RBF	0,61	0,91	0,59	0,51	0,772	0,89
70	RBF	0,61	0,61	0,60	0,47	0,769	0,83
80	RBF	3	0,31	0,68	0,56	0,825	1
90	RBF	0,91	0,31	0,62	0,54	0,804	1
			Mean	0,61	0,52	0,79	0,89
	Standard deviation (std)			0,04	0,03	0,02	0,10

Table 4 shows that SVMs would be a good choice as the prediction model for esterase promiscuity, since both the overall test MCC and precision scores are considerably high, with a mean around 0.61 and 0.9 respectively, and consistently superior than the training scores, with a mean around 0.52 and 0.8 respectively. This indicates lack of overfitting and it is true regardless of the train/test sets examined here, except for the first random state where the training precision is slightly superior than the test precision.

With regards to the best performing hyperparameters there is a clear preference for the gaussian type kernel function, RBF (radial basis function), where the distance between all samples is rescaled according to its width or variance and where higher variances leads to smaller distances between samples[24].

In addition, there is also a tendency towards smaller values of gamma and C. Gamma, being a parameter of RBF, controls the width of the gaussian and it is the inverse of the variance[24], which means smaller values will lead to gaussians with higher width and samples that are further away in the feature space to appear more similar. Regarding C, that penalizes misclassifications, smaller values will generate margins that are more permissive.

Ridge Classifier

Table 5 | Overall MCC and precision scores at different random states using chi-20 as input to a Ridge Classifier, in addition to its hyperparameters.

	Hyperparameter	Overall		Class 1	
Random states	**α**	**Test MCC**	**Train MCC**	**Train Precision**	**Test Precision**
20	8	0,64	0,55	0,85	0,83
40	3	0,68	0,54	0,82	1
70	5	0,60	0,53	0,83	0,83
80	0,51	0,68	0,54	0,82	1
90	2	0,62	0,56	0,84	1
	Mean	0,64	0,54	0,83	0,93
	std	0,03	0,010	0,01	0,09

Similarly, the Ridge linear model could also be used to predict promiscuity because of the same reasoning as in SVM, and might even be preferred over the previous one due to higher mean performances, 0.64 for MCC and 0.93 for precision, and lower deviations as evidenced in **table 5**. However, in this case there is no clear tendency with the α hyperparameter.

KNN

Table 6 | Overall MCC and precision scores for both class 1 and class 2 using RF-30 with different processing steps as input to a KNN, in addition to its hyperparameters.

	Hyperparameters			Overall		class 1	
Processing	**Neighbours**	**p**	**Distance**	**Test MCC**	**Train MCC**	**train-precision**	**test precision**
NCA	3	3	Minkowski	0,71	0,94	1	0,84
no-NCA	7	5	Minkowski	0,78	0,63	0,87	0,91

Finally, KNN, if trained using RF features, would also be a great option as a prediction model, with a test MCC of 0.78 and test precision of 0.91. No NCA preprocessing seems to be needed since it causes overfitting as indicated by the scores of the training samples being superior than the test samples.

Regarding the hyperparameters, the Minkowski distance with a p = 3 and 10 neighbours seems to be the optimal choices.

It is hard to tell if KNN performs better than the others merely by looking at this one score, even though it is much higher than the others specially regarding the overall MCC and further studies, possibly experimental verification, might be required to discern which one is better. It might even be preferable to combine the 3 algorithms together into a single predictor and classify via a voting system. Such procedure should reduce the error and increase precision, given that an esterase would only be considered promiscuous if 2 or model models predicts so.

MULTI-CLASSIFICATION

In multi-classification more strategies needed to be tested because, by applying the binary strategy, it became clear that no feature set were able to generalize for all the random states considering that, for some of them, training MCC scores were higher than the test MCC scores, a clear indication of overfitting.

After speculating about the source of the problem, it was theorized that maybe the feature selection should be performed with the features from the 2 web servers combined (Fig. 5 B). Such a set up would facilitate the selection methods to consider the most important features from the whole set and not only a part of it.

In addition, it was thought that feature selection should also be performed at each cross-validation round or fold (Fig. 5 B), unlike in binary classification where it was only performed at one random state and used for the rest of the states (Fig. 5 A).

Because, by combining the models generated at each fold with different features into a single predictor, it would potentially be the same as having 1 feature set that can generalize to all random states. In consequence the splitting approach A and feature selection strategy B was implemented, however the results were not satisfactory. There were still some folds that had training MCC scores higher than the test MCC scores for all the subsets generated.

