Comparative study on the utility of protein spectra and protein descriptors in the analysis of sequence activity relationships

Adam McKenna[1], Sandhya P N Dubey[2]

[1]*School of Electronics, Electrical Engineering and Computer Science, Queen's University of Belfast*
[2]*Department of Computer Applications, Manipal Academy of Higher Education (MAHE)*

[1]amckenna41@qub.ac.uk, [2]sandhya.dubey@manipal.edu

Abstract*— Accurately establishing the connection between a protein sequence and its function remains a focal point within the field of protein engineering, especially in the context of predicting the effects of mutations. From this, there has been a continued drive to build accurate and reliable predictive models via machine learning that allow for the virtual screening of a large number of protein mutant sequences, measuring the relationship between sequence and 'fitness' or 'activity', commonly known as a Sequence-Activity-Relationship (SAR). An important preliminary stage in the building of these predictive models is the encoding of the chosen sequences. Evaluated in this work is a plethora of encoding strategies using the Amino Acid Index database, where the indices are transformed into their spectral form via Digital Signal Processing (DSP) techniques, as well as numerous protein structural and physiochemical descriptors. The encoding strategies are explored on a dataset curated to measure the thermostability of various mutants from a recombination library, designed from parental cytochrome P450s. It is found that the choice of encoding strategy can significantly impact the results, highlighting that there exists an optimal way of encoding protein sequences for each dataset. The implementation of protein spectra in concatenation with protein descriptors gave a noteworthy increase in the quality of the predictive models, highlighting their utility in identifying an SAR. The accompanying software produced for this paper is termed *pySAR* (Python Sequence-Activity-Relationship), which allows for a user to find the optimal structural and or physiochemical properties to encode their specific mutant library dataset; the source code is available at: *https://github.com/amckenna41/pySAR

Keywords*— *digital signal processing, directed evolution, machine learning, protein spectra, physiochemical descriptors.

I. Introduction

Proteins are involved in virtually every known biological and cellular process, therefore accurate prediction and determination of a protein's functionality in an organism remains a primary goal within the field of Bioinformatics. Proteins are involved in replicating and transcribing DNA, controlling cell division, metabolism, enzymes, cell signaling and ligand binding, and many other processes [1]. They are made up of one or more polypeptide chains of amino acid residues, arranged in primary to quaternary structures. Much focus in Bioinformatics focuses on the primary structure, which remains the most widely available and used type of data for much protein-related research [2].

Protein engineering involves the process of modifying a protein sequence through substitution, deletion or insertion of amino acids, creating 'mutants' of the original sequence. The goal of this process is the construction of modified proteins with desired properties; be it catalytic activity, enzyme stability, optimum temperature or PH, substrate specificity or other property of interest [3]. The two main procedures for protein engineering have been rational design and directed evolution, as well as a hybrid of both.

Directed Evolution (DE) is a methodology for protein engineering that mimics the process of natural selection, optimizing a protein and its variants through iterative rounds of mutagenesis, selection, and amplification, towards a desired characteristic or user-defined goal, normally known as 'fitness' or 'activity' [4] ('fitness' and 'activity' are used interchangeably throughout this paper). The initial step involves sequence diversification such as site-saturation mutagenesis, random mutagenesis, or recombination to generate a library of mutant sequences derived from a parent sequence(s), with experimentally determined 'fitness' values, creating a 'fitness landscape.' Secondly is the screening or selection process to identify mutants/variants with enhanced or neutral 'fitness' values to keep for the next iteration of diversification [4]. Contrastingly, deleterious variants that have been found to give a reduction in 'fitness' value can be discarded for the next iteration. Figure 1 shows the three stages of DE, as well as the natural selection process being mimicked, used within the evolution of a viral protease enzyme to enhance its catalysis, robustness, and specificity [5]. Overall, the process can be costly and time-consuming, thus many in silico approaches have been realized to assist in correlating the effect of mutating a sequence on the characteristic/activity being studied; known as a Sequence Activity Relationship (SAR) [6].

The process of DE can be hugely challenging, owing to the high-dimensionality nature of the problem, with an inconceivable number of possible amino acid combinations in a medium to large-sized protein sequence. For example, a medium-sized protein of 300 residues in length has around 4.6×10^{396} possible combinations of the 21 proteinogenic amino

acids. Hence, it is impossible to screen 21^N protein variants, either computationally or in the lab, where N is the length of the sequence. Due to this barrier, Machine Learning (ML) methodologies have been widely explored, allowing for the virtual screening of a large number of protein variants. The aim of in silico approaches for DE is to speed up and make more efficient the process of acquiring sequences and mutations that give the best results, in terms of the desired characteristic [7].

The inclusion of ML within the field of protein engineering is fairly new, with existing studies in the literature giving variable results, exploring small numbers of proteins and predictive methodologies. Yang et al. exploited Partial Least Squares (PLS) Regression and Bayesian Optimization to study the productivity of enzymes and the thermostability of cytochrome P450s [8]. Mason et al. used deep neural networks to help in the accurate prediction of antigen-binding based on antibody sequence, through the screening and deep sequencing of small libraires of protein sequences [9]. Other methodologies have assessed a wide range of different algorithmic models and hyperparameters on a selection of both public and proprietary datasets [10] – [13].

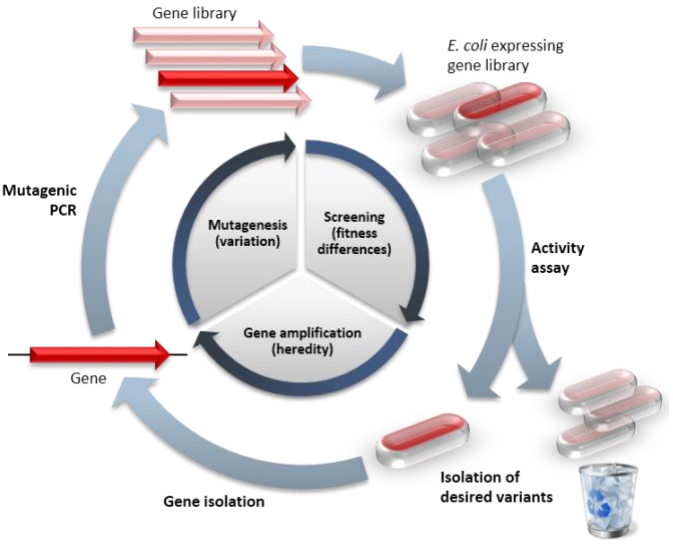

Fig.1. DE cycle, inner circle shows the 3 stages of the cycle, with the natural process mimicked in brackets. Outer circle illustrates the procedures for a typical DE experiment [5].

Many of the aforementioned techniques all follow a similar approach of three phases – encoding, modelling and predicting. The encoding phase is arguably one of the most widely studied and reviewed of these de facto phases, in the context of protein engineering, as it is an essential pre-cursor for the subsequent two phases and can have a significant impact on their results. The encoding stage usually involves encoding the protein sequences as vectors of numbers according to certain criteria, so that they are compatible with the mathematical operations of ML models. Therefore, careful consideration needs to be made in this process, as how a protein is numerically represented determines what can be learned from it by via ML techniques [14]. Deciding on the encoding criteria, and thus the properties of the protein sequences to use is difficult to know a priori, attributed to the molecular properties that dictate functional properties being unknown, highly constrained or differing between different functional properties [15]. Some strategies for encoding have included one-hot encoding [8], residual frequency and Doc2Vec embeddings [16]. A consideration with these strategies is the lack of attention for the residue characteristics and physiochemical properties. Other encoding strategies have mitigated this downfall by explicitly encoding the sequences according to physiochemical properties and descriptors of the sequence residues.

Physiochemical properties are experimentally measured values for each of the 20 amino acids. Some of the properties relate to properties of hydrophobicity, solubility, secondary structure, or residue propensity, to name a few. They have seen a wide variety of use in many Bioinformatics related topics, including in some recent protein engineering applications [6] – [15]. In the methodology proposed in this paper, these physiochemical properties are transformed into protein spectra via DSP-techniques.

DSP consists of the mathematics, algorithms and techniques used to manipulate digitized continuous or discrete signals and allows for the decomposition and processing of signals to reveal embedded information within them [17]. DSP has been used in various fields and applications, including within Bioinformatics to aid in understanding the biological characteristics of a protein; being used in capturing protein sequence activity [7], genome classification [18] and predicting secondary protein structure [19]. DSP has also shown promise when used in conjunction with physiochemical properties of amino acids and has featured in various studies [7], [11], [13], [18], [20].

Veljkovic et al. [21] proposed a methodology that encoded protein sequences according to the Electron-Ion Interaction Potential (EIIP) property followed by a Discrete Fourier Transform (DFT) to extract information corresponding to biological function. The authors concluded that peak frequencies within a cross spectrum analysis are similar for functionally related sequences and vice versa. Various approaches have followed since this, including the Resonant Recognition Model (RRM) and Informational Spectrum Method (ISM). These two approaches have been thoroughly reviewed by Nwankwo N. and Seker H [20]. In these methodologies it was found that the generated protein spectral characteristics reveal shared biological functionalities, demonstrated by peaks within the frequency spectrum. Therefore, within the context of DE, the generated protein spectra and their associated frequencies may prove useful in identifying the highest achieving mutants, whereby mutants with similar activity produce a similar characteristic frequency peak in their respective spectra. Correspondingly, the influence of a mutation on the activity value should also be apparent from the differing protein spectra for each mutated sequence.

Most recent approaches have gone one step further than the previous techniques and created statistical predictive models from the generated protein spectra to assist in the virtual screening and prediction of biological fitness in the process of DE. The ML methodology known as innov'Sar (innovative sequence-activity relationship) was introduced by Cadet et al. [7]. The author's approach involved the similar technique of transforming the protein mutant sequences to spectral form, then utilizing all the generated spectra as a training set to build a predictive model via PLS Regression that effectively

identifies enantioselective mutants of the epoxide hydrolase from the *Aspergillus Niger* species (ANEH). This procedure of generating statistical predictive models to predict the sought-after fitness value form the modelling and predictive phases.

