# Specificity Prediction of Plant Acyl-ACP Thioesterases using Machine Learning

Deepro Banerjee\*, Michael Jindra†, Brian F. Pfleger†, Costas D. Maranas\*

\* Department of Chemical Engineering, Pennsylvania State University, University Park, Pennsylvania, USA

† Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, Wisconsin, USA

## Introduction

A key value proposition of synthetic biology is providing access to chemicals which are not sustainably produced at commercial scales. Medium-chain oleochemicals, with eight to twelve-carbon free fatty acids and derivatives, are one such class of products. While these chain lengths have traditionally been sourced from the tropical crops, such as palm, palm kernel, and coconut, the 8, 10, and 12-carbon products are not major constituents of the produced oil [1]. Furthermore, the displacement of rainforest habitat due to the cultivation of the oil palm has been identified as having the single largest impact on decreasing biodiversity observed in the Southeast Asian jungle ecosystem [2]. Processes have been established to create the higher value oleochemical derivatives, such as fatty alcohols, directly from petrochemical building blocks. However, these processes yield a distribution of alcohols\*\*fatty alcohols or acids?\*\*, and thus do not provide a highly selective route to the medium-chain products [3].

As an alternative, the field of synthetic biology has achieved fatty acid and fatty alcohol distributions with over 90% of the product belonging to the C8 species [4], [5]. This has been achieved via rewiring of the fatty acid biosynthesis pathway in *E. coli*, namely by the incorporation of an engineered 8-carbon specific acyl-ACP Thioesterase (TE) from *Cuphea palustris*. Indeed, the expression of various acyl-ACP TEs, either homologs from nature or variants thereof, has enabled control over the chain-length distribution in *E. coli* production systems [4], [5], [6], [7], [8], [9] **(Figure 1)**. Of these studies, acyl-ACP TEs from select plant species have been shown to have greater native specificity toward the medium-chain substrates when compared to bacterial homologs [7], [10], [11], [12]. Thus, several efforts have been made to bioprospect genomes of plants with high fractions of the medium-chain oils to identify and implement the TE gene responsible for the narrow substrate specificity [13], [4]. While progress has been demonstrated in identifying the features which dictate specificity in acyl-ACP TEs among plants [14], the throughput for bioprospecting, characterizing, and in some cases, engineering the acyl-ACP TE is largely inhibited by the testing pipeline, which requires derivatization of the free fatty acids into fatty acid methyl esters prior to analysis with gas chromatography [15]. An *in-silico* method for inferring substrate specificity from TE primary sequence would therefore expedite this process, by homing in and expressing only the most promising homologs in a host to gain insight to its selectivity profile.

A variety of *in-silico* approaches to infer enzyme functional groups, including substrate specificity \cite, based on their primary sequences have been proposed, which broadly fall under two categories, generative and discriminative. The generative approach builds a model of the feature distribution for each protein category and assigns a particular class or functional group to a candidate protein sequence by evaluating how well the sequence fits the model. Methods that fall under the generative category are based on sequence similarity comparison using local alignment similarity scores [16], [17], position specific scoring tables or profiles created from a group of previously aligned sequences [18], [19], consensus sequence patterns or motifs [20] and Hidden Markov Models (HMMs) [21]–[23]. However, generative approaches are either highly dependent on the database used to search for sequence similarity (local alignment and profile based similarity search) [24] or are computationally expensive (HMMs) [25]. Moreover, Jing *et al.* showed that phylogenetic and sequence identity analysis alone were not sufficient to distinguish plant TE substrate specificity [8]. The discriminative approach on the other hand focuses on accurately learning the decision boundary between classes. Commonly used discriminative approaches rely on training Machine Learning (ML) classifiers like Support Vector Machine (SVMs) or Neural Network (NNs) to learn discriminative rules from both positive (belonging to a particular protein class) and negative (not belonging to that protein class) set of protein sequences and using the learnt rules to predict the class of any new protein sequence [26]–[28]. ML algorithms like SVMs and NNs are relatively computationally inexpensive methods that have been used before to classify proteins into different functional classes with high accuracy [25], [27]–[35]. While some of these approaches have also incorporated structural information [36], [37], most ML algorithms have utilized only information from primary sequence to distinguish between highly similar proteins (e.g., guanylyl vs. adenylyl cylases, lactate vs. malate dehydrogenases, or trypsins vs. chymotrypsins [38]).

Diagram

Description automatically generated

**Figure 1:** The acyl-ACP Thioesterase plays a key role in fatty acid biosynthesis in E. coli. By intercepting the growing acyl-ACP chains, the Thioesterase hydrolyzes the acyl chain from the ACP and redirects flux to the free fatty acid pool. These free fatty acids can be further derivatized in vivo or ex vivo.

Recent results suggest that discriminative approaches relying on ML have outperformed generative approaches both in terms of accuracy and computational efficiency of solving the protein classification problem [30]. SVM is among the most widely used discriminative learning algorithm which has been proven to be extremely effective\*\*vague, quantify!\*\* on sequence based classification [26]–[28], [30], [33], [39]–[41]. SVM has been also used to detect remote protein homologies and classify proteins in the SCOP database into major structural classes [25], [26]. Ultimately, the performance of an SVM classifier is highly influenced by the feature extraction technique employed to encode the protein sequences [42].

Feature extraction of protein sequences generates a discrete numerical representation of a protein to create feature vectors that are correlated with the desired attribute of the protein one would like to predict. In order to train an SVM, a number of feature extraction techniques for protein sequences have been suggested in literature which can be divided into two categories, kernel based methods and vector representation of protein sequences [34]. The Fisher kernel introduced by Jakkola et. al. is one of the first kernel based feature extraction technique used to classify proteins based on their sequence information [43]. It was followed by the spectrum kernel [27] and its more generalized form, the mismatch kernel [30], both introduced by Leslie et. al. which achieved similar performances in terms of accuracy when compared to the Fisher kernel but with less computational expense. The weighted degree kernel introduced by Ratsch et. al. encoded the position of the substrings within the protein sequence as opposed to the spectrum and mismatch kernel introduced by Leslie et. al. [44]. It was used to identify alternatively spliced exons in C. elegans. Apart from kernel based methods, the second class of feature representation technique extracts structural and physicochemical properties embedded in the protein sequence and converts it into a numerical vector. One of the first and simplest discrete model to represent protein sequences that falls under the second class is Amino Acid Composition (AAC) developed by Nakashima et. al. which was used to classify proteins into different folding types with high accuracy [45]. Dubchak et. al. developed the more complicated Composition-Transition-Distribution (CTD) descriptor that takes into account different physical and stereochemical properties of the amino acids in the protein sequences such as amino acid composition, predicted secondary structure, hydrophobicity, normalized van der waals volume, polarity and polarizability to construct the feature vector [26]. Chou et. al. upgraded the simple AAC encoder developed by Nakashima et. al. to a pseudo Amino Acid (pseAA) composition encoder that is able to retain some pattern specific information embedded in the protein sequence [46]. Features have also been derived from Position Specific Scoring Matrices (PSSMs) profiles, which contains evolutionary information\*\*how?\*\* [47]. While selection of the most informative feature extraction technique has resulted in an improved performance of a classifier, the use of ensemble methods which combines the output of multiple classifiers has also helped to attain greater accuracy while solving the classification problem [48].

Several studies have shown that ensemble methods performed better than any individual classification method especially in problems relevant to the protein classification domain [29], [31], [32]. Camoglu et. al used a decision tree based ensemble classifier to classify protein in the SCOP database and showed how it is possible to attain much lower error rates using the ensemble classifier than any individual method [32]. Diplaris et. al. performed an empirical study where they compared the performance of several individual algorithms to solve the motif based classification problem and demonstrated the positive effect of combining different classification algorithms on prediction accuracy [31]. Tan et. al. illustrated the advantage of using ensemble classifiers on imbalanced datasets while solving the protein fold classification problem [29]. Similarly, Caragea et. al. trained an ensemble of SVM classifiers to predict glycosylation sites in amino acid residues and found that an ensemble of SVMs outperformed an individual SVM trained on imbalanced data [49].