After some considerations, It was hypothesized that the first train/test split in splitting approach A might had been the problem since it was reducing the samples available for testing and training of an already small dataset, from 120 and 23 samples for training and testing respectively to 102 and 20 samples. However, it was desirable to reserve a test set to compare the performance of individual models with that of the single predictor.

Consequently, strategy B was developed and implemented for the 3 algorithms and finally, positive results were returned although only for KNN as seen in **table 7**. The criteria for choosing the best performing models and feature subsets are similar than in binary classification with the addition of the precision of a class 2, those highly promiscuous.

KNN

Table 7 | Overall MCC and precision scores of class 2 and class 1 for different KNN models and feature sets, in addition to its optimal hyperparameters.

		Hyperparameters				Overall		Class 2		Class 1	
Split	**Features**	**Processing**	**Neighbours**	**p**	**Distance**	**MCC**	**train MCC**	**precision**	**train prec.**	**precision**	**train prec.**
0	xgboost-20	NCA	9	1	Canberra	0,49	0,46	0,66	0,64	0,84	0,63
1	RFE-20	No-NCA	3	2	Minkowski	0,67	0,64	0,71	0,67	0,81	0,72
2	IG -50	NCA	8	1	Minkowski	0,63	0,40	0,33	0,58	0,92	0,63
3	RFE-40	no-NCA	8	5	Minkowski	0,62	0,58	0,85	0,74	0,63	0,72
4	RForest-30	no-NCA	10	1	Minkowski	0,58	0,40	1	0,77	0,66	0,65
5	RFE-20	NCA	5	1	Canberra	0,81	0,77	1	0,87	0,72	0,80
					Mean	0,63	0,54	0,76	0,71	0,77	0,69
					std	0,10	0,14	0,25	0,10	0,11	0,06

As displayed, the overall MCC is consistently higher for the testing samples, with a mean score of 0.63, a demonstration of the models' general performance. Nevertheless in the case of precision, even though the mean testing scores for class 1 and class 2, 0.77 and 0.76 respectively, surpass those of the training samples, there are splits where that is untrue, for example the precision of class 2 in split 2. A s results, standard deviation is high, around 0.25. However, each of the splits should be treated independently, because they are unrelated models using different feature sets, so the individual scores might be more informative.

Despite this, it is still a major improvement compared to the results from strategy A, where testing MCC scores for some splits were lower than those of the training sets. This shows the importance of the size of the dataset specially for a multi-classifier since the samples available for each class is smaller in comparison to a binary classification.

Concerning the hyperparameters, there is a tendency for higher numbers of K or neighbours probably because of the low similarity between the samples, in addition to a preference for the Minkowski distance and smaller numbers of P. Regarding NCA preprocessing, it is only preferable in certain splits.

With respect to why only KNN succeeded, the most logical answer would be that the algorithm's biases were favorable in this case. However, if more samples were provided for each class, the other algorithms would, likely, perform well too similarly as in binary classification.

As a side note, the same strategies from both binary and multi-classification were applied to phosphatases to see if they were transferable, because in the case they are, it would potentially mean that the results could be replicated for all the other industrially interesting enzymes. Nevertheless, none of the approaches tested here were reproducible in phosphatases (results not shown) probably due to the dataset composition.

Unlike the esterases, where the majority of the samples share little sequence similarity, in the phosphatase's dataset, there are clusters of samples that share more than 40% of sequence similarity but less than 40% if they are from different clusters. In addition, there is one or two clusters that contain most of the samples leaving the others scattered around the other clusters[45]. Thus, proving the importance of the dataset for any machine learning project, although there might be other causes to this issue such as that the features used are not suited for phosphatases, even though is already a very thorough list.

Still, the fact that different algorithms were successfully trained in binary classifications for esterases implies that promiscuity might be a property that can be deduced from the sequence, given the right dataset. This possibility opens a new avenue that might make massive sequence-wide screening of promiscuous ester hydrolases feasible.

Conclusions

In conclusion, the objectives set up for this thesis were accomplished satisfactorily. For the binary classification all the 3 algorithms tested were successfully trained and achieved relatively high mean precision and MCC scores individually.

In contrast, for multi-classification only KNN models were generated without overfitting, probably, due to the fact that the design of the algorithm was more suited in this case. Nevertheless, it has achieved acceptable mean test performances, 0.63 ± 0.2 for MCC, 0.77 ± 0.22 for precision of class 1 and 0.76 ± 0.50 for precision of class 2 despite having less samples per class. Overall, having a sequence predictor for promiscuity in hydrolases will introduce a significant boost in future enzyme bioprospecting.