Furthermore, additional to protein physiochemical properties are protein descriptors. These descriptors differ from physiochemical properties in that they encode sequence-derived structural information, although there does exist a slight overlap, with some descriptors created from a combination of physiochemical properties. These descriptors relate to amino acid composition, residue autocorrelation, conjoint triad, quasi-sequence-order and CTD (composition, transition and distribution). They have seen use in several ML-based protein engineering applications [6], [10], [22] – [25].

In reviewing the past literature, it was found that there has not been an explicit quantitative or qualitative comparative analysis of the utility of protein spectra and protein descriptors in the building of predictive models, within the context of analyzing SAR's. The methodology in this work will evaluate the performance and utility of the strategies of encoding using protein spectra built from physiochemical properties and protein descriptors in the building of the predictive models, using both in isolation and in combination with one another. The physiochemical properties were taken from the AAIndex (AAI) database, which to date has 566 different property indices [26]. Fifteen descriptors were calculated with help from the PyBioMed Python package [27], generating a feature space of size N X 9920 where N is the number of protein sequences in the dataset and 9920 is the total number of calculable features from the 15 descriptors. Firstly, predictive models were built on all 566 indices individually, followed by using combinations of 1, 2 or 3 descriptors (there being 15, 105 and 455 combinations of 1, 2 and 3 descriptors, respectively) and finally models built using a combination of 1 AAI and 1 or 2 descriptors, equating to (8490 and 59,430 different combinations). Further, six regression algorithms in total were tried and tested for each strategy. Therefore, a total of 414,366 different combinations of encoding techniques are explored in this work (69,061 per algorithm), falling under three main encoding strategies: *A. AAI Indices*, *B. Descriptors* and *C. AAI Indices + Descriptors*. This provides a comprehensive evaluation of which technique and associated properties are the most applicable and useful for studying the effect of mutations on the protein fitness value investigated.

The dataset used to evaluate all the aforementioned encoding strategies, as well as the *pySAR* software itself, was a dataset produced to measure the thermostability of various mutants from a recombination library designed from parental cytochrome P450s [28]. Thus, the aim of each encoding strategy is to extract as much useful information from the protein sequences, with their thermostability value measured, such that an accurate predictive model is built that, with high accuracy, can predict the thermostability of new and unseen protein sequences.

II. Methods

A. AAI Indices

The first step of the AAI Indices-based encoding strategy was in the encoding of the protein sequences in the dataset according to physiochemical properties of amino acids from indices in the AAI database, which is a database of various physiochemical properties of each amino acid that have been measured and compiled through a large body of experimental and theoretical research [26].

Prior to index selection, each index was standardized via the Z-score (1), such that the values were centered around the mean with a unit standard deviation, so that the mean of the index became 0.

$$X' = \frac{X - \mu}{\sigma} \tag{1}$$

where μ and σ are the mean and standard deviation of the index values, respectively, X is the observed value and X' is the standardized score.

B. Preprocessing of Protein Sequences

Preceding the conversion of the encoded protein sequences to spectra in the AAI Indices-based encoding strategy, pre-processing was done that has been shown to assist in enhancing the features extracted from the sequences [29]. Firstly, zero-padding was performed that involved appending zeros to the end of the numerical sequences, after their AAI encoding, so that they were all the same length, such that:

$$\forall n: n \ni N \mid |n| = \max(N) \tag{2}$$

where n is the encoded protein sequence, N is the dataset of all sequences, $|n|$ is the cardinality/length of a sequence and max(N) is the longest sequence in the dataset N.

Additional pre-processing undertaken was the replacing of any null or missing values in the encoded sequences with 0's, as was also done for any of the measured thermostability activity values in the dataset. The next pre-processing step was applying a window function/windowing, with its application shown to enhance the features extracted from the protein sequences [29]. A widely used and powerful window function is the Hamming Window (3), which is a taper formed by using a raised cosine with non-zero endpoints, optimized to minimize the nearest side lobe [29]. It has also been used within the analysis of protein sequences and can be represented as:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{M-1}\right) \qquad 0<=n<=M-1 \tag{3}$$

where $w(n)$ is the filter coefficients of the window of length M.

C. Signal Processing of Encoded Protein Sequences

Following the encoding of the protein sequences using indices from the AAI and pre-processing, the next significant step was in the transformation of the sequences into protein spectra via a Fourier Transform (4). A Fast Fourier Transfer (FFT) is an algorithm for computing the DFT that converts a signal from its original domain (time or space) to a representation in the frequency domain [30].

The application of a Fourier Transform has seen recurrent use within the context of analyzing proteins in protein engineering [7], [11], [13], [22], and allows for the protein sequences to be represented via a protein spectrum to help extract information corresponding to biological function [20]. In [7], the authors evaluated numerically encoded physiochemical amino acid

properties with and without the application of an FFT, finding that coupling multiple physiochemical descriptors with an FFT significantly improved the predictive performance of their models. Additionally, in the iSAR methodology [22], the authors also found that not using an FFT reduced their models' predictive performance. The Fourier Transform can be represented as:

$$f_j = \sum_{k=0}^{N-1} x_k e^{\frac{-2i\pi}{N}jk} \quad (4)$$

where f_j is the output spectrum of complex numbers, j is an index of the Fourier Transform, x_k is the input signal (encoded protein sequence) of length N where $0 \leq k \leq N-1$ and $N \geq 1$, k is the frequency in the spectrum and i is the imaginary number such that $i^2 = -1$.

The output of the FFT is a complex sequence $\mathbb{C}$ where each element is in the form $a + bi$ where a is the real component, b is the imaginary component and i is an imaginary number. The two components of the complex sequence can be represented as:

$$X(n) = \big(\Re(n) + \Im(n)\big), n = 1,2,\dots,\frac{N}{2} \quad (5)$$

where $\Re(n)$ and $\Im(n)$ are the real and imaginary components of the protein sequence, respectively.

From these two components, a real and imaginary protein spectrum can be generated, as well as an absolute or power spectrum. The power spectrum (6) displays the coefficients for each frequency measured by the FFT as a graph of power values [31] and was the spectra of choice for the AAI-based encoding strategy studied, being represented as:

$$PS(m) = |f_j|^2, m=1,2,\dots,M \quad (6)$$

where f_j is the output of the FFT and M is the number of protein sequences.

D. Descriptors

In addition to the encoded protein sequences via selected AAI indices and protein spectra, protein descriptors were also used as an encoding strategy within the methodology to ascertain whether their inclusion improved the predictability of the models, and if so, which descriptors worked best. Protein descriptors have been widely utilized in similar areas as to the AAI, including protein structure and functional class prediction [23], protein subcellular location prediction [32] and protein-protein interactions [24].

The similarity of the research areas can be attributed to the slight overlap between the descriptors and the indices in the AAI, with some of the descriptors composed of an ensemble of different index values, for example eight amino acid properties are used for deriving the autocorrelation and CTD (composition, transition and distribution) descriptors [25]. This overlap was taken into consideration upon analysis of the results obtained. Some of the descriptors explored included the amino-acid, dipeptide, and tripeptide composition, all in the descriptor group of amino acid composition, which outlines the proportion of each amino acid, dipeptide and tripeptide, respectively, in each of the protein sequences. Also explored are numerous descriptors coming under the groups of autocorrelation, CTD, conjoint triad, quasi-sequence-order and pseudo amino acid composition, for a total of 15 descriptor types explored, with all of the associated descriptors shown in Table I. The derivations and calculations for each of the descriptors was extensively outlined by [25], where the authors utilized protein descriptors in predicting protein functional families.

All the protein descriptor values are calculated in the *descriptors* module of the *pySAR* software, with the help of the PyBioMed Python library [27]. PyBioMed is a feature-rich and highly customized library that allows for the extraction of a plethora of chemical and biological features and descriptors. It can also calculate a total of 9920 protein features using its built-in PyProtein module. To get the required descriptors was a matter of pre-processing the protein sequences in such a way that they can be used by the package, mainly removing any gaps ('-') from the sequences. The resultant descriptor values were

TABLE I

PROTEIN DESCRIPTORS AVAILABLE IN *PYSAR*

Descriptor	Dimension	Descriptor Group
Amino Acid Composition (AAComp[a])	20	Amino Acid Composition
Dipeptide Composition (DPComp[a])	400	Amino Acid Composition
Tripeptide Composition (TPComp[a])	8000	Amino Acid Composition
Normalized Moreau-Broto Autocorrelation (NMBAuto[a])	240[b]	Autocorrelation
Moran Autocorrelation (MAuto[a])	240[b]	Autocorrelation
Geary Autocorrelation (GAuto[a])	240[b]	Autocorrelation
Composition (C_CTD[a])	21	CTD
Transition (T_CTD[a])	21	CTD
Distribution (D_CTD[a])	105	CTD
CTD	147	CTD
Conjoint Triad (CTriad[a])	343	Conjoint Triad
Sequence-Order-Coupling Number (SOCNum[a])	60[c]	Quasi-sequence-order
Quasi-Sequence-Order (QSOrder[a])	100	Quasi-sequence-order
Pseudo Amino Acid Composition (PAAComp[a])	50[d]	Pseudo Amino Acid Composition
Amphiphilic Pseudo-Amino Acid composition (APAAComp[a])	80[e]	Pseudo Amino Acid Composition

[a] The descriptor abbreviations will be used in reference to them from herein.

[b] The dimension of the Autocorrelation descriptors depend on the number of amino acid properties used and the maximum lag value. In *pySAR*, by default, 8 properties are used with a maximum lag of 30.

[c] The dimension of the SOCNum depends on the maximum lag value, by default it is set to 30.

[d] The dimension of the PAAComp descriptor depends on the number of amino acid properties used. In *pySAR*, by default, three types of properties are used (hydrophobicity, hydrophilicity and residue mass), with a lambda of 30.

[e] The dimension of the APAAComp descriptor depends on the chosen lambda value, default is set to 30.

then post-processed in a way that they can be used with *pySAR*, such as reshaping and reformatting the output from PyBioMed. Using *pySAR*, it took less than two minutes to calculate 13 out of the 15 descriptors for the 261 sequences, but it took a considerably lengthier time to calculate the remaining two descriptors of PAAComp and APAAComp, taking around 75 minutes in total to calculate, on average. Although, this process can just be executed once per dataset, with all the descriptors then being exported to a csv for future use.