Herein we put forth a machine learning based discriminatory approach to predict substrate specificity from their primary sequence for novel, uncharacterized, medium-chain TEs. To test the developed computational base, we trained an SVM-based ensemble classifier with TE sequences previously characterized in *E. coli*. Information about characterized TEs were collected as a part of this study from multiple literature sources [7], [50]–[52] and they were categorized into three different classes based on their substrate specificity as (i) medium chained, (ii) long chained and (iii) mixed, yielding a multi-class classification problem. To solve the classification problem, a stacked ensemble framework was developed comprised of three base learners trained using three different feature extraction techniques \*(i.e., x, y and z)\* from protein primary sequences. A meta learner was subsequently applied which combines the output of the three base learners by applying a majority voting criterion to predict the substrate specificity class of TEs. An ensemble method was used to circumvent the imbalanced dataset and retain high prediction accuracy. The achieved classification accuracy (i.e., XX%) validates our decision of using an ensemble method with imbalanced dataset improving over previous studies [29], [49]. The base learners in the ensemble are all SVM classifiers similar to the works of Caragea et. al. [49] and Nanni et. al. [34] where they have used an ensemble of SVMs to boost prediction accuracy. SVM was selected as the primary learning algorithm because they are generalizable and can handle high dimensional datasets [53]. We have also experimented with more complex/nonlinear models such as Neural Network and Gradient Boosted Trees as the base learner, but they were both outperformed by SVM in terms of accuracy on separate held-out validation sets. Proteins were represented by their amino acid sequences. The three feature extraction techniques used were (i) the spectrum kernel introduced by Leslie et. al. [27], (ii) a variant of the spectrum kernel where amino acids were first grouped into 6 categories to create a compressed representation based on their physicochemical attributes and the spectrum kernel was used to extract features from this compressed representation, and (iii) a newly introduced feature representation technique termed autopos detection which automatically detects the positions within the protein sequence which are maximally correlated with the functional attribute of the proteins. These three different feature representation techniques were used to train the three base models in the ensemble. We have also analyzed the efficacy of using 21 other representation techniques including other kernel based methods such as mismatch kernel and methods that extract physicochemical attributes from protein sequences such as CTD [26]. They all exhibited worse performance (detailed study given in the Results section). The feature vectors generated through the extraction process were decomposed into lower dimensional and linearly uncorrelated features using Principal Component Analysis. The reduction in dimensionality of the feature vectors was performed to prevent overfitting. The lower dimensional and decomposed set of feature vectors were used to train the classifiers and predict substrate specificity of TEs. The ensemble method achieved a mean validation accuracy of 0.76 across 10,000 simulations of this study using different training and validation sets. However, the worst case accuracy across simulations was 0.45 which indicates that the method is not extremely robust to the training set. One possible reason behind the lack of robustness could be that the decision boundaries between the protein classes is not well defined. SVMs are maximal margin classifiers and the thickness of the decision boundary plays a very important role governing the robustness of the model [54]. The mean precision score of the model across the simulations for the medium chained TEs, the product of interest, was 0.84. We used this model to identify 10-carbon specific acyl-ACP TE among a set of uncharacterized TE enzymes from select plants known to have predominantly decanoyl chains in their seed oils. This study provides an exemplar of how small to medium-sized datasets can be leveraged to guide bioprospecting efforts while simultaneously supporting the endeavor of the synthetic biology community to provide access to chemicals not easily obtained through conventional methods.

## Methods

### Dataset Compilation

The dataset was compiled by integrating primary sequence and substrate specificity of TEs from multiple literature sources [7], [50]–[52]. The dataset includes primary sequence and accompanying in vivo *E. coli* product distributions for 116 acyl-ACP plant TEs previously reported in scientific and patent literature [7], [50]–[52]. *E. coli* was chosen because it remains the most common and facile method for characterization of heterologous TEs. The product distribution data was subsequently used to classify each TE into discrete categories to be used in a classification framework. Based on their product distributions, the TEs were divided into three categories, 1) the “medium-chain” category contained TE which resulted in distributions of at least 50% C8 to C12 free fatty acids, 2) the “long-chain” category contained TE which produced 50% C14 to C18 free fatty acids and less than 10% C8 to C12 free fatty acids and 3) the “mixed distribution” category contained TE which yielded distributions between 10% and 50% C8 to C12 free fatty acids. This discrete classification into three bins is a useful abstraction but it does not account for the fact that there exists TEs with a continuum of specificities across all chain lengths.

### Feature Extraction

In this work, 47 alignment-free feature extraction techniques that encode primary sequence information of the enzymes into fixed length feature vectors were employed. The feature extraction techniques fall under four categories, Kernel methods, N-gram methods, Physicochemical encoding methods and PSSM profile based methods. The feature extraction category, name, software package used to deploy them and literature from which they are adopted is listed in Table Y. A brief description of the 47 feature extraction techniques divided into their respective categories is also presented in this section.

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Software Used | Literature Reference |
| Kernel | Spectrum Kernel | KeBABS | [27] |
| Kernel | Mismatch Kernel | KeBABS | [30] |
| Kernel | Gappy Pair Kernel | KeBABS | [55] |
| N-gram | Kmer | Numpy | [56] |
| N-gram | GAA-kmer | Numpy | NA |
| Physicochemical | AAC | iFeature | [57] |
| Physicochemical | CKSAAP | iFeature | [58] |
| Physicochemical | TPC | iFeature | [57] |
| Physicochemical | DPC | iFeature | [59] |
| Physicochemical | DDE | iFeature | [59] |
| Physicochemical | GAAC | iFeature | [60] |
| Physicochemical | CKSAAGP | iFeature | [61] |
| Physicochemical | GTPC | iFeature | [61] |
| Physicochemical | GDPC | iFeature | [61] |
| Physicochemical | Moran | iFeature | [62] |
| Physicochemical | Geary | iFeature | [63] |
| Physicochemical | NMBroto | iFeature | [64] |
| Physicochemical | CTDC | iFeature | [65]–[67] |
| Physicochemical | CTDT | iFeature | [65]–[67] |
| Physicochemical | CTDD | iFeature | [65]–[67] |
| Physicochemical | CTriad | iFeature | [68] |
| Physicochemical | KSCTriad | iFeature | [61] |
| Physicochemical | SOCNumber | iFeature | [61] |
| Physicochemical | QSOrder | iFeature | [61] |
| Physicochemical | PAAC | iFeature | [69] |
| Physicochemical | APAAC | iFeature | [69] |
| PSSM-based | AAC-PSSM | POSSUM | [70] |
| PSSM-based | DPC-PSSM | POSSUM | [70] |
| PSSM-based | AADP-PSSM | POSSUM | [70] |
| PSSM-based | PSSM-AC | POSSUM | [71] |
| PSSM-based | PSSM-CC | POSSUM | [72] |
| PSSM-based | RPSSM | POSSUM | [73] |
| PSSM-based | Tri-gram | POSSUM | [74] |
| PSSM-based | EDP | POSSUM | [75] |
| PSSM-based | EEDP | POSSUM | [75] |
| PSSM-based | MEDP | POSSUM | [75] |
| PSSM-based | TPC-PSSM | POSSUM | [76] |
| PSSM-based | AATP | POSSUM | [76] |
| PSSM-based | k-separated bigrams | POSSUM | [77] |
| PSSM-based | D-FPSSM | POSSUM | [78] |
| PSSM-based | S-FPSSM | POSSUM | [78] |
| PSSM-based | PSE-PSSM | POSSUM | [79] |
| PSSM-based | DP-PSSM | POSSUM | [80] |
| PSSM-based | PSSM-Composition | POSSUM | [81] |
| PSSM-based | Smoothed-PSSM | POSSUM | [82] |
| PSSM-based | AB-PSSM | POSSUM | [83] |
| PSSM-based | RPM-PSSM | POSSUM | [83] |

#### Kernel methods

Three kernel based methods, described in this section, were used in our ensemble model. They were all implemented using the KeBABS software package [84].

##### Spectrum kernel

The k-spectrum kernel proposed by Leslie et. al. [27] is the set of all k-length contiguous subsequences that can occur in a given input sequence. If we assume to be the set of all characters of a sequence, to be a k-mer and to be function that denotes the number of times a given k-mer is present in a protein sequence , then the feature map is as follows:

##### Mismatch kernel

The mismatch kernel is a generalized form of the spectrum kernel. The kernel has two parameters k and m, where k is the length of all possible k-mers that might be present in a protein sequence and m is the maximum number of mismatches allowed for a single k-mer. If we assume to be the set of all characters of a sequence, to be a k-mer, then for a fixed k-mer u, the () mismatch pattern is the set of all -length subsequences generated from that differ from by at most mismatches. The feature map of a fixed k-mer u can be defined as:

Where is 1 if belongs to the set of -length subsequences that differ from by at most mismatches, 0 otherwise. The feature map on an input sequence is defined as:

#### 

##### Gappy pair kernel

The gappy pair kernel looks for pairs of matching -mers separated by a fixed number of amino acids in between them. It has two parameters and , where k denotes the length of the -mers and denotes the maximum gap between the matching amino acid -mer pairs to consider.