Future perspectives

The project is not finished, there are still several tasks left by the time of the writing, such as building the single predictor from all the binary classifiers or the single multi-classifier from all the KNN models.

Preliminary results suggest that the ensemble of binary classifiers can produce a MCC score of 0.76 in the test set compared to the mean score of 0.67 ± 0.09 of the 3 separate models, thus demonstrating that the ensemble learner can indeed produce better results.

In addition, the identification of the applicability domain of the models is also an essential step to be developed in the future if they are to be used for making predictions on uncharacterized esterases. Moreover, the performance of the predictors needs to be verified by experimental data and models retrained if they differ substantially.

Finally, esterases are the first enzymes of industrial interest tested in this project but eventually, the group’s interest would be to extend the process to other enzymes such as phosphatases, from which promiscuity data is also available.

Bibliography

1. Peled, S. *et al.* De-novo protein function prediction using DNA binding and RNA binding proteins as a test case. *Nat. Commun.* **7**, 1–9 (2016).

2. Kamble, A., Srinivasan, S. & Singh, H. In-Silico Bioprospecting: Finding Better Enzymes. *Mol. Biotechnol.* **61**, 53–59 (2019).

3. Radivojac, P. *et al.* A large-scale evaluation of computational protein function prediction. *Nat. Methods* **10**, 221–227 (2013).

4. Punta, M. & Ofran, Y. The rough guide to in silico function prediction, or how to use sequence and structure information to predict protein function. *PLoS Comput. Biol.* **4**, (2008).

5. Li, Y. *et al.* DEEPre: Sequence-based enzyme EC number prediction by deep learning. *Bioinformatics* **34**, 760–769 (2018).

6. Watanabe, N. *et al.* Exploration and Evaluation of Machine Learning-Based Models for Predicting Enzymatic Reactions. *J. Chem. Inf. Model.* (2020) doi:10.1021/acs.jcim.9b00877.

7. Dalkiran, A. *et al.* ECPred: A tool for the prediction of the enzymatic functions of protein sequences based on the EC nomenclature. *BMC Bioinformatics* **19**, 1–13 (2018).

8. Camacho, D. M., Collins, K. M., Powers, R. K., Costello, J. C. & Collins, J. J. Next-Generation Machine Learning for Biological Networks. *Cell* **173**, 1581–1592 (2018).

9. Thapa, S. *et al.* Biochemical Characteristics of Microbial Enzymes and Their Significance from Industrial Perspectives. *Mol. Biotechnol.* **61**, 579–601 (2019).

10. Zhang, G. & Ge, H. Support vector machine with a Pearson VII function kernel for discriminating halophilic and non-halophilic proteins. *Comput. Biol. Chem.* **46**, 16–22 (2013).

11. Lin, H., Chen, W. & Ding, H. AcalPred: A Sequence-Based Tool for Discriminating between Acidic and Alkaline Enzymes. *PLoS One* **8**, (2013).

12. Pucci, F., Kwasigroch, J. M. & Rooman, M. SCooP: an accurate and fast predictor of protein stability curves as a function of temperature. *Bioinformatics* **33**, 3415–3422 (2017).

13. Chicco, D. Ten quick tips for machine learning in computational biology. *BioData Min.* **10**, 1–18 (2017).

14. Vabalas, A., Gowen, E., Poliakoff, E. & Casson, A. J. Machine learning algorithm validation with a limited sample size. *PLoS One* **14**, 1–21 (2019).

15. Saidi, R., Aridhi, S., Maddouri, M. & Nguifo, E. M. Feature extraction in protein sequences classification : A

new stability measure. *2012 ACM Conf. Bioinformatics, Comput. Biol. Biomed. BCB 2012* 683–689 (2012) doi:10.1145/2382936.2383060.

16. Wang, J. *et al.* POSSUM: A bioinformatics toolkit for generating numerical sequence feature descriptors based on PSSM profiles. *Bioinformatics* **33**, 2756–2758 (2017).

17. Pai, P. P., Shree Ranjani, S. S. & Mondal, S. PINGU: Prediction of eNzyme catalytic residues usinG seqUence information. *PLoS One* **10**, 1–15 (2015).

18. Guyon, I. & Elisseefl, A. An introduction to feature extraction. *Stud. Fuzziness Soft Comput.* **207**, 1–25 (2006).

19. Zhao, Y., Nasrullah, Z. & Li, Z. PyOD: A python toolbox for scalable outlier detection. *J. Mach. Learn. Res.* **20**, 1–7 (2019).