Similar to the AAI indices, the protein descriptors were each passed through a preliminary pre-processing step of standardization via the Z-score (1), so each descriptor was centered around its mean with a unit standard deviation such that the mean of the descriptor became 0

E. Predictive Models

The final step in the methodology, after acquisition of the required datasets and selection of the encoding technique for the sequences is the choosing of the predictive model, the modelling phase (Figure 2c). The problem in question is a regression problem, which is a mathematical technique used to predict a continuous outcome dependent variable (Y) based on the value of one or more predictor independent variables (X). In these models the X will consist of a feature vector made up of various combinations of encoded protein spectra via AAI indices and or protein descriptors, with the Y target variable being the desired protein activity/characteristic, a continuous variable. The objective of these models is to learn from the encoded protein sequence data so that they can generalize and establish an SAR between the sequences and activity, such that they can predict the sought activity value for new unseen sequences. There is a wide array of applicable and useful supervised ML regression models and algorithms; those explored and evaluated in this work were PLS[a], Random Forest (RF[a]), AdaBoost (Ada[a]), SVR[a] (Support Vector Regression), Bagging (Bag[a]) and KNN[a] (K-Nearest-Neighbors). Each of these algorithms have shown great results in a range of ML applications, including within the context of protein engineering and DE [6] – [12], [15], [18], [22].

pySAR implements all of the model related functions and methods via the *Model* module, which itself builds upon the Sci-Kit Learn package. All models were initially implemented using their default parameters - see the Sci-Kit learn documentation for the full list of the models' parameters.

[a] From herein the regression algorithms will be denoted by their abbreviation in the brackets, KNN, SVR and PLS will remain unchanged.

III. Implementation

All the predictive models and encoding strategies created and evaluated using the *pySAR* software were built using Python 3.7. The Sci-Kit Learn framework is a popular and robust ML library and was used for all the ML processes. Furthermore, SciPy was used for any of the DSP related processes, including the encoding of the protein sequences into protein spectra via an FFT. Numpy was used heavily throughout to work with arrays and matrices that many functions relied on as input or generated as output and Pandas played a role in transforming the results into a readable and manageable data format for analysis. The visualization libraries Seaborn and Matplotlib were used for any of the visualizations, both in *pySAR* and in the results section of this paper. Google Collab was employed for much of the required results generation, with the default Collab notebook being a n1-highmem-2 Google machine type which possesses the hardware specifications of 2 vCPUs (Intel Xeon E3-1220L v2 @ 2.30GHZ) and 13GB of RAM.

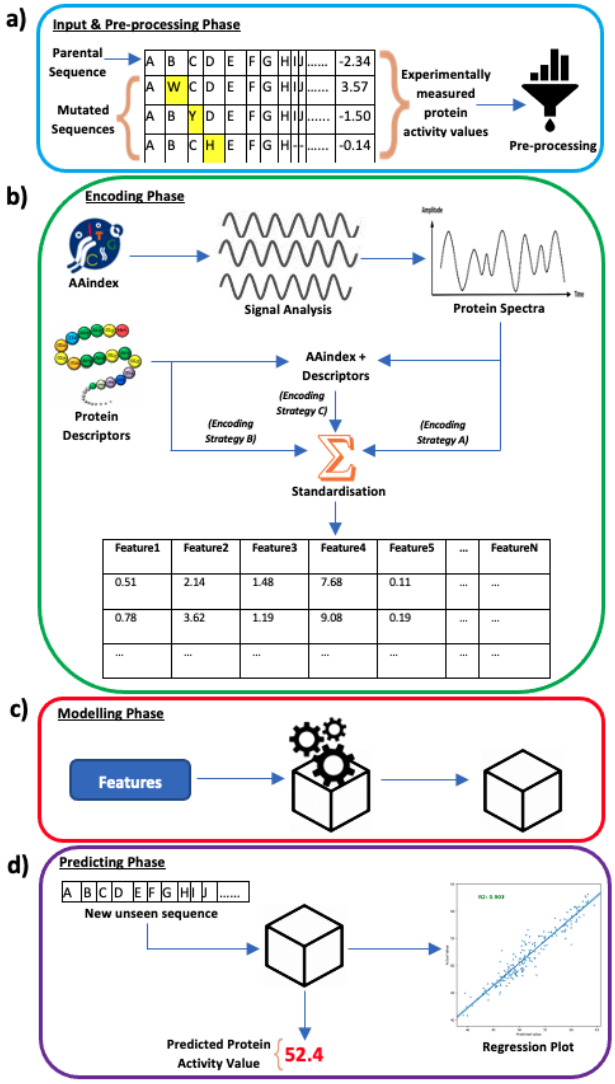

Fig. 2. *pySAR* methodology. a) shows the process of data acquisition of the mutated protein sequences from a parental sequence and their associated protein activity values as well as the pre-processing of these sequences, b) is the encoding phase where the sequences are encoded via AAI indices, descriptors or AAI indices + Descriptors, c) is the modelling phase where the predictive model is built using the generated features b) and the d) is the predicting phase, where a new sequence's activity value is predicted from the built predictive model.

IV. Results

To build and evaluate the models to predict the chosen protein activity value (thermostability), the protein sequences are firstly required to be encoded into numerical values (Figure 2b). The two main encoding strategies were explained and outlined in the previous *Methods* section that involved utilizing indices from the AAI database (*A)* and sequence-derived structural and physiochemical protein descriptors (*B*). Each of these two strategies are made up of a plethora of possible features at which to encode the protein sequences. The main objective of the methodology is to evaluate whether using the

indices from the AAI or protein descriptors prove more effective in building a model with high predictability/generalizability. A third and final strategy is also generated involving a combination of the two main strategies – AAI indices + Descriptors (*C*).

The notation for the final predictive models is in the general form: *encoding_algorithm* where e*ncoding* shows the AAI index, descriptor, or AAI index + Descriptor encoding strategy and *algorithm* is the regression algorithm used to build the predictive model for the dataset using the mentioned encoding strategy. The AAI indices will be denoted by their index in the form AA_X where *x = 1,2,3,..,566* (the AAI index code). Correspondingly, the descriptors are represented by their abbreviations as listed in Table I and the algorithms are abbreviated in the form mentioned in the footnote of *E. Predictive Models*. For cases where multiple descriptors or AAI indices have been used alongside descriptors, the '^' symbol indicates the concatenation of the two feature sets either side of the symbol. An example of a predictive model would be: $AA_{\text{BUNA790101}}$^*AAComp_PLS;* where $AA_{\text{BUNA790101}}$ shows the BUNA790101 AAI index (alpha-NH chemical shifts), *AAComp* is the amino acid composition descriptor, $AA_{\text{BUNA790101}}$^*AAComp* is the concatenation of the 2 feature sets and *PLS* is the chosen regression algorithm.

These three main strategies were used in building the models and evaluated and compared for their predictive performance. The main mode of evaluation was the R^2 score and RMSE (root-mean-squared-error) metrics, with the best models maximizing the former and minimizing the latter. The R^2 score (7), also called the coefficient of determination, is usually referred to as the square of the correlation coefficient (Pearson's r) between observed and predicted values in a regression and helps in determining the predictability of the models, ranging from 0 to 1, and is represented as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \quad (7)$$

where y_i is the measured activity of the i^{th} protein sequence, $\hat{y}_i$ is the model's predicted activity for the i^{th} sequence, $\bar{y}$ is the average activity and *N* is the number of sequences.

Next, the RMSE (8), also called the RMSD (root-mean-square-deviation), is a popular statistical metric for measuring the performance of an estimator and represents the standard deviation of the residuals (prediction errors), which themselves are a measure of how far the predicted values are from the regression line. RMSE is represented as:

$$RMSE = \sqrt{\sum_{i=1}^{N}\frac{(\hat{y}_i - y_i)^2}{N}} \quad (8)$$

where y_i is the measured activity of the i^{th} protein sequence, $\hat{y}_i$ is the model's predicted activity for the i^{th} sequence and N is the number of sequences.

Furthermore, other metrics were captured to additionally assist in assessing the predictability and performance of the models including the mean squared error (MSE), mean-absolute-error (MAE), ratio of performance to deviation (RPD) and explained variance regression score. The values for these extra metrics for each of the results below in this section can be found in the supplementary results tables S1 to S6.

As previously described, the dataset used to evaluate each of the aforementioned encoding strategies was a dataset where the desired protein activity/characteristic was in the maximization of the thermostability of a particular protein sequence [28]. Therefore, the results below show the effectiveness and predictability of the various models and strategies in predicting the thermostability value (T50) of previously unseen sequences. It should also be noted the stochastic nature of some of the algorithms and techniques used to get the results, therefore there may some variability if reproducing them. The full results files from each of the examined strategies can be found in the supplementary materials. Finally, each of the encoding strategies outlined below were executed using the *encoding* module within the *pySAR* software, which allows for the encoding of the protein sequences via all available feature sets as well as the building and evaluating of a predictive regression model.

A. Encoding Strategy: AAI Indices

Indices from the AAI were used to numerically encode each amino acid within the sequences according to various physiochemical properties, as discussed in the *Methods* section. Pre-processing steps of zero-padding and windowing preceded a Fourier Transform of the numerical sequences to transform the sequences into the frequency domain, generating various informational protein spectra, with the power spectra (6) being the primary focus for this study (as shown in the upper part of Figure 2b). With there being a total of 566 indices to date, and six regression algorithms investigated, this equated to 3396 total predictive models built and evaluated. The feature spaces for each of these models was in the form M × N, where M is the number of protein sequences and N is the length of the zero-padded protein sequences; for the thermostability dataset explored, this equated to 261 × 466, per predictive model.