#### N-gram methods

Two n-gram based methods, described below, were used in our ensemble model. They were all implemented using python’s numpy package [85].

##### k-mer motif builder

This representation is similar to the N-gram representation for language models [56]. Here, the entire training dataset (all enzyme sequences) is scanned to search for all possible sequence motifs of length k. A one hot encoded feature set is created with sequence motifs which are repeated in at least 2 instances of enzymes in the training data. This step is taken for two reasons, 1) To avoid building a large set of features and 2) It is expected that enzymes of a specific substrate specificity will have common motifs and the main aim of the model is to learn these common motifs, not a motif specific to a particular enzyme.

##### Grouped amino acid encoded k-mer motif builder

The grouped amino acid encoded k-mer motif builder performs an additional pre-processing step on the primary sequences of the enzymes before encoding them into a feature vector representation similar to the k-mer motif builder. In the preprocessing step, 20 amino acid types which can occur at a particular position in the sequence of an enzyme were categorized into five classes according to their physicochemical properties, hydrophobicity, charge and molecular size. The five classes are the aliphatic group (: GAVLMI), aromatic group (: FYW), positive charge group (: KRH), negative charged group (: DE) and uncharged group (: STCPNQ). Finally, each amino acid in the sequence was replaced by its respective class to create an encoded sequence with only five different types of values per position. For example, a primary sequence ‘MLTPWS’ is represented as . Using the encoded representation, k-mer motif builder was used to represent the protein sequence. It should be noted that the preprocessing step significantly reduces the set of characters of the sequence and can help prevent overfitting since the kernel feature map size will be much lower than the original feature map size obtained by the k-mer motif builder. The reduced feature map size will require much lower number of parameters to train a classifier and thus may prevent overfitting.

#### Physicochemical encodings

21 physicochemical encoding based methods, described below, were used in our ensemble model. They were all implemented using the iFeature software package [61].

##### Amino Acid Composition (AAC)

AAC encoding calculates the frequency of each of the 20 types of amino acids in a protein sequence and encodes them in a vector of length 20. Amino acid composition for a protein sequence x can be defined as:

Where is the count of an amino acid , is the length of the protein sequence and is the set of all amino acid types.

##### Composition of k-spaced Amino Acid Pairs (CKSAAP)

CKSAAP encoding calculates the frequency of amino acid pairs separated by at most k residues. For example, for k=2, the feature vector can be defined as:

Where can be any of the 20 different amino acid types and represents the total number of 2-spaced amino acid pairs present in a protein sequence.

##### Tri-Peptide Composition (TPC)

TPC calculates the frequency of the tripeptides in a protein sequence. The TPC encoding of a protein x can be represented as:

Where is the number of tripeptides represented by amino acid types , and that belongs to the set of all amino acids .

##### Di-Peptide Composition (DPC)

DPC calculates the frequency of the dipeptides in a protein sequence. The DPC encoding of a protein x can be represented as:

Where is the number of dipeptides represented by amino acid types and that belongs to the set of all amino acids .

##### Dipeptide Deviation from Expected Mean (DDE)

DDE feature vector is a function of three variables, dipeptide composition (), theoretical mean () and theoretical variance () to calculate the final feature encoding. is defined in the Di-Peptide Composition section. is calculated as:

Where is the number of codons that code for amino acid , is the number of codons that code for amino acid and is the total number of possible codons excluding the stop codons. is calculated as:

Where is the length of the protein peptide. The is calculated as:

##### Grouped Amino Acid Composition (GAAC)

GAAC encoding performs an additional preprocessing step of categorizing the amino acids into five classes according to their physicochemical properties before computing the composition of the five groups similar to [AAC](#_Amino_Acid_Composition). The five classes are the aliphatic group (: GAVLMI), aromatic group (: FYW), positive charge group (: KRH), negative charged group (: DE) and uncharged group (: STCPNQ).

##### Composition of k-spaced Amino Acid Group Pairs (CKSAAGP)

The CKSAAGP encoding performs an additional preprocessing step of categorizing the amino acids into five classes according to their physicochemical properties before computing the frequency of the grouped amino acid pairs separated by at most residues similar to [CKSAAP](#_Composition_of_k-spaced). The five classes are the aliphatic group (: GAVLMI), aromatic group (: FYW), positive charge group (: KRH), negative charged group (: DE) and uncharged group (: STCPNQ).

##### Grouped Di-Peptide Composition (GDPC)

The GDPC encoding performs an additional preprocessing step of categorizing the amino acids into five classes according to their physicochemical properties before computing the dipeptide composition of the amino acid groups similar to [DPC](#_Di-Peptide_Composition_(DPC)). The five classes are the aliphatic group (: GAVLMI), aromatic group (: FYW), positive charge group (: KRH), negative charged group (: DE) and uncharged group (: STCPNQ).

##### Grouped Tri-Peptide Composition (GTPC)

The GTPC encoding performs an additional preprocessing step of categorizing the amino acids into five classes according to their physicochemical properties before computing the tripeptide composition of the amino acid groups similar to [TPC](#_Tri-Peptide_Composition_(TPC)). The five classes are the aliphatic group (: GAVLMI), aromatic group (: FYW), positive charge group (: KRH), negative charged group (: DE) and uncharged group (: STCPNQ).

##### Moran Correlation (Moran)

Moran descriptor is computed based on the numerical values representing the biological and physicochemical attributes of different types of amino acids given in the AAindex database [86]. The descriptor is defined as

Where is the lag in correlation, is the maximum value of the lag, and are the properties of amino acids at positions and as obtained from the AAindex database and is the average value of the property P across the entire protein sequence.

##### Geary Correlation (Geary)

Geary descriptor is computed based on the numerical values representing the biological and physicochemical attributes of different types of amino acids given in the AAindex database. The descriptor is defined as

Where is the lag in correlation, is the maximum value of the lag, and are the properties of amino acids at positions and as obtained from the AAindex database and is the average value of the property across the entire protein sequence.

##### Normalized Moreau-Broto Autocorrelation (NMBroto)

NMBroto descriptor is defined as

Where is the length of the protein sequence, is the lag in autocorrelation, is the maximum value of lag to be considered and is the Moreau-Broto autocorrelation descriptor calculated as:

Here, and are the properties of amino acids at positions and as obtained from the AAindex database.

##### Composition Transition Distribution – Composition (CTDC)

CTDC feature encoding transforms the amino acid sequence of a protein into three subgroups of 13 types of physicochemical properties including hydrophobicity, normalized Van der Waals Volume, polarity, polarizability, charge, secondary structures and solvent accessibility. After dividing the sequence into subgroups of a specific physicochemical property, the frequencies of these subgroups are calculated, and the feature descriptor is described as follows:

Where denotes the subgroup type, denotes the set of all subgroups and denotes the physicochemical property.

##### Composition Transition Distribution – Transition (CTDT)

CTDC feature encoding transforms the amino acid sequence of a protein into three subgroups of 13 types of physicochemical properties including hydrophobicity, normalized Van der Waals volume, polarity, polarizability, charge, secondary structures and solvent accessibility. After dividing the sequence into subgroups of a specific physicochemical property, the frequencies of the transition from one subgroup to another are calculated, and the feature descriptor is described as follows:

Where and denote the subgroup type, denotes the set of all subgroups and denotes the physicochemical property.

##### Composition Transition Distribution – Distribution (CTDD)

CTDD feature encoding transforms the amino acid sequence of a protein into three subgroups of 13 types of physicochemical properties including hydrophobicity, normalized Van der Waals volume, polarity, polarizability, charge, secondary structures and solvent accessibility. After dividing the sequence into subgroups of a specific physicochemical property, the fraction of the entire sequence where the first residue, 25%, 50%, 75% and 100% of residues of any given subgroup is marked. The process is followed for all physicochemical properties and the numerical fractions are combined in a vector to represent the feature descriptor.