20. Guyon, I., Gunn, S., Nikravesh, M. & Zadeh, L. A. *Feature Extraction*. *Studies in Fuzziness and Soft Computing* vol. 207 (2006).

21. Zadeh, L. A. Studies in Fuzziness and Soft Computing: Foreword. in *Studies in Fuzziness and Soft Computing* vol. 261 121–122 (2010).

22. Ben-Hur, Asa; Weston, J. A User's Guide to Support Vector Machines. in *Springer Protocols - Methods in Molecular Biology 609* vol. 1415 17 (2010).

23. Noble, W. S. What is a support vector machine? *Nat. Biotechnol.* **24**, 1565–1567 (2006).

24. Tharwat, A. Parameter investigation of support vector machine classifier with kernel functions. *Knowl. Inf. Syst.* **61**, 1269–1302 (2019).

25. Abu Alfeilat, H. A. *et al.* Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. *Big Data* **7**, 221–248 (2019).

26. Lee, W. Python® Machine Learning. in *Python® Machine Learning* 119–146 (2019). doi:10.1002/9781119557500.

27. 1.1. Linear Models — scikit-learn 0.23.1 documentation. https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression-and-classification.

28. Kubat, M. An Introduction to Machine Learning. in *An Introduction to Machine Learning* 213–233 (2017). doi:10.1007/978-3-319-63913-0.

29. Sahigara, F., Ballabio, D., Todeschini, R. & Consonni, V. Defining a novel k-nearest neighbours approach to assess the applicability domain of a QSAR model for reliable predictions. *J. Cheminform.* **5**, 1 (2013).

30. Panda, T. & Gowrishankar, B. S. Production and applications of esterases. *Appl. Microbiol. Biotechnol.* **67**,

160–169 (2005).

31. Martínez-Martínez, M. *et al.* Determinants and Prediction of Esterase Substrate Promiscuity Patterns. *ACS Chem. Biol.* **13**, 225–234 (2018).

32. Chen, Z. *et al.* IFeature: A Python package and web server for features extraction and selection from protein and peptide sequences. *Bioinformatics* **34**, 2499–2502 (2018).

33. Ong, S. A. K., Lin, H. H., Chen, Y. Z., Li, Z. R. & Cao, Z. Efficacy of different protein descriptors in predicting protein functional families. *BMC Bioinformatics* **8**, 1–14 (2007).

34. Wang, Y. C., Wang, Y., Yang, Z. X. & Deng, N. Y. Support vector machine prediction of enzyme function with conjoint triad feature and hierarchical context. *BMC Syst. Biol.* **5**, 1–11 (2011).

35. Chou, K.-C. Pseudo Amino Acid Composition and its Applications in Bioinformatics, Proteomics and System Biology. *Curr. Proteomics* **6**, 262–274 (2009).

36. POSSUM | Server Page. http://possum.erc.monash.edu/server.jsp.

37. All Models — pyod 0.8.1 documentation. https://pyod.readthedocs.io/en/latest/pyod.models.html#kriegel2008angle.

38. Pilnenskiy, N. & Smetannikov, I. Feature selection algorithms as one of the python data analytical tools. *Futur. Internet* **12**, (2020).

39. Kursa, M. B. & Rudnicki, W. R. Feature selection with the boruta package. *J. Stat. Softw.* **36**, 1–13 (2010).

40. Li, J. *et al.* Feature selection: A data perspective. *ACM Comput. Surv.* **50**, (2017).

41. Chen, T. & Guestrin, C. XGBoost: A scalable tree boosting system. *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.* **13**-**17**-**Augu**, 785–794 (2016).

42. Chou, K. C. & Shen, H. Bin. MemType-2L: A Web server for predicting membrane proteins and their types by incorporating evolution information through Pse-PSSM. *Biochem. Biophys. Res. Commun.* **360**, 339–345 (2007).

43. Zahiri, J., Yaghoubi, O., Mohammad-Noori, M., Ebrahimpour, R. & Masoudi-Nejad, A. PPIevo: Protein-protein interaction prediction from PSSM based evolutionary information. *Genomics* **102**, 237–242 (2013).

44. sklearn.neighbors.NeighborhoodComponentsAnalysis — scikit-learn 0.23.1 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NeighborhoodComponentsAnalysis.html.

45. Huang, H. *et al.* Panoramic view of a superfamily of phosphatases through substrate profiling. *Proc. Natl. Acad. Sci. U. S. A.* **112**, E1974–E1983 (2015).