Figures 3a and b show the distribution of the R2 and RMSE metric values, respectively, for each of the explored algorithms. The Ada, RF and Bag algorithms are clearly the best-performing techniques from the six investigated, all generally clumping around similar R2 and RMSE values, as well as possessing similar quartile distributions. Although Ada receives the best result for both metrics, receiving an R2 and RMSE of ~0.75 and ~2.94, respectively. The three algorithms dominate the top 500 out of 3396 results, with the scarce addition of a PLS or KNN. The poorest performing techniques were that of KNN and SVR, receiving a peak R2 of only ~0.65 and ~0.57, respectively. Comparatively, for KNN's and SVR's, the error also steeply increases, with the error of the best performing indices (in terms of R2) for the two techniques being ~3.39 and ~3.61, respectively. One can also note from Figures 3a and b that the samples for each of the algorithms are generally symmetric and normally distributed, with no clear positive or negative skew for either metric. This absence of a clear skew is evident with much of the mean R2 and RMSE's (white circle in boxplot) for each algorithm lying around the median. Another noticeable trait of the boxplots is the large range for some of the algorithms, represented by the max value

minus the minimum value, which is further shown in Table II with the R2 decreasing by 0.039 and RMSE increasing by 0.36, for the first 10 results. These may be relatively small incremental values, but they become noteworthy when one considers that this encoding strategy generated 3396 total models. The presence of this significant range prompts the hypothesis that some feature sets perform poorly, regardless of the algorithm implemented, emphasizing that the choice of feature set (indices in this case) may be more important than choice of algorithm – Hyp_1.

Table II highlights the top ten predictive models found for this encoding strategy. The best-performing indices were the PONJ960101 (Average volumes of residues), followed by the MEIH800101 (Average reduced distance for C-alpha) and ARGP820101 (Hydrophobicity index), which come under the categories of Geometry, Geometry and Hydrophobic, respectively. Looking at the Figures 3a and b as well as Table II, one may conclude that the choice of algorithm plays a significant part in the gathered results, but it is also useful to point out the importance of the encoding index as well. For example, the indices KARS160111, NAKH900112, KRIW790102, and many others all provided unremarkable results, regardless of algorithm, with a total of 18 models producing a negative R2 score, thus upholding the previous hypothesis Hyp_1.

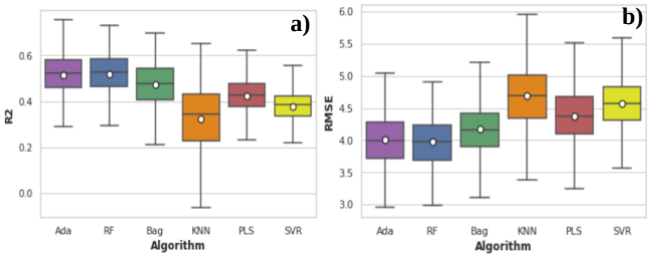

Fig. 3a, b. Boxplot of R2 and RMSE values for each algorithm, using encoding strategy *A(i)*. Note that any outliers have been removed. (mean of metric shown in white circle).

Overall, this strategy gives respectable R2 values, peaking at ~0.75, but with a quite considerable error rate (RMSE), emphasising that the strategy generates models with high predictability, in terms of predicting the activity value of the test sequences, but an associated high generalisation error, indicating that the models seem to perform well when testing in-sample but poorly out-of-sample and thus may be overfitting to the training dataset. The full results for all the captured metrics can be found in Supplementary Table S1.

TABLE II

BEST PERFORMING PREDICTIVE MODELS USING ENCODING STRATEGY A(I)

Predictive Model	Index Category	R2	RMSE
$AA_{PONJ960101}$_Ada	Geometry	0.749	2.938
$AA_{MEIH800101}$_Ada	Geometry	0.730	2.942
$AA_{ARGP820101}$_Ada	Hydrophobic	0.730	3.286
$AA_{RACS820109}$_Ada	Geometry	0.722	3.035
$AA_{QIAN880133}$_RF	Sec Struct[a]	0.720	3.039
$AA_{AURR980113}$_RF	Sec Struct[a]	0.716	3.241
$AA_{KIMC930101}$_RF	Sec Struct[a]	0.714	3.544
$AA_{PONP800101}$_Ada	Hydrophobic	0.714	3.544
$AA_{KARP850101}$_Ada	Flexibility	0.712	3.490
$AA_{GUYH850102}$_Ada	Hydrophobic	0.710	3.299

[a] Sec struct is the notation used herein to denote the secondary structure category for the AAI indices.

B. *Encoding Strategy: Descriptors*

The next encoding strategy that was evaluated was in the exclusive use of protein descriptors. These descriptors were outlined in the previous *Methods* section; all explored descriptors and their associated groups were listed in Table I. The three techniques for implementing the descriptors were in the building of the models using (*i*) a single descriptor, (*ii*) a combination of two descriptors and (*iii*) a combination of three descriptors. The output feature space for each of these will be in the form M × N, M × (N ^ O) and M × (N ^ O ^ P), respectively, where M is the number of protein sequences and N, O and P are the dimensionality of the descriptors, as citied in Table I. The '^' symbol indicates the concatenation of the descriptor on the left with the descriptor on the right.

Firstly for *(i)*, all 15 descriptors were individually used for the feature data in the training of the predictive models, generating 90 total models for the six algorithms. In comparison to the previous encoding strategy *A(i)*, there is a noticeable improvement of the results with an average increase of 0.0475 in R2 and a ~0.43 enhancement of the RMSE, from Table II to Table III, respectively. The best model was that of the DPComp descriptor with the RF algorithm, giving a peak R2 of ~0.80 and a predictive error (RMSE) of ~2.55, followed closely behind by a TPComp model, similarly using the RF algorithm.

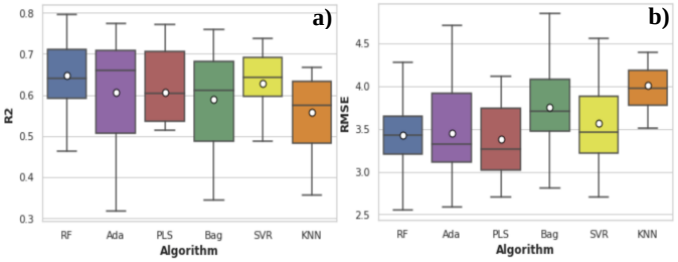

Fig. 4a, b. Boxplot of R2 and RMSE values for each algorithm, using encoding strategy *B(i)*.

Figures 4a and b visualize the distribution of the metric values for the encoding strategy. In comparison to the previous boxplots, there is a clear increase in the variability and spread of the data, identified by the enhanced size of the interquartile range (IQR) and the longer whiskers. For this encoding strategy, the RF algorithm performs best, achieving the highest R2 and lowest RMSE, although PLS, Bag and Ada equally feature more prominently in the best predictive models (Table III. Another thing to note is the skewness of the algorithms, with most algorithms, except RF, possessing a noticeable negative skew, shown by the mean (white circle) falling below the median. This generally means that most of the best-performing models, for each algorithm, perform higher than the average R2/RMSE for each. The cause of this skew can naturally be attributed to outliers that are pulling the mean down and up for R2 and RMSE, respectively. For example, each algorithm possessed at least one predictive model with an R2 <=0.46, each with an anomalously high RMSE. The effect of the outliers is showcased in the Supplementary Figures S2a and b, which shows the Figures 4a and b with the outliers included. Although, contrastingly the best-performing algorithm (RF) has a slight positive skew with the mean above the median, indicating a lesser number of outliers and better resultant R2

and error values. As hypothesized in the previous strategy (*Hyp₁)*, some features (descriptors) will perform poorly regardless of algorithm; for example, the SOCNum, T_CTD and NMBAuto descriptors all performed generally poorly regardless of algorithm, achieving average R2 scores of ~0.51, ~0.49 and ~0.46, respectively, thus again agreeing with hypothesis *Hyp₁*.

Next, Table III displays the top ten best-performing models obtained from the 90 explored. Composition-related descriptors feature in five out of the best ten models, with TPComp featuring in four of these, which achieves an average R2 and RMSE of ~0.73 and ~3.14, respectively, across the six algorithms. Close behind the two descriptors was GAuto, CTriad and MAuto, achieving average R2 scores of ~0.63, ~0.68, ~0.71, respectively. The previously mentioned variability of the data, visualized by the size of the IQR for some of the models, is emphasized when only considering the top ten results shown in Table III; from the 1st to 10th best-performing model, the R2 decreases by ~0.05 and the RMSE generally increases by ~0.80. Overall, the average R2 and RMSE for the 90 models was calculated as ~0.61 and ~3.60, respectively. The full results for all the captured metrics can be found in Supplementary Table S2.

TABLE III

BEST PERFORMING PREDICTIVE MODELS USING ENCODING STRATEGY B(I)

Predictive Model	Descriptor Group	R2	RMSE
DPComp_RF	Composition	0.796	2.547
TPComp_RF	Composition	0.789	2.633
TPComp_Ada	Composition	0.774	2.587
GAuto_PLS	Autocorrelation	0.773	2.749
TPComp_PLS	Composition	0.772	2.702
CTriad_PLS	Conjoint Triad	0.771	2.914
MAuto_Bag	Autocorrelation	0.760	3.005
MAuto_Ada	Autocorrelation	0.758	2.932
CTriad_Ada	Conjoint Triad	0.754	2.599
TPComp_Bag	Composition	0.745	3.372

The next encoding strategy solely using descriptors was *(ii)*, whereby all combinations of two descriptors were used in the building of the models, generating a total of 630 models. There is a further noticeable improvement in both metrics, in comparison to the previous descriptor strategy *B(i)*, with R2 and RMSE improving on average by ~0.07 and ~0.42 from Table III and Table IV , respectively. Compared to the previous descriptor-based strategy, RF has been overtaken by PLS which seems to be the best-performing algorithm for this strategy, which is indicated when looking at the top models shown in Table IV. Expanding to the top 100 results, PLS features in 40% of these, compared to the next best RF featuring in 21%. Overall, the average R2 and RMSE for all 630 models was ~0.654 and ~3.363, respectively, an increase in R2 of ~0.04 and a similar RMSE to that of strategy *B(i)*. As expected, PLS received the highest average metric values of ~0.68 and ~3.154, with KNN getting the weakest averages of ~0.60 and ~3.59 for the two metrics, respectively.