##### Conjoint Triad (CTriad)

CTriad considers three contiguous amino acids as a single unit. It creates a vector space where each element in the vector represents a triad type and another vector space that is the frequency vector corresponding to where each element denotes the number of triad type appearing in the protein sequence. Since the values of are directly correlated to the length of the protein, the numerical vector is normalized using Min-Max scaling technique.

##### k-spaced Conjoint Triad (KSCTriad)

KSCTriad not only takes into account contiguous amino acids to create the feature space as in CTriad but also considers amino acids that are separated by at most residues as a single unit.

##### Sequence-Order-Coupling Number (SOCNumber)

SOCNumber is calculated from the distance matrices proposed by Schneider-Wrede and Grantham. It is defined as:

Where is the entry in a distance matrix that denotes the distance between the two amino acids at position and , is the lag or the positional difference between two amino acid residues, is the maximum value of the lag to be considered and is the length of the protein sequence.

##### Quasi-sequence-order (QSOrder)

QSOrder is based on SOCNumber and is defined as:

Where is the normalized count of amino acid , is a weighing factor, is the SOCNumber, is an amino acid type and is the set of all amino acids. The above definition is valid for up to 20. For above 20, the definition is:

##### Pseudo-Amino Acid Composition (PAAC)

PAAC is defined as:

Where is the normalized count of amino acid , is a weighing factor and is a sequence order correlated factor defined as:

Where is the length of the protein sequence, is an integer parameter to be chosen that must be less than , and is defined as:

Where is the -th property in the amino acid property set for amino acid . The amino acid properties are normalized hydrophobicity values, normalized hydrophilicity values and normalized side chain masses of the 20 amino acids.

##### Amphiphilic Pseudo-Amino Acid Composition (APAAC)

APAAC is similar to PAAC and is defined as:

Where is the normalized count of amino acid , is a weighing factor and is a sequence order correlated factor defined as:

Where is the length of the protein sequence, is an integer parameter to be chosen that must be less than , and is defined as:

Where is the normalized hydrophobicity value for amino acid and is the normalized hydrophilicity value of amino acid .

#### PSSM methods

21 feature extraction techniques were derived from the PSSM profiles of TE sequences. PSSM profile of a sequence is constructed from multiple sequence alignment of the highest scoring hits in a BLAST search. PSI-BLAST program [17] was used to search NCBI’s UniRef50 database [87] with 5 iterations and cut-off E-value of 0.001 and create the PSSM profile for the TE sequences. The PSSM profile of an enzyme is an matrix where is the length of the enzyme sequence and 20 represents the number of amino acids that might be present at a specific position in a sequence. The -th element of the PSSM profile represents the score of the amino acid residue in the -th position of the enzyme sequence being mutated to amino acid type during the evolution process wherein a large value indicates a highly conserved position, and a small value indicates a weakly conserved position. The PSSM elements are usually scaled to a (0,1) range using the sigmoid function ,

Where x is an element in the PSSM profile.

##### Amino Acid Composition using PSSM profiles (AAC-PSSM)

AAC-PSSM uses the numerical values of the PSSM matrix instead of the actual count of the amino acids as in traditional [AAC](#_Amino_Acid_Composition) to calculate composition of each amino acid. The PSSM profile of a protein sequence obtained using PSI-BLAST can be denoted as a matrix of length where is the length of the protein sequence. Thus, AAC-PSSM feature vector can be represented as:

Where is an element of the PSSM matrix corresponding to the -th position of the sequence and the -th amino acid type.

##### Di-Peptide Composition using PSSM profiles (DPC-PSSM)

DPC-PSSM uses the numerical values of the PSSM matrix instead of the actual count of di-peptides as in traditional DPC to calculate frequency of each dipeptide. The PSSM profile of a protein sequence obtained using PSI-BLAST can be denoted as a matrix of length L\*20 where L is the length of the protein sequence. Thus, DPC-PSSM feature vector can be represented as:

Where is an element of the PSSM matrix corresponding to the -th position of the sequence and the -th amino acid type.

##### Amino Acid and Di-Peptide composition using PSSM profiles (AADP-PSSM)

AADP-PSSM is a concatenation of the feature vectors obtained using [AAC-PSSM](#_Amino_Acid_Composition_1) and [DPC-PSSM](#_Di-Peptide_Composition_using).

##### PSSM Auto-Covariance (PSSM-AC)

PSSM-AC applies auto-covariance transformation to each column of the PSSM matrix to measure the average correlation of the same property between residues separated by a distance of at most positions. For a protein , it can be denoted as:

Where is the length of the protein sequence, is an element of the PSSM matrix corresponding to the -th position of the sequence and the -th amino acid type, is the maximum distance between residues to consider and is pssm profile composition of amino acid throughout the length of the protein sequence as described in [AAC-PSSM](#_Amino_Acid_Composition_1).

##### PSSM Cross-Covariance (PSSM-CC)

PSSM-CC applies cross-covariance transformation to each column of the PSSM matrix to measure the average correlation of different properties between residues separated by a distance of at most D positions. For a protein , it can be denoted as:

Where is the length of the protein sequence, is an element of the PSSM matrix corresponding to the -th position of the sequence and the -th amino acid type, is the maximum distance between residues to consider and () is pssm profile composition of amino acid () throughout the length of the protein sequence as described in [AAC-PSSM](#_Amino_Acid_Composition_1). It should be noted that unlike [PSSM-AC](#_PSSM_Auto-Covariance_(PSSM-AC)), and are two different amino acid types.

##### Reduced PSSM (RPSSM)

RPSSM works with a reduced PSSM by grouping the 20 amino acids into 10 categories and compressing the PSSM into an matrix. At first, the elements in the PSSM matrix are mapped to the (0,1) range using sigmoid function. The grouping procedure is followed according to the rules of Li. et. al. \cite which is described as follows:

Where represents the new columns of the PSSM matrix and represents the columns in the PSSM matrix that correspond to amino acid . After creating a reduced PSSM profile, the pseudo composition of the amino acids and all the dipeptides in the protein sequence is calculated from the reduced profile as follows:

Where is the pseudo composition of amino acids, is an element of the reduced pssm matrix that corresponds to location in a sequence of length and grouped amino acid type , is the amino acid composition obtained from the reduced PSSM profile as defined in AAC and is the pseudo amino acid composition of all the dipeptides present in the protein sequence. The final feature vector is a concatenation of the two vectors and .

##### Tri-gram PSSM (tri-gram)

Tri-gram calculates the probabilities of the occurrence of a certain combination of amino acid triplets using the PSSM profile of the protein. At first, the elements in the PSSM matrix are mapped to the (0,1) range using sigmoid function. Then the feature vector is calculated as follows:

Where is an element of the pssm matrix that corresponds to location in a sequence of length and amino acid type .

##### EDP

EDP feature vectors are obtained from an Evolutionary Difference Matrix created by transforming the PSSM matrix into a dimensional ED-PSSM matrix where each element of the matrix represents a mutational difference between adjacent residues. To create the evolutionary difference matrix, the average of evolutionary score between adjacent residues are calculated as follows:

Where is the average evolutionary score between positions and , is the average evolutionary score between and and is an element of the PSSM matrix where denotes a position along the length of the protein sequence and denotes an amino acid. The evolutionary difference formula is subsequently calculated as follows:

The evolutionary difference formula is used to convert the PSSM matrix into ED-PSSM where an element of the ED-PSSM matrix is calculated as:

Where is an element of the ED-PSSM matrix and is the length of the protein sequence. The ED-PSSM matrix is used to calculate EDP feature vector as follows:

##### EEDP

The EEDP feature vector is also constructed from the ED-PSSM matrix described in [EDP](#_EDP). The feature vector is as follows:

Where is an element of the ED-PSSM matrix.

##### MEDP

MEDP feature vector is the concatenation of two feature vectors [EDP](#_EDP) and [EEDP](#_EEDP).

##### TPC-PSSM

TPC feature representation is obtained from the PSSM matrix as follows:

Where is an element of the PSSM matrix and is the length of the protein sequence.

##### AATP

AATP is the concatenation of two feature vectors [AAC-PSSM](#_Amino_Acid_Composition_1) and [TPC-PSSM](#_TPC-PSSM).

##### k-separated-bigrams-PSSM

k-separated-bigrams model the relationship between non-adjacent amino acids separated by a distance of k-positions using the sequential evolution probabilities from the PSSM matrix of a protein sequence. It can be calculated as follows:

Where is an element of the pssm matrix and is the length of the protein sequence.

##### D-FPSSM

D-FPSSM is calculated from the FPSSM matrix obtained after filtering out all the negative scores from the PSSM matrix. After pre-processing, D-FPSSM is calculated as follows:

Where is an element of the FPSSM matrix.

##### S-FPSSM

S-FPSSM is also calculated from the FPSSM matrix described in [D-FPSSM](#_D-FPSSM) section as follows:

Where is an element in the FPSSM matrix that represents the -th position along the sequence length and -th amino acid and is an indicator function defined as:

##### PSE-PSSM

PSE-PSSM concatenates [AAC-PSSM](#_Amino_Acid_Composition_1) with a new set of descriptors adopted from pseudo-PSSM composition which can be calculated as:

Where is an element of the PSSM matrix that represents the score of the amino acid residue in -th position of the protein sequence being changed to amino acid type during the evolution process, is the length of the protein sequence and is the maximum distance between two residues whose PSSM scores are coupled.

##### DP-PSSM

DP-PSSM uses the PSSM profile to construct a feature vector as follows:

Where is an element of the PSSM matrix that represents the score of the amino acid residue in -th position of the protein sequence being changed to amino acid type during the evolution process, () is the number of positive (negative) terms in the column of the PSSM matrix, () is the number of positive (negative) terms obtained after calculating the difference where k denotes the spaces between amino acid residue positions, ranges from 1 to , being the length of the protein sequence under consideration and is the amino acid type to be considered.

##### PSSM-Composition

The PSSM-Composition feature vector is calculated as:

Where is the amino acid residue at position of the sequence and is an element in the PSSM matrix.

##### Smoothed-PSSM

Smoothed-PSSM preprocesses the PSSM profile by incorporating information about the neighboring residues using a smoothing window size of w. Here, each row vector of a residue is represented as follows:

Where is an element of the smoothed pssm profile and is an element of the original pssm profile. After preprocessing, the feature vector of a protein is created by selecting residues based on a sliding window of size .

##### AB-PSSM

AB-PSSM constructs a feature set from PSSM matrix by dividing the matrix into 20 blocks where each block has number of rows equivalent to 5% of the length of protein sequence under consideration and 20 columns corresponding to the 20 types of amino acids. The feature vector is constructed as follows:

Where is the size of the block, is a vector of size 20 extracted from the PSSM profile corresponding to the -th row in the -th block.

##### RPM-PSSM

The RPM-PSSM vector is calculated as:

Where is the amino acid residue at position of the sequence and is an element in the PSSM matrix and is the number of amino acids of type in column of the PSSM matrix.

### Ensemble Method for TE Classification

#### Model Description

Our ensemble framework comprises of 47 base learners which provide their outputs to a meta learner that predicts the enzyme specificity class. Although all of our base learners are trained using the same principle (SVM or NN or GBT), the heterogeneity among them is governed by the 47 different feature extraction techniques (described in the [Feature Extraction](#_Feature_Extraction) section) used to encode the set of TE sequences. Each of the 47 different feature extraction techniques creates a unique feature vector representation of TEs which serve as input to a designated base learner. The base learner trained on the set of encoded TE sequences yields the predicted substrate specificity of a given TE sequence as an output. The outputs of the -best base learners are passed on to the meta learner that uses a majority voting scheme (described in [The Meta Learner](#_The_Meta_Learner) section) to predict the enzyme specificity category. The -best base learners are selected on the basis of their performance on validation set as described in [Model Training](#_Model_Training_1) section. The workflow of our ensemble model is presented in **Figure 2**.

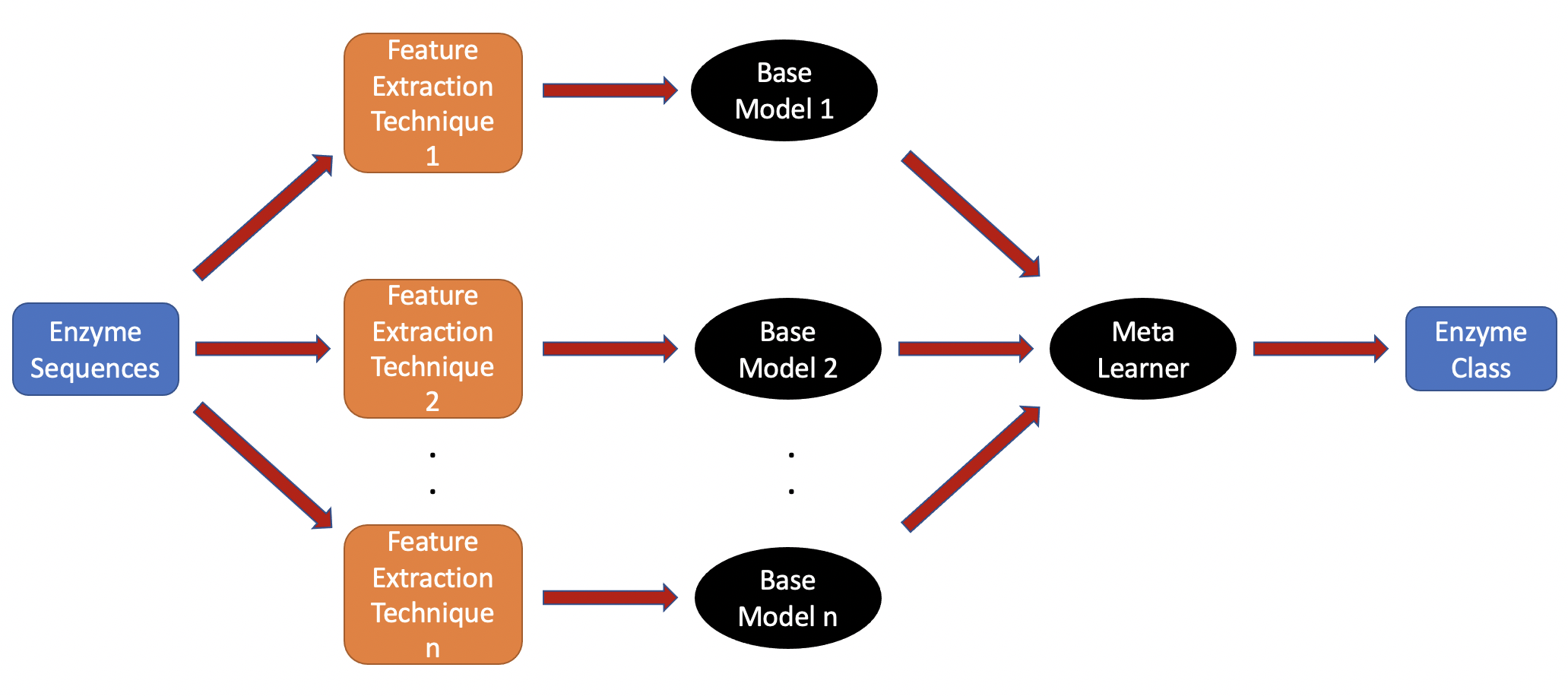
#### The Base Learners

##### Support Vector Machine

The Support Vector Machine based learner of enzyme specificity prediction included Principal Component Analysis (PCA) for dimensionality reduction of the feature space followed by a Support Vector Classifier [88] to predict enzyme specificity class. The PCA based dimensionality reduction step was carried out to decrease the number of parameters required to train an SVM model and make the model more generalizable. The one versus one strategy was used to adapt the SVM for multi-class classification [89]. The number of PCA components, SVM model kernel, regularization parameter C and kernel coefficient gamma were selected by optimizing these hyperparameters using a 10-fold cross validation scheme (described in [Model Training](#_Model_Training) section).

##### Neural Network

The Neural Network based learner of enzyme specificity prediction included Principal Component Analysis for dimensionality reduction of the feature space followed by an Artificial Neural Network classifier to predict enzyme specificity class. The PCA based dimensionality reduction step was carried out to decrease the number of parameters required to train an NN model and make the model more generalizable. The number of PCA components, hidden layer size of NN, initial learning rate and L2 regularization parameter alpha were selected by optimizing these hyperparameters using a 10-fold cross validation scheme (described in [Model Training](#_Model_Training) section).