Boxplots of the metrics are shown in Figures 5a and b. A clear distinction from the same visualization in strategy *B(i)* (Figures 4a and b) is the reduced spread and variability, indicated by the smaller IQR, but there is still an observed skew

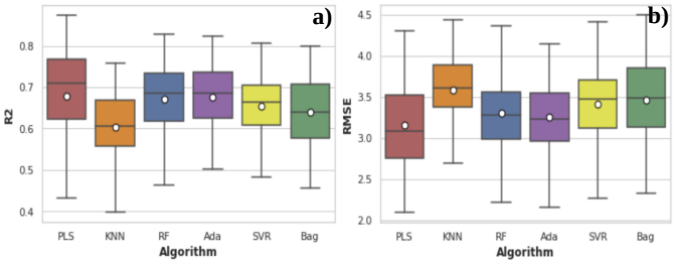

Fig. 5a, b. Boxplot of R2 and RMSE values for each algorithm, using encoding strategy *B(ii)*.

in the metric values, with the mean again falling slightly below the median for each algorithm, excluding Bag. Like the previous strategy, this slight skew indicates that most of the models are possessing above average R2 and RMSE values, but there are still a few samples that bring the average down. Looking at the results for all models, it can be observed that each algorithm has its own poorer performing models that overall affect the mean for the two metrics; for example, in relation to R2, KNN has eight models that have a value <=0.45, Bag has three models below <=0.45, and RF has two models with a value <=0.47. It is clear from the boxplots that the PLS algorithm has the largest range, but this is explained with there being exactly one predictive model with a negative R2 score, created from a model using PLS (*D_CTD^SOCN_PLS)*. This model possesses the only negative R2 score in all the results, which contrasts to strategy *A(i)* which featured 18 models with a negative R2. Although, the range and spread of the results should be somewhat expected due to the increased total number of models produced from the strategy and it further emphasizes what has been hypothesized thus far, that some feature sets (combinations of descriptors) perform poorly regardless of algorithm (*Hyp₁)*.

TABLE IV

BEST PERFORMING PREDICTIVE MODELS USING ENCODING STRATEGY B(II)

Predictive Model	Descriptor Groups	R2	RMSE
TPComp^T_CTD_PLS	Composition CTD	0.874	2.092
DPComp^QSOrder_PLS	Composition Quasi-sequence-order	0.858	2.391
AAComp^C_CTD_KNN	Composition CTD	0.856	2.419
TPComp^CTD_PLS	Composition CTD	0.854	2.481
TPComp^APAAComp_PLS	Composition Pseudo-composition	0.837	2.454
CTriad^PAAComp_PLS	Conjoint Triad Pseudo-composition	0.832	2.428
TPComp^GAuto_RF	Composition Autocorrelation	0.828	2.220
MAuto^CTriad_PLS	Autocorrelation Conjoint Triad	0.820	2.610
TPComp^C_CTD_Ada	Composition CTD	0.824	2.333
CTriad^QSOrder_PLS	Conjoint Triad Quasi-sequence-order	0.8228	2.366

Table IV demonstrates the highest-performing models, their associated metric values, and the descriptor's associated

descriptor group. Composition-related descriptors lead the best models again, either when two descriptor type. The descriptors TPComp in concatenation with T_CTD produced the best model with an R2 and RMSE of ~0.86 and ~2.24, respectively, using the PLS algorithm. Close behind this model were ones which further featured composition and CTD or autocorrelation-related descriptors, although CTriad and QSOrder each make two appearances in the top models. The full results for all metrics captured for this strategy is viewable in Supplementary Table S3.

The final descriptor-exclusive encoding technique was in using three descriptors (*iii*). This produced 455 total combinations for each algorithm, thus making 2730 total predictive models. Table V visualizes the top ten best-performing predictive models, and in comparison to Table IV (strategy *B(ii)*), there is again a noteworthy enhancement of the R2, improving on average by ~0.12, with the RMSE average staying unchanged between the tables. There is also a noteworthy improvement of the overall average R2 and RMSE for the 2730 models, getting values of ~0.66 and ~3.31, respectively. This continual improvement in the sought metrics has coincided with the increased number of features (increased feature space) used in the models, thus one may hypothesis that using more features improves the predictability and reduces the generalization error of the models – Hyp_2.

The results, in terms of choice of descriptor, is similar to the previous strategies (*B(i)-(ii)*), whereby most of the composition-related descriptors generally feature in the top-performing models. Although, autocorrelation and CTD descriptor types have also increased in number, featuring in all the top models at least once. Expanding the table to account for the top 500 predictive models, composition, autocorrelation and CTD descriptor types feature at least once in all but 8 of the 500 models, highlighting their usefulness in concatenation with other, equally useful, descriptor types.

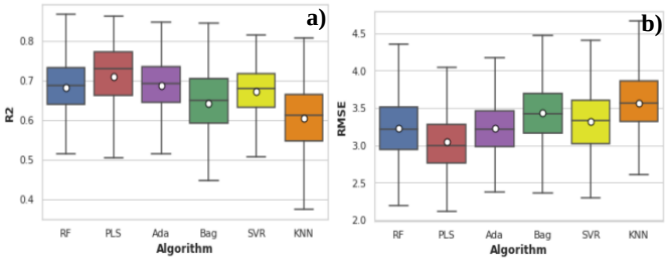

Fig. 6a, b. Boxplot of R2 and RMSE values for each algorithm using encoding strategy *B(iii)*.

The model with the highest R2 and lowest RMSE was a model built with the ensemble of the GAuto, CTriad and APAAComp descriptors, which equates to a feature space of N × 663, where N is the number of protein sequences in the dataset (N=261). Next, for the second and third models, the feature space is N × 890 and N × 8364, respectively. This further reiterates the previous hypothesis Hyp_2. Although, it's clear that choice of features can sometimes be more important than the total size of the feature space. This can be seen with models with smaller feature spaces getting slightly better results than ones using the TPComp descriptor, which itself has the largest feature space (N × 8000).

Likewise to strategies *B*(*i*) and *B*(*ii*), there is a continued prominence of the PLS algorithm in the top-performing results, featuring in all but six out of the top ten models. Considering the top 500 results out of the 2730 models, PLS features in ~40% of these. Contrastingly, the generally poorest algorithms SVR and KNN feature in only 10% and 4% of these top 500 results, respectively.

Figures 6a and b show the distribution of the R2 and RMSE results for this encoding strategy (*B*(*iii*)). The boxplots for all algorithms show slightly improved results in comparison to the previous two encoding strategies (*B(i)-(ii)*). All algorithms could be said to be fairly symmetric, having a more normal distribution, with a much lesser, although still present, negative skew, shown by the mean being much closer to the median. The range and or spread, when compared with strategy *B(ii)* is fairly consistent, but there is a noticeable change from strategy *B(i)*, indicating that, on average, the models are generating much greater predictability in the current strategy. Additionally, the presence of this large range advances hypothesis Hyp_1, with some feature sets (combinations of descriptors) performing poorly regardless of algorithm. Moreover, the full results for all the metric values for the predictive models can be found in Supplementary Table S4.

TABLE V

BEST PERFORMING PREDICTIVE MODELS USING ENCODING STRATEGY B(III)

Predictive Model	Descriptor Group	R2	RMSE
GAuto^CTriad ^APAAComp_RF	Autocorrelation Conjoint Triad Pseudo-composition	0.867	2.192
DPComp^CTD ^CTriad_PLS	Composition CTD Conjoint Triad	0.861	2.395
TPComp^C_CTD ^CTriad_PLS	Composition CTD Conjoint Triad	0.856	2.229
TPComp^GAuto ^QSOrder_PLS	Composition Autocorrelation Quasi-sequence-order	0.852	2.393
GAuto^T_CTD ^PAAComp_PLS	Autocorrelation CTD Pseudo-composition	0.852	2.521
GAuto^CTD ^T_CTD_PLS	Autocorrelation CTD CTD	0.850	2.109
AAComp^TPComp ^GAuto_PLS	Composition Composition Autocorrelation	0.847	2.699
DPComp^MAuto ^CTriad_Ada	Composition Autocorrelation Conjoint Triad	0.846	2.431
T_CTD^D_CTD ^CTriad_RF	CTD CTD Conjoint Triad	0.845	2.401
TPComp^D_CTD ^CTriad_Bag	Composition CTD Conjoint Triad	0.845	2.357

C. *Encoding Strategy: AAI Indices + Descriptors*

The final encoding strategy explored is an ensemble of the previously described two in *A* and *B*. Two main sub-strategies were explored, namely using one descriptor *C*(*i*), combining strategy *A(i)* + *B(i)*, and a combination of two descriptors *C*(*ii*), combining strategy *A(i)* + *B(ii)*. The former of these equated to a total of ~51,000 predictive models being built, with there being 566 AAI indices, 15 descriptors and 6 regression

algorithms. The latter of the two sub-strategies generated a total of ~356,000 models; similarly, there being 566 AAI indices, 6 regression algorithms, but a total of 105 combinations of two descriptors. Again, the former of these two strategies generates an output feature space of size M × (N ^ O), where M is the number of protein sequences and (N ^ O) is the concatenated features from the encoded AAI indices (N) and protein descriptors (O), N will be the length of the zero-padded protein sequences and O is the dimension of the descriptor as shown in Table I. For the latter strategy, the feature space is of size M × (N ^ O ^ P), where M, N and O are the same as before and P is the dimension of the 2nd descriptor used in the encoding, as cited in Table I. For the dataset and protein activity investigated (thermostability), M = 261 and N = 466.

Firstly, Table VI showcases the top ten results of the ~51,000 total models, in terms of R2, for this strategy. In comparison to Table II (*A(i)*) and Table III (*B(i)*), R2 sees an average increase of ~0.159 and ~0.11, respectively. Additionally, the predictive error (RMSE) decreases on average by ~1.19 and ~0.76 between the same tables. Overall, the average R2 and RMSE for the ~51,000 models were ~0.439 and ~4.297, respectively; a dramatic degradation of ~0.23 and ~0.99 for the two metrics, in comparison to the next highest average strategy (*B(iii)*). Although, the huge jump in the total number of produced models should be considered when interpreting these average metric values, with the number of models jumping from 2730 to ~51,000, from the previous strategy.