**Figure 2:** Workflow of the ensemble model. Different feature representations create separate models, and the final model output is the majority vote of the predictions made by each individual model.

##### Gradient Boosting Trees

The Gradient Boosting Tree based learner of enzyme specificity prediction included Principal Component Analysis for dimensionality reduction of the feature space followed by a Gradient Boosting Tree classifier to predict enzyme specificity class. The PCA based dimensionality reduction step was carried out to decrease the number of parameters required to train an NN model and make the model more generalizable. The number of PCA components, number of estimators or decision trees to consider, learning rate and maximum depth of the trees were selected by optimizing these hyperparameters using a 10-fold cross validation scheme (described in [Model Training](#_Model_Training_1) section).

#### The Meta Learner

The meta learner accepts the outputs of all the base learners as an input vector, implements a hard majority voting scheme on the input vector and returns the consensus prediction of TE substrate specificity as an output. The meta learner output is calculated as follows:

Where is the output of the meta learner, is the output of the base learner , is the number of best performing base learners to consider and is a function that selects the most frequent value among a set of values.

#### Model Hyperparameters

The model has several hyperparameters which can be tuned to improve model performance. The ensemble model has a hyperparameter that denotes the number of best performing base learners that the meta learner needs to take into account. Each base learner has a number of hyperparameters depending on the learning algorithm used to train them. The respective learning algorithm dependent hyperparameters of the base learners are shown in Table X. The hyperparameters are learnt through a 10-fold cross validation scheme discussed in [Model Training](#_Model_Training_1) section.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Learning Algorithm | Hyperparameter I | Hyperparameter II | Hyperparameter III | Hyperparameter IV |
| SVM | PCA Components | Kernel | Regularization C | Gamma |
| NN | PCA Components | Hidden layer size | Learning rate | Regularization |
| GBC | PCA Components | Number of estimators | Learning rate | Tree max depth |

#### Model Training

The model was trained using python’s numpy and scikit-learn modules [85], [90]. Model training can be divided into five stages.

##### Random seed assignment and dataset division

At first, a random seed was specified using numpy to reproduce results. The dataset of 116 TE enzyme sequences labeled according to their corresponding substrate specificity category was divided into a training set and a validation set by a 75-25 percentage split. Setting the random seed allows us to reproduce results. Additionally, changing the random seed will produce different training and validation sets, an event which will be used later to evaluate model performance.

##### Feature Representation

The training and validation set of sequences were encoded by the 47 different feature representation techniques, described in the [Feature Extraction](#_Feature_Extraction) section, into 47 distinct feature vector representation of the sequences. The distinct feature vectors of the training set of sequences were used to train 47 separate base learners operating on the same principle (PCA+SVM or PCA+NN or PCA+GBT).

##### Base Model Training

The base models accept the feature vector representation of TE sequences as input and predicts TE substrate specificity category. They were trained using scikit learns dedicated modules for PCA, SVC, NN and GBC. A scikit-learn pipeline object consisting of PCA instance followed by the learning algorithm instance (SVC, NN or GBC) was created. The pipeline object was trained on the feature vector representation of the training set of TE sequences. The hyperparameters of the base learners were optimized using the GridSearchCV module of scikit-learn and setting the cross validation split to 10.

##### Model Evaluation

The 47 base models trained only on the training set were used to independently predict the substrate specificity category of enzymes in both training and validation sets. The output predictions of these base learners were passed on to the meta learner that used a hard-voting based majority vote classifier to output the final prediction of the enzyme substrate specificity class. The parameter of the ensemble model , that represents the base models to pass on to the meta learner was chosen to be 17. The training and validation accuracies and precision score (on the medium chained TEs) of the 47 base learners and the ensemble model were recorded. This whole procedure was repeated 10,000 times by varying the random seed specified initially, which resulted in different training and validation set, thus affecting the model performance and yielding a distribution of training and validation set accuracies for the three base learners and the ensemble model. The objective of training our model multiple times was to check its robustness to the training set. A parametric sweep of the ensemble model parameter was performed to illustrate its effect on validation score.

##### Model Prediction

To predict an uncharacterized TE sequence from the test set, at first, the test sequences were converted to feature vectors by the 47 different feature extraction techniques. Then, 47 base models delegated to each feature vector representation were trained and hyperparameter optimized using all the characterized TE sequences from both training and validation sets. The trained base models were used to independently predict the substrate specificity category of test set enzymes. The output predictions of the best performing base learners were passed on to a meta learner that used a hard-voting based majority vote classifier to output the final prediction of the enzyme substrate specificity class. The parameter k was selected based on the results of the parametric sweep study discussed in the [Model Evaluation](#_Model_Evaluation) section.

## References

[1] W. Rupilius and S. Ahmad, “Palm oil and palm kernel oil as raw materials for basic oleochemicals and biodiesel,” *Eur. J. Lipid Sci. Technol.*, vol. 109, no. 4, pp. 433–439, 2007, doi: 10.1002/ejlt.200600291.

[2] D. S. Wilcove and L. Pin, “Addressing the threats to biodiversity from oil-palm agriculture,” *Biodivers. Conserv.*, vol. 19, no. 4, pp. 999–1007, 2010, doi: 10.1007/s10531-009-9760-x.

[3] K. Noweck and H. Ridder, “Fatty Alcohols - Industrial Production,” in *Ullmann’s encyclopedia of industrial chemistry*, 5th ed., Wiley-VCH, 1988, pp. 277–295.

[4] N. J. Hernández Lozada *et al.*, “Highly Active C 8 -Acyl-ACP Thioesterase Variant Isolated by a Synthetic Selection Strategy,” *ACS Synth. Biol.*, vol. 7, no. 9, pp. 2205–2215, 2018, doi: 10.1021/acssynbio.8b00215.

[5] N. J. Hernández Lozada, T. R. Simmons, K. Xu, M. A. Jindra, and B. F. Pfleger, “Production of 1-octanol in Escherichia coli by a high flux thioesterase route,” *Metab. Eng.*, vol. 61, no. April, pp. 352–359, 2020, doi: 10.1016/j.ymben.2020.07.004.

[6] M. J. Grisewood *et al.*, “Computational Redesign of Acyl-ACP Thioesterase with Improved Selectivity toward Medium-Chain-Length Fatty Acids,” doi: 10.1021/acscatal.7b00408.

[7] L. Yuan, T. A. Voelker, and D. J. Hawkins, “Modification of the substrate specificity of an acyl-acyl carrier protein thioesterase by protein engineering,” *Proc. Natl. Acad. Sci. U. S. A.*, 1995, doi: 10.1073/pnas.92.23.10639.

[8] F. Jing *et al.*, “Phylogenetic and experimental characterization of an acyl-ACP thioesterase family reveals significant diversity in enzymatic specificity and activity,” pp. 1–16, 2011.

[9] X. Deng, L. Chen, M. Hei, T. Liu, Y. Feng, and G. Y. Yang, “Structure-guided reshaping of the acyl binding pocket of ‘TesA thioesterase enhances octanoic acid production in E. coli,” *Metab. Eng.*, vol. 61, no. January, pp. 24–32, 2020, doi: 10.1016/j.ymben.2020.04.010.

[10] T. A. Voelker and H. M. Davies, “Alteration of the Specificity and Regulation of Fatty Acid Synthesis of Escherichia coli by Expression of a Plant Medium- Chain Acyl-Acyl Carrier Protein Thioesterase,” *J. Bacteriol.*, vol. 176, no. 23, pp. 7320–7327, 1994.

[11] Y. J. Choi and S. Y. Lee, “Microbial production of short-chain alkanes,” *Nature*, vol. 502, no. 7472, pp. 571–574, 2013, doi: 10.1038/nature12536.

[12] S. Sarria, T. G. Bartholow, A. Verga, M. D. Burkart, and P. Peralta-Yahya, “Matching Protein Interfaces for Improved Medium-Chain Fatty Acid Production,” *ACS Synth. Biol.*, vol. 7, no. 5, pp. 1179–1187, 2018, doi: 10.1021/acssynbio.7b00334.

[13] P. Gordon Roessler and G. Roy, “ACYL-ACP THOESTERASE GENES AND USES THEREFOR,” 8956834 B2, 2015.

[14] F. Jing, L. Zhao, M. D. Yandeau-Nelson, and B. J. Nikolau, “Two distinct domains contribute to the substrate acyl chain length selectivity of plant acyl-ACP thioesterase,” *Nat. Commun.*, vol. 9, no. 1, p. 860, 2018, doi: 10.1038/s41467-018-03310-z.

[15] M. Politz, R. Lennen, B. Pfleger, and B. Engineering, “Quantification of Bacterial Fatty Acids by Extraction and Methylation,” *Bio Protoc.*, vol. 3, no. 21, 2016.

[16] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, “Basic local alignment search tool,” *J. Mol. Biol.*, 1990, doi: 10.1016/S0022-2836(05)80360-2.

[17] S. F. Altschul *et al.*, “Gapped BLAST and PSI-BLAST: A new generation of protein database search programs,” *Nucleic Acids Research*. 1997, doi: 10.1093/nar/25.17.3389.

[18] M. Gribskov, A. D. McLachlan, and D. Eisenberg, “Profile analysis: detection of distantly related proteins.,” *Proc. Natl. Acad. Sci. U. S. A.*, 1987, doi: 10.1073/pnas.84.13.4355.

[19] A. Bairoch, “PROSITE: A dictionary of sites and patterns in proteins,” *Nucleic Acids Research*. 1992, doi: 10.1093/nar/20.suppl.2013.

[20] T. K. Attwood, M. E. Beck, D. R. Flower, P. Scordis, and J. N. Selley, “The PRINTS protein fingerprint database in its fifth year,” *Nucleic Acids Res.*, 1998, doi: 10.1093/nar/26.1.304.

[21] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. Mcclure, “Hidden Markov models of biological primary sequence information,” *Proc. Natl. Acad. Sci. U. S. A.*, 1994, doi: 10.1073/pnas.91.3.1059.

[22] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler, “Hidden Markov Models in computational biology applications to protein modeling,” *J. Mol. Biol.*, 1994, doi: 10.1006/jmbi.1994.1104.

[23] S. R. Eddy, “Multiple alignment using hidden Markov models.,” *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 1995.

[24] J. McDowall and S. Hunter, “InterPro protein classification.,” *Methods Mol. Biol.*, 2011, doi: 10.1007/978-1-60761-977-2\_3.

[25] W. Chmielnicki and K. Staçpor, “A hybrid discriminative/generative approach to protein fold recognition,” *Neurocomputing*, 2012, doi: 10.1016/j.neucom.2011.04.033.

[26] C. H. Q. Ding and I. Dubchak, “Multi-class protein fold recognition using support vector machines and neural networks,” *Bioinformatics*, 2001, doi: 10.1093/bioinformatics/17.4.349.

[27] C. Leslie, E. Eskin, and W. S. Noble, “The spectrum kernel: a string kernel for SVM protein classification.,” *Pac. Symp. Biocomput.*, 2002, doi: 10.1142/9789812799623\_0053.

[28] T. Jaakkola, M. Diekhans, and D. Haussler, “Using the Fisher kernel method to detect remote protein homologies.,” *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 1999.

[29] A. C. Tan, D. Gilbert, and Y. Deville, “Multi-class protein fold classification using a new ensemble machine learning approach.,” *Genome Inform.*, 2003, doi: 10.11234/gi1990.14.206.

[30] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, “Mismatch string kernels for discriminative protein classification,” *Bioinformatics*, 2004, doi: 10.1093/bioinformatics/btg431.

[31] S. Diplaris, G. Tsoumakas, P. A. Mitkas, and I. Vlahavas, “Protein classification with multiple algorithms,” 2005, doi: 10.1007/11573036\_42.

[32] O. Çamoǧlu, T. Can, A. K. Singh, and Y. F. Wang, “Decision tree based information integration for automated protein classification,” *J. Bioinform. Comput. Biol.*, 2005, doi: 10.1142/S0219720005001259.

[33] Z. Xing, J. Pei, and E. Keogh, “A brief survey on sequence classification,” *ACM SIGKDD Explor. Newsl.*, 2010, doi: 10.1145/1882471.1882478.

[34] L. Nanni, A. Lumini, and S. Brahnam, “An empirical study of different approaches for protein classification,” *Sci. World J.*, 2014, doi: 10.1155/2014/236717.

[35] S. Nojoomi and P. Koehl, “String kernels for protein sequence comparisons: Improved fold recognition,” *BMC Bioinformatics*, 2017, doi: 10.1186/s12859-017-1560-9.

[36] P. Khurana, R. S. Gokhale, and D. Mohanty, “Genome scale prediction of substrate specificity for acyl adenylate superfamily of enzymes based on active site residue profiles,” *BMC Bioinformatics*, 2010, doi: 10.1186/1471-2105-11-57.

[37] S. R. Amin, S. Erdin, R. M. Ward, R. C. Lua, and O. Lichtarge, “Prediction and experimental validation of enzyme substrate specificity in protein structures,” *Proc. Natl. Acad. Sci. U. S. A.*, 2013, doi: 10.1073/pnas.1305162110.

[38] G. X. Yu, B. H. Park, P. Chandramohan, R. Munavalli, A. Geist, and N. F. Samatova, “In silico discovery of enzyme-substrate specificity-determining residue clusters,” *J. Mol. Biol.*, vol. 352, no. 5, pp. 1105–1117, 2005, doi: 10.1016/j.jmb.2005.08.008.

[39] M. Deshpande and G. Karypis, “Evaluation of techniques for classifying biological sequences,” 2002, doi: 10.1007/3-540-47887-6\_41.

[40] G. Rätsch, S. Sonnenburg, and C. Schäfer, “Learning interpretable SVMs for biological sequence classification,” *BMC Bioinformatics*, 2006, doi: 10.1186/1471-2105-7-S1-S9.

[41] S. Sonnenburg, G. Rätsch, and B. Schölkopf, “Large scale genomic sequence SVM classifiers,” 2005, doi: 10.1145/1102351.1102458.

[42] H. Saigo, J. P. Vert, N. Ueda, and T. Akutsu, “Protein homology detection using string alignment kernels,” *Bioinformatics*, 2004, doi: 10.1093/bioinformatics/bth141.

[43] T. Jaakkola, M. Diekhans, and D. Haussler, “A discriminative framework for detecting remote protein homologies,” *Journal of Computational Biology*. 2000, doi: 10.1089/10665270050081405.

[44] G. Rätsch, S. Sonnenburg, and B. Schölkopf, “RASE: Recognition of alternatively spliced exons in C.elegans,” *Bioinformatics*, 2005, doi: 10.1093/bioinformatics/bti1053.

[45] H. Nakashima, K. Nishikawa, and T. Ooi, “The folding type of a protein is relevant to the amino acid composition,” *J. Biochem.*, 1986, doi: 10.1093/oxfordjournals.jbchem.a135454.

[46] K.-C. Chou, “Pseudo Amino Acid Composition and its Applications in Bioinformatics, Proteomics and System Biology,” *Curr. Proteomics*, 2009, doi: 10.2174/157016409789973707.

[47] L. Nanni, A. Lumini, and S. Brahnam, “An empirical study on the matrix-based protein representations and their combination with sequence-based approaches,” *Amino Acids*, 2013, doi: 10.1007/s00726-012-1416-6.

[48] S. Whalen and G. Pandey, “A comparative analysis of ensemble classifiers: Case studies in genomics,” 2013, doi: 10.1109/ICDM.2013.21.

[49] C. Caragea, J. Sinapov, A. Silvescu, D. Dobbs, and V. Honavar, “Glycosylation site prediction using ensembles of Support Vector Machine classifiers,” *BMC Bioinformatics*, 2007, doi: 10.1186/1471-2105-8-438.

[50] A. Jones, H. M. Davies, and T. A. Voelker, “Palmitoyl-acyl carrier protein (ACP) thioesterase and the evolutionary origin of plant acyl-ACP thioesterases,” *Plant Cell*, 1995, doi: 10.1105/tpc.7.3.359.

[51] T. A. Voelker and H. M. Davies, “Alteration of the specificity and regulation of fatty acid synthesis of Escherichia coli by expression of a plant medium-chain acyl-acyl carrier protein thioesterase,” *J. Bacteriol.*, 1994, doi: 10.1128/jb.176.23.7320-7327.1994.