Composition-related descriptors seem to govern the majority of the top results again, featuring in six out of the ten in Table VI. Expanding the results table to the top 1000 will show that 50% of those are that of composition-related descriptors, mainly DPComp and TPComp, but also PSAAComp and AAComp. Contrastingly, looking into the worst-performing 1000 results, the NMBAuto is present in 342 out of these, followed by the C_CTD descriptor featuring in 192 and T_CTD appearing in 160. Another thing of note is none of these three aforementioned descriptors feature in any of the top 1000 results, indicating that they may be inherently poor choice of descriptor for this particular strategy and or set of protein sequences, further backing up the previously outlined Hyp_1.

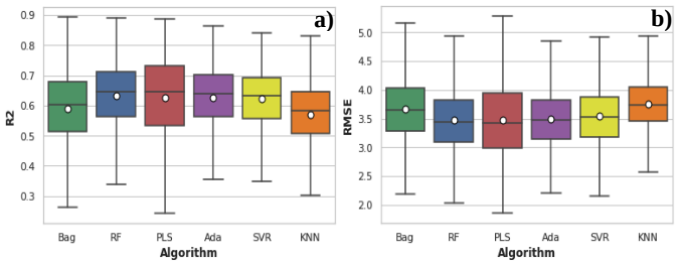

Fig. 7a, b. Boxplot of R2 and RMSE values for each algorithm, using encoding strategy *C(i)*.

Another noticeable trend from the top results is that of the algorithm. As has been the trend thus far, the PLS algorithm features in six of the top ten results, and likewise expanding this table to the top 1000 results, PLS finds itself in about ~50% of them. Although the Bag algorithm gives the model with the best predictive power (highest R2), Bag only features in about ~10% of the top 1000 models, with RF having a ~25% share, Ada with ~12% and KNN and SVR both only featuring in exactly 1%.

The individual algorithm boxplots in Figures 7a and b do not differ significantly, with each having a large range, similar spread of data and all possessing generally good and poor models. This alludes to hypothesis Hyp_1; for example, although the PLS, RF and Bag achieve peak R2 results of >=0.89, each still possesses a handful of models with negative R2 and an associated large RMSE. These generally poor models are most likely responsible for the negative skew, with the mean metric value of each algorithm falling below the median value. Although, the increased prevalence of outliers and a larger range should be expected due to the significant jump in the total number of models built using both *C* strategies, compared to those in *A* and *B*.

Overall, the best performing index was the *KIDA850101*-Hydrophobicity-related index – falling under the Hydrophobic category. Close behind this index was the *PALJ810114* – the normalized relative frequency of bend R – and the *YUTK870102* - Unfolding Gibbs energy in water, pH9.0 -, both coming under the secondary structure and observable category, respectively. Investigating these indices further it was found that they all perform generally well for each algorithm and descriptor, emphasizing they may be appropriate and useful indices to consider and further evaluate. Looking into the three top indices mentioned when they were individually used for encoding the protein sequences in strategy *A(i)*, they each receive R2 scores of ~0.68, ~0.70 and ~0.51, respectively, for the same Bag, RF and PLS algorithms listed in Table VI. Coincidentally, these three indices also feature in the models with the top attained R2 score from strategy *A(i)*. Thus, the three indices see a ~0.21, ~0.19 and ~0.377 improvement in their predictability when they are combined with a descriptor, in this case all three of the top models use the DPComp descriptor. This is the exact same situation for the remaining top ten results in Table VI, whereby the individual indices of the models generate respectable results from strategy *A(i)*, but ultimately, they all see a considerable improvement when concatenated with a particular descriptor. Correspondingly, the same indices see a reduction in their generalization error.

TABLE VI

BEST PERFORMING PREDICTIVE MODELS USING ENCODING STRATEGY C(I)

Predictive Model	Index Category	Descriptor Group	R2	RMSE
$AA_{KIDA850101}$^DPComp_Bag	Hydrophobic	Composition	0.894	1.973
$AA_{PALJ810114}$^DPComp_RF	Sec Struct	Composition	0.890	1.900
$AA_{YUTK870102}$^DPComp_PLS	Observable	Composition	0.887	2.038
$AA_{RICJ880108}$^CTriad_PLS	Sec Struct	Conjoint Triad	0.887	1.852
$AA_{NAKH920103}$^CTriad_RF	Composition	Conjoint Triad	0.885	1.882
$AA_{ROBB760111}$^DPComp_PLS	Sec Struct	Composition	0.876	2.051
$AA_{ISOY800108}$^CTriad_Bag	Sec Struct	Conjoint Triad	0.873	2.074
$AA_{AVBF000108}$^TPComp_PLS	Polar	Composition	0.873	2.198
$AA_{RADA880101}$^GAuto_PLS	Hydrophobic	Autocorrelation	0.870	2.456
$AA_{AURR980106}$^DPComp_PLS	Sec Struct	Composition	0.869	2.021

Furthermore, the most present descriptor types in the top results were those of DPComp and CTriad, five and three entries in Table VI, respectively. Upon investigating these two descriptors when they were individually used for the encoding (*B(i)*), they received peak R2's of ~0.80 and ~0.77 using the RF and PLS algorithms, respectively. Thus, this equates to a general improvement of ~0.094 and ~0.11 when AAI indices are added to the descriptor feature space, with the top predictive model ($AA_{KIDA850101}$^*DPComp*) having a total feature

dimensionality of 261 × 866 – further supporting *Hyp*$_2$. From this noticeable improvement upon the addition of AAI indices with the descriptors, it can be further hypothesized that AAI indices and composition-related descriptors perform well when used in isolation in the modelling phase, but the predictability of the models is greatly enhanced, and the generalization error reduced when they are used in concatenation with one another – *Hyp*$_3$. Moreover, the full results for Table VI, listing the values of all metrics captured, is viewable in Supplementary Table S5.

The next, and final strategy *C(ii)* involved an ensemble of strategy *A(i)* and *B(ii)* (AAI indices + two descriptors). With 566 indices in the AAI and 105 combinations of two descriptors, being evaluated on six types of regression algorithms, this equated to a total of over 356,000 models created and evaluated. Table VII shows the top ten results, in terms of R2, for this strategy. The results generally match the expectations gathered thus far and mentioned in the numerous hypotheses. Again, there is a general improvement of the predictability (R2) and error (RMSE), in comparison to previous strategies. Although, in relation to the previous strategy *C(i)*, the improvements are less significant, with an inconsequential average increase of ~0.016 in R2 and an average ~0.002 decrease in RMSE, from Table VI to Table VII. Correspondingly, when comparing to the constituent strategies that make up *C(ii) (A(i) and B(ii))*, R2 rises by ~0.15 and ~0.033, respectively, and RMSE reduces by ~0.868 and ~0.122, respectively, in terms of the top-performing predictive models. For example, the AAI index in the best-performing predictive model from Table VII ($AA_{RACS820114}$), received a peak R2 of ~0.59 in *A(i)*. Similarly, this models' constituent descriptors (*TPComp* and *D_CTD*) each received an R2 of ~0.79 and ~0.67 from *B(i)*, respectively. Thus, this shows that the index and composition-related descriptors perform well in isolation, but much better in concatenation, and this conclusion can be drawn for all the results in Table VII, with each possessing at least one composition descriptor – further supporting the previous *Hyp*$_3$ hypothesis

Hypothesis *Hyp*$_2$ is also furthered in this strategy. The feature space for the top three best-performing models equated to 261 × 8571, 261 × 8020, and 261 × 8343, respectively. This supports the *Hyp*$_2$ hypothesis that more features enhance the predictability of the models, although, the relatively small enhancements in the metrics from the previous strategies *C(i)* and *B(iii)* indicate that this only occurs up until a point, where additional features and or a larger feature space have no added effect on the predictability of the models. This was also highlighted with the insignificant improvement in the average metric values for *C(ii)*. In this dataset, it seems that the peak R2 tapers off at around 0.90 and the RMSE at around 1.80.

Figures 8a and b showcase a still ever-present large range of metric values for each algorithm. As noted in the previous strategy, this range is to be expected due to the exponential increase in the number of predictive models built, jumping from ~51,000 to ~356,000 from *C(i)* to *C(ii)*. The spread of the results still give a small, but present negative skew, with the means falling slightly below the median for each algorithm, suggesting the prevalence of outliers that are affecting the overall results still. The Bag algorithm is also present in the top ten models, achieving a peak R2 of ~0.896, followed by RF with a peak of ~0.891, Ada with ~0.872, KNN with ~0.853 and finally SVR with ~0.851. Overall, the average R2 and RMSE for the ~356,000 models were ~0.65 and ~3.38, respectively, further propping up *Hyp*$_2$.

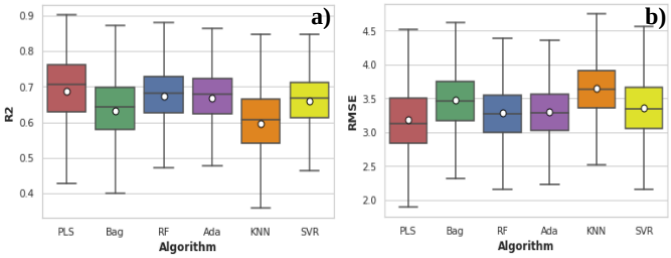

Fig. 8a, b. Boxplot of R2 and RMSE values for each algorithm, using encoding strategy *C(ii)*.

Furthermore, once again the PLS algorithm dominates the best models, featuring in ~69% of the top 5000 predictive models. Contrastingly, the SVR and KNN algorithms are generally the worst-performing, with SVR featuring in only about ~1% of these 5000 models and KNN in ~0.7%. Although, as has been mentioned much before and hypothesized in *Hyp*$_1$, each algorithm also possesses its own set of models with poor generalizability and associated error. The continued prominence of the PLS algorithm, with respect to the top-performing results, and building on hypotheses *Hyp*$_2$ and *Hyp*$_3$, a final hypothesis – *Hyp*$_4$ - can be proposed. This hypothesis states that ensemble-based algorithms (e.g., RF, Bag and Ada) are more appropriate for dense and smaller feature sets (feature spaces), and the PLS algorithm is more appropriate for more sparse feature sets. In the context of this work, this would mean that ensemble-based algorithms are more suitable for the earlier

TABLE VII

BEST PERFORMING PREDICTIVE MODELS USING ENCODING STRATEGY C(II)

Predictive Model	Index Category	Descriptor Group	R2	RMSE
$AA_{RACS820114}$^TPComp ^D_CTD_PLS	Geometry	Composition CTD	0.903	2.070
$AA_{JANJ790101}$^AAComp ^TPComp_PLS	Hydrophobic	Composition Composition	0.899	1.808
$AA_{WILM950103}$^TPComp ^CTriad_PLS	Hydrophobic	Composition Conjoint Triad	0.898	2.089
$AA_{CHAM820101}$^AAComp ^DPComp_PLS	Polar	Composition Composition	0.897	2.137
$AA_{QIAN880110}$^TPComp ^QSOrder_PLS	Sec Struct	Composition Quasi-sequence-order	0.897	2.072
$AA_{TANS770110}$^DPComp ^CTriad_Bag	Sec Struct	Composition Conjoint Triad	0.896	2.048
$AA_{QIAN880120}$^TPComp ^C_CTD_PLS	Sec Struct	Composition CTD	0.894	2.013
$AA_{OLSK800101}$^DPComp ^T_CTD_PLS	Geometry	Composition CTD	0.894	2.053
$AA_{GEIM800103}$^DPComp ^PAAComp_PLS	Sec Struct	Composition Pseudo-Composition	0.893	2.022
$AA_{LIFS790101}$^AAComp ^TPComp_PLS	Sec Struct	Composition Composition	0.893	2.113

strategies (*A(i)* and *B(i))* and PLS is better suited to the latter strategies (*B(ii), B(iii), C(i)* and *C(ii))*. Finally, the results for

all the captured metrics for this strategy can be seen in Supplementary Table S6.

V. Discussion

Screening of all possible mutant combinations of protein sequences is an impossibility due to the massively high-dimensional nature of the problem, thus data-driven ML techniques are vital in the extrapolation of useful features and information to guide the process and ultimately better understand the relationships between a protein sequence and its associated activity. Arguably, one of the most important stages in this is how to encode the sequences, and specifically its constituent amino acids. This encoding is not a one-size-fits-all situation, whereby one strategy for one mutant library may not be applicable or useful for another, thus there is an element of trial and error required to find the optimal encoding technique. The software created that accompanies this work (*pySAR*) alleviates much of that trial-and-error process, allowing for a verbose evaluation of a plethora of encoding techniques. Three of the main strategies focused on in this work were that of encoding via indices from the AAI (*A(i)*), protein descriptors (*B(i)-(iii)*) as well as a combination of the two (*C(i)-(ii)*). Throughout the results section, various trends were noticed that prompted the postulation of four main hypotheses; each of which are discussed further below.

A. Hypothesis 1: "Some feature sets perform poorly, regardless of algorithm implemented."

Numerous examples were described throughout each strategy that emphasize this hypothesis. For example, in strategy *A(i)*, the Ada, RF and Bag algorithms performed best, featuring in 86% of the top 1000 results, but there were some observed indices such as NISK860101, KARS160102 and LEVM780105, along with many others, receiving poor results regardless of chosen algorithm, possessing an average peak R2 of <=0.55. Also, in strategy *B(i) – B(iii)*, the C_CTD, T_CTD, SOCNum and NMBAuto descriptors generally (with some exceptions) performed lesser, in terms of model predictability, in comparison to some other descriptors, either when the descriptor was used in isolation (*B(i)*) or in concatenation with other descriptors (*B(ii)-(iii)*). Finally, in *C(i)* and *C(ii)* there were similarly obtained results with the same descriptors as in *B* generally underperforming, including the NMBAuto, C_CTD, T_CTD and SOCNum, which featured in the bottom tier of results, irrespective of the algorithm implemented.

A noticeable pattern for these descriptors is their associated feature space, with the CTD descriptors having a feature space of 21 features, NMBAuto possessing 240 and SOCNum having 60 features. This contrasts to some of the composition-related descriptors which possess 400 and 8000 features (DPComp and TPComp) and CTriad having 343 features, alluding to *Hyp*$_2$ that more features improve the predictability and performance of the models. Overall, the results from the three strategies highlight the capped effect that the choice of algorithm can have on the models: no matter how high the generalizability and low error that an algorithm may generate on average, if a poor choice is made regarding the feature set (AAI index and or protein descriptor), then the model will perform poorly, regardless. Although, it is difficult to know a priori how well a particular descriptor or feature set will perform in encoding a set of protein sequences in the mapping of an SAR. As has been mentioned, a particular encoding strategy and predictive model for one dataset may be completely inoperable and inconsequential for another dataset, owing to the complexities and variabilities of protein sequences and their constituent amino acids between datasets. Therefore, one of the primary aims of *pySAR* was to provide an interface that makes more efficient the process of selecting a particular encoding strategy with associated feature set and algorithm, in the building of predictive models to analysis SAR's.

Further investigation into the reason behind some feature sets performing poorly should be confirmed by exploring the methodology on more diverse datasets with different sought-after activity values, as well as reading deeper into the literature of each AAI index and or protein descriptor. This would also assist in confirming if the underachieving feature sets are exclusive to the specific sequence-activity dataset used in this work (thermostability), or are comprehensively poor features to use, regardless of dataset.

B. Hypothesis 2: "The use of more features (larger feature sets/spaces) improves the predictability of the predictive models and reduces their generalization error. "

Several examples were cited from strategy *A(i)* to *C(ii)* that back up this hypothesis. The feature space of strategy *A* was 261 × 466 (466 being the length of the zero-padded protein sequences). Next, the feature spaces of strategy *B* varied from 261 × 20 (AAComp – *B(i)*) to 261 × 8743 (TPComp ^ DPComp ^ CTriad – *B(iii)*) and finally strategy *C* had a range of feature spaces from 261 × 486 (AA_x ^ AAComp - *C(i)*) to 261 × 8866 (AA_x ^ DPComp ^ TPComp – *C(ii)*). It can be observed that the upper end of the feature spaces steadily increases from strategy to strategy. Coinciding with this increase in the number of features is the general increase in the predictability of the models, as well as the reduction in generalization error, with the top-performing model in *A(i)* achieving an R2 and RMSE of ~0.75 and ~2.94, respectively, compared to *C(ii)* with its best-performing model achieving an R2 of ~0.90 and RMSE of ~2.

It should be noted that some feature sets overshadow others in terms of size, for example TPComp, DPComp and CTriad each possess 8000, 400 and 343 features, respectively, compared to, for example C_CTD, PAAComp or SOCNum which each only have 21, 50 and 60 total features, respectively. This is reflected in many of the results from Tables 4 to 8, whereby the top-performing results contain descriptors with relatively high feature spaces, mainly TPComp, DPComp and or CTriad, with infrequent appearances for descriptors with comparatively smaller feature spaces.

Moreover, one may potentially conclude that more features seem a rational approach to improving the predictability of the results, but with more features and a higher dimensionality comes the curse of dimensionality and potential overfitting. More features may ultimately lead to redundant information being passed into the models, reducing the signal to noise ratio such that the model is learning the noise, obscuring some of the relevant and important information from the sequences [33]. Hence, feature selection should be applied in the aim of finding the optimal set of features that provide the highest prediction accuracy and lowest error, as is discussed in *Hyp*$_4$. The curse of

dimensionality not only relates to the number of features and or feature space but also the size of the training dataset. A significant downfall of many ML-based SAR or protein engineering tasks is the lack of available data and the expense of obtaining such data in vitro, which creates a low-signal to noise ratio for the ML models. Thus, it's vital that focus remains on generating larger datasets through in silico ML-based screening methods, and that these such methods are continually improved, ensuring that more features are used in the building of the data-driven ML models alongside larger datasets, mitigating the curse of dimensionality and potential overfitting.

C. Hypothesis 3: "AAI indices and composition-related descriptors perform well when used in isolation in the building of the predictive models, but the predictability is greatly enhanced when they're used in concatenation with one another."

As noted in the results for strategies *C(i)* and *C(ii),* the feature sets that made up the predictive models each performed respectably well individually, but they all gave enhanced performance when used in concatenation. Combining multiple descriptor types into one feature set has shown to improve predictive performance, compared to the use of individual descriptors [25]. Specifically, it was observed that descriptors that fall under the group of composition gave the best results when combined with the protein spectra of indices from the AAI. This alludes to the previous hypothesis Hyp_2, in that more features gave better results, which is emphasized with the composition-related descriptors DPComp and TPComp having 400 and 8000 features, respectively. Although, when comparing the individual encoding of the descriptors to the individual encoding of the AAI indices, the latter sees the biggest jump in performance, suggesting that descriptors have a higher weighting on the overall predictability and performance of the models.

The most featured composition-related descriptors were the DPComp and TPComp. A benefit of these two types of composition descriptors compared to AAComp and or PAAComp is that sequence order information is captured, where the local neighborhood of amino acids is taken into account [25]. AAComp only calculates the proportion of residues in a sequence, but not their arrangement/order and PAAComp builds on AAComp, but also contains the distribution of hydrophobic and hydrophilic amino acids along the sequence. A benefit of the composition-related descriptors (excluding PAAComp) is the ease-of-use and understandability, with the only requirement being the explicit protein sequence and no necessity for physiochemical properties or structural information.

Contrastingly, this inattention to the physiochemical properties can also be considered as a negative for the descriptor. Although, this is eventually mitigated with the presence of the encoded protein spectra via the AAI indices, built using physiochemical properties. Thus, strategy *C* captures important compositional, structural, and sequence-order information using the descriptors as well as physiochemical/biochemical properties via the protein spectra.

The preferred type of AAI index to use for the studied SAR dataset seemed to be structure-related indices, falling under the secondary structure and or geometry category. This emphasizes the utility of structure-related properties in the encoding process, proving suitable for building an SAR due to the problem itself essentially being a protein function prediction problem, with the function of a protein being directly dependent on its structure, which itself is dependent on its initial sequence [1].

Moreover, an observed side-effect when evaluating both AAI indices and descriptors in strategy *C* was the greater verbosity of results. This inevitably creates the difficulty of interpreting such a large number of results. For example, encoding strategy *A(i)* generated a total of 566 models, compared to strategy *C(i),* which generated ~51,000. Although, a useful approach for interpreting the latter results was via a somewhat process of elimination. For example, if one was to set a threshold R2 score of >=0.5 and discard all those models below this, this would remove ~9,000 models. If one went further and set a threshold in terms of RMSE of <2.5, then that would further eliminate 42,000 models, leaving just 647 sets of results. Thus, the number of total models in the results has been reduced by over 50,000 by just setting two thresholds using two metrics. This process could be continued further to repeatedly eliminate result sets, eventually obtaining a much smaller and manageable subset of predictive models.

D. Hypothesis 4: "Ensemble-based algorithms are more appropriate for dense, smaller feature sets (feature spaces) and the PLS algorithm is more appropriate for more sparse feature sets."

From most of the results gathered from strategies *A* and *B*, there was a clear distinction in what the 'best-performing' algorithms were, with the RF, Ada and Bag observably outperforming the others. The most important thing that these three algorithms share in common is that they are all ensemble methods. An ensemble consists of a collection of individual estimators that work together and combine into one predictive model. In these algorithms, the default individual estimators are made up of decision trees. A benefit of ensemble learning is the generally improved predictive performance and generalization of the models, being more robust in reducing the variance and bias by averaging over many different predictors [34]. Ensemble-based methods feature more prominently in the top models for strategies *A(i)* and *B(i)-(ii).* These cited strategies possess the densest feature data out of all those explored, with some of the other strategies involving concatenation of large feature sets that lead to a much higher dimensional and sparse feature space, the effect of which was discussed in Hyp_2.

Due to the associated increase in feature space sparsity that occurs with inclusion of more features in the predictive models, the prominence of ensemble-based models decreases in the topmost attained results, being overtaken generally by the PLS algorithm. The PLS algorithm, is similar to PCR (principal components regression) and has shown to be particularly useful for data where, for the predictors, there are more variables than observations [35]. The default number of components used in the predictive models in this work was two, meaning that the sparse X and Y spaces were projected into a 2-dimensional feature space, capturing the most important information from the high-dimensional feature space. The algorithm has shown its utility in several ML, protein engineering use-cases,

including in the analysis of the same thermostability dataset studied in this work [8], as well as in the prediction of the enantioselectivity of protein sequences [7]. In the realm of sequence-activity correlation, PLS provides more predictive accuracy and lower risk of chance correlation, but at the expense of possibly overlooking 'real' correlations and a high sensitivity to the relative scaling of descriptor variables [35].

Therefore, one can postulate that ensemble-based methods are more appropriate for dense feature matrices (strategies *A(i)*, *B(i)*) and PLS is better for more sparse data (strategies *B(ii)-B(iii)*, *C(i)-C(iii))*. Although this should be further confirmed with a different set of base estimators. As mentioned, the base estimator for each of the ensemble algorithms explored was the decision tree, thus the encoding strategies may only perform well with this specific type of estimator. Moving forward, in place of a decision tree, an ensemble of SVR's, KNN's or PLS's could be evaluated and tuned to find their optimal number/depth and associated parameters.

For the sparser feature matrices generated in the latter encoding strategies, a procedure of dimensionality reduction prior to the predictive models could be considered, mimicking the general working of PLS, where the feature space is dimensionally reduced prior to a regression being completed. This procedure may increase the predictive performance and generalizability in the ensemble-based models for the sparser feature matrices, allowing for the latter encoding strategies to exploit the benefits of ensemble learning, whilst also mitigating some of the effects from the curse of dimensionality, as discussed in *Hyp*$_2$. Although, with the inclusion of ensemble-based algorithms comes the enhanced computational requirement and time, that needs to be considered.

TABLE VIII

TOTAL EXECUTION TIME (IN MINUTES) OF EACH ENCODING STRATEGY, FOR EACH ALGORITHM

	A(i)	*B(i)*	*B(ii)*	*B(iii)*	*C(i)*	*C(ii)*
PLS	4.585	0.036	0.092	0.411	9.348	41.412
KNN	4.239	0.038	0.140	0.754	9.716	68.486
SVR	4.524	0.048	0.278	1.606	15.667	145.179
Bag	7.411	0.061	0.445	2.615	23.017	240.080
Ada	13.947	0.111	1.062	6.562	52.449	588.074
RF	35.531	0.284	3.327	21.059	154.424	1900.062

Table VIII shows the total execution time (in minutes) of each algorithm, in ascending order, for all the explored encoding strategies in this work. As one would expect, with increasing number of features and dimensions in the feature space (from strategy *A(i)* - *C(ii)*), the computational time significantly increases. The fastest strategy is clearly *B(i)*, achieving a highest execution time of ~17 seconds, followed by strategies *B(ii)*, *B(iii)* and *A(i)*, seeing steady increases in the longest execution time of ~3.3 minutes, ~21 minutes and ~35 minutes, respectively. Each of these peak times were acquired from the RF algorithm, with none of the other algorithms edging near the RF's execution times.

Moreover, a significant and exponential climb in execution time is observed for both the *C(i)* and *C(ii)* strategies. *C(i)* sees a peak of ~154 minutes (2 ½ hours), an exponential increase of ~119 minutes compared to the previous longest strategy (*A(i)*) and *C(ii)* acquires a peak time of ~1900 minutes (~31.5 hours), a substantially further exponential increase of 1746 minutes, in comparison to *C(i)*. Similarly, each of these execution times were attained when using the RF algorithm.

Contrastingly, the PLS algorithm clearly outperforms all the other algorithms, achieving the fastest execution time for each strategy, closely followed by KNN then SVR. This emphasises the disparity between the ensemble models (RF, Ada, Bag) and the individual estimators (PLS, SVR, KNN), which can be clearly explained by the enhanced computational time complexity of ensemble methods, brought upon by them having to build many estimators to average over. Therefore, although ensembles may offer slightly better performance in some strategies, their associated computational expense cannot be overlooked.

E. Future work

There remains ample opportunity for further development and improvement on the outlined techniques mentioned throughout this work. One of the primary focuses is on the utilization of the methodology with additional datasets. This would ideally help in confirming if some of the prior conclusions and hypotheses are specific to the dataset and protein activity explored (thermostability) or are general to any dataset and activity. Another additional focus for future work is that of diving deeper into the potential of DSP techniques and representing sequences via protein spectra. In this work, the power spectrum (6) was the sole type of spectrum evaluated. Multiple different variations of protein spectra are available such as the absolute, imaginary, and real, but were not individually evaluated due to the complexity of generating all the results again for each spectrum type. There exists other DSP-related functions and techniques that could also be implemented and evaluated, including the use of different window functions, filtering, and convolutions. The accompanying software, *pySAR*, has existing functionality that easily allows each of these techniques to be realized, including offering a variety of potential window and filtering functions.

There is also scope to explore different types of available regression algorithms, with a focus on using different base estimators in the ensemble methods, as discussed in *Hyp*$_4$. Alongside evaluating more types of algorithms, hyperparameter tuning should also be undertaken for each model to find the optimal arrangement of parameters, in the aim of further improving the predictability of the models. Once again, the accompanying software already possesses the existing functionality to achieve all of this via its *Model* module. Finally, a further focus is on the continued expansion in functionality of the software, in an aim to make it publicly available in an easy-to-use web app.

VI. CONCLUSION

The utility of physiochemical and structural protein descriptors in combination with informational protein spectra seems to be a rational approach in the building of predictive ML models to map the relationship between protein sequences and

their associated activity value of thermostability. Each of the encoding strategies discussed give a clear indication of what sets of AAI indices from the AAI database [26] and or protein descriptors maximize the predictability of a particular predictive regression model. The process of finding this optimal set of features and algorithm for a predictive model can be an exhaustive a priori process of trial and error, but the accompanying software for this work (*pySAR)* aims to alleviate much of that effort, allowing the user to encode their protein sequence dataset in hundreds of thousands of possible ways. A total of ~410,000 models were built and evaluated in this work, but the software already has existing functionalities that provide ample room for future work and opportunities, as outlined in the *Future Work* section. The primary aim moving forward remains that of evaluating the software on different types of datasets with a range of activity characteristics, as well as utilizing the software for other protein engineering tasks.

VII. Supplementary Materials

All the required data files used in this work, including the dataset, AAI database and the pre-calculated descriptor values can be found within the code repository at:
https://github.com/amckenna41/pySAR/tree/main/pySAR/data
Additionally, all the results from each of the studied encoding strategies, as well as a pdf of the supplementary tables and figures, is also available on the repository:
https://github.com/amckenna41/pySAR/tree/main/Results

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, Analyzing protein structure and function. London, England: Garland Science, 2002.

[2] Y. Zhou, Y. Duan, Y. Yang, E. Faraggi, and H. Lei, "Trends in template/fragment-free protein structure prediction," *Theor. Chem. Acc.*, vol. 128, no. 1, pp. 3–16, 2011.

[3] M. K. M. Engqvist and K. S. Rabe, "Applications of protein engineering and directed evolution in plant research," *Plant Physiol.*, vol. 179, no. 3, pp. 907–917, 2019.

[4] S. Lutz, "Beyond directed evolution--semi-rational protein engineering and design," *Curr. Opin. Biotechnol.*, vol. 21, no. 6, pp. 734–743, 2010.

[5] T. Shafee, "Evolvability of a viral protease: experimental evolution of catalysis, robustness and specificity." Apollo - University of Cambridge Repository, 04-Feb-2014.

[6] M. H. Barley, N. J. Turner, and R. Goodacre, "Improved descriptors for the quantitative structure–activity relationship modeling of peptides and proteins," *J. Chem. Inf. Model.*, vol. 58, no. 2, pp. 234–243, 2018.

[7] N. T. Fontaine, X. F. Cadet, and I. Vetrivel, "Novel descriptors and digital signal processing- based method for protein sequence activity relationship study," Int