[52] F. Jing, L. Zhao, M. D. Yandeau-Nelson, and B. J. Nikolau, “Two distinct domains contribute to the substrate acyl chain length selectivity of plant acyl-ACP thioesterase,” *Nat. Commun.*, 2018, doi: 10.1038/s41467-018-03310-z.

[53] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch, “Support vector machines and kernels for computational biology,” *PLoS Comput. Biol.*, 2008, doi: 10.1371/journal.pcbi.1000173.

[54] Y. Yang *et al.*, “Boundary thickness and robustness in learning models,” *arXiv*. 2020.

[55] P. Kuksa, P. H. Huang, and V. Pavlovic, “A fast, large-scale learning method for protein sequence classification,” 2008.

[56] D. Jurafsky and J. H. Martin, “Language Modeling with N- grams,” *Speech Lang. Process.*, 2016.

[57] M. Bhasin and G. P. S. Raghava, “Classification of nuclear receptors based on amino acid composition and dipeptide composition,” *J. Biol. Chem.*, 2004, doi: 10.1074/jbc.M401932200.

[58] K. Chen, L. Kurgan, and M. Rahbari, “Prediction of protein crystallization using collocation of amino acid pairs,” *Biochem. Biophys. Res. Commun.*, 2007, doi: 10.1016/j.bbrc.2007.02.040.

[59] V. Saravanan and N. Gautham, “Harnessing computational biology for exact linear B-cell epitope prediction: A novel amino acid composition-based feature descriptor,” *Omi. A J. Integr. Biol.*, 2015, doi: 10.1089/omi.2015.0095.

[60] T. Y. Lee, Z. Q. Lin, S. J. Hsieh, N. A. Bretaña, and C. T. Lu, “Exploiting maximal dependence decomposition to identify conserved motifs from a group of aligned signal sequences,” *Bioinformatics*, 2011, doi: 10.1093/bioinformatics/btr291.

[61] Z. Chen *et al.*, “IFeature: A Python package and web server for features extraction and selection from protein and peptide sequences,” *Bioinformatics*, 2018, doi: 10.1093/bioinformatics/bty140.

[62] Z. P. Feng and C. T. Zhang, “Prediction of membrane protein types based on the hydrophobic index of amino acids,” *J. Protein Chem.*, 2000, doi: 10.1023/A:1007091128394.

[63] R. R. Sokal and B. A. Thomson, “Population structure inferred by local spatial autocorrelation: An example from an Amerindian tribal population,” *Am. J. Phys. Anthropol.*, 2006, doi: 10.1002/ajpa.20250.

[64] D. S. Horne, “Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities,” *Biopolymers*, 1988, doi: 10.1002/bip.360270308.

[65] C. Z. Cai, L. Y. Han, Z. L. Ji, X. Chen, and Y. Z. Chen, “SVM-Prot: Web-based support vector machine software for functional classification of a protein from its primary sequence,” *Nucleic Acids Res.*, 2003, doi: 10.1093/nar/gkg600.

[66] I. Dubchak, I. Muchnik, S. R. Holbrook, and S. H. Kim, “Prediction of protein folding class using global description of amino acid sequence,” *Proc. Natl. Acad. Sci. U. S. A.*, 1995, doi: 10.1073/pnas.92.19.8700.

[67] L. Y. Han, C. Z. Cai, S. L. Lo, M. C. M. Chung, and Y. Z. Chen, “Prediction of RNA-binding proteins from primary sequence by a support vector machine approach,” *RNA*, 2004, doi: 10.1261/rna.5890304.

[68] J. Shen *et al.*, “Predicting protein-protein interactions based only on sequences information,” *Proc. Natl. Acad. Sci. U. S. A.*, 2007, doi: 10.1073/pnas.0607879104.

[69] K. C. Chou, “Prediction of protein cellular attributes using pseudo-amino acid composition,” *Proteins Struct. Funct. Genet.*, 2001, doi: 10.1002/prot.1035.

[70] T. Liu, X. Zheng, and J. Wang, “Prediction of protein structural class for low-similarity sequences using support vector machine and PSI-BLAST profile,” *Biochimie*, 2010, doi: 10.1016/j.biochi.2010.06.013.

[71] T. Liu, X. Geng, X. Zheng, R. Li, and J. Wang, “Accurate prediction of protein structural class using auto covariance transformation of PSI-BLAST profiles,” *Amino Acids*, 2012, doi: 10.1007/s00726-011-0964-5.

[72] Y. Guo, L. Yu, Z. Wen, and M. Li, “Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences,” *Nucleic Acids Res.*, 2008, doi: 10.1093/nar/gkn159.

[73] S. Ding, Y. Li, Z. Shi, and S. Yan, “A protein structural classes prediction method based on predicted secondary structure and PSI-BLAST profile,” *Biochimie*, 2014, doi: 10.1016/j.biochi.2013.09.013.

[74] P. Tao, T. Liu, X. Li, and L. Chen, “Prediction of protein structural class using tri-gram probabilities of position-specific scoring matrix and recursive feature elimination,” *Amino Acids*, 2015, doi: 10.1007/s00726-014-1878-9.

[75] L. Zhang, X. Zhao, and L. Kong, “Predict protein structural class for low-similarity sequences by evolutionary difference information into the general form of Chou[U+05F3]s pseudo amino acid composition,” *J. Theor. Biol.*, 2014, doi: 10.1016/j.jtbi.2014.04.008.

[76] S. Zhang, F. Ye, and X. Yuan, “Using principal component analysis and support vector machine to predict protein structural class for low-similarity sequences via PSSM,” *J. Biomol. Struct. Dyn.*, 2012, doi: 10.1080/07391102.2011.672627.

[77] *et al.*, “Protein Fold Recognition Using Genetic Algorithm Optimized Voting Scheme and Profile Bigram,” *J. Softw.*, 2016, doi: 10.17706/jsw.11.8.756-767.

[78] J. Zahiri, O. Yaghoubi, M. Mohammad-Noori, R. Ebrahimpour, and A. Masoudi-Nejad, “PPIevo: Protein-protein interaction prediction from PSSM based evolutionary information,” *Genomics*, 2013, doi: 10.1016/j.ygeno.2013.05.006.

[79] K. C. Chou and H. Bin Shen, “MemType-2L: A Web server for predicting membrane proteins and their types by incorporating evolution information through Pse-PSSM,” *Biochem. Biophys. Res. Commun.*, 2007, doi: 10.1016/j.bbrc.2007.06.027.

[80] E. Y. T. Juan, W. J. Li, J. H. Jhang, and C. H. Chiu, “Predicting protein subcellular localizations for gram-negative bacteria using DP-PSSM and support vector machines,” 2009, doi: 10.1109/CISIS.2009.194.

[81] L. Zou, C. Nan, F. Hu, and J. Hancock, “Accurate prediction of bacterial type IV secreted effectors using amino acid composition and PSSM profiles,” *Bioinformatics*, 2013, doi: 10.1093/bioinformatics/btt554.

[82] C. W. Cheng, E. C. Y. Su, J. K. Hwang, T. Y. Sung, and W. L. Hsu, “Predicting RNA-binding sites of proteins using support vector machines and evolutionary information,” 2008, doi: 10.1186/1471-2105-9-S12-S6.

[83] J. C. Jeong, X. Lin, and X. W. Chen, “On position-specific scoring matrix for protein function prediction,” *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, 2011, doi: 10.1109/TCBB.2010.93.

[84] J. Palme, S. Hochreiter, and U. Bodenhofer, “KeBABS: An R package for kernel-based analysis of biological sequences,” *Bioinformatics*, 2015, doi: 10.1093/bioinformatics/btv176.

[85] NumPy, “NumPy — NumPy,” *NumPy Website*, 2017. .

[86] S. Kawashima and M. Kanehisa, “AAindex: Amino acid index database,” *Nucleic Acids Research*. 2000, doi: 10.1093/nar/28.1.374.

[87] B. E. Suzek, H. Huang, P. McGarvey, R. Mazumder, and C. H. Wu, “UniRef: Comprehensive and non-redundant UniProt reference clusters,” *Bioinformatics*, 2007, doi: 10.1093/bioinformatics/btm098.

[88] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “Training algorithm for optimal margin classifiers,” 1992, doi: 10.1145/130385.130401.

[89] M. Aly and <malaa@caltech Edu>, “Survey on multiclass classification methods,” *Neural Netw*, 2005.

[90] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, 2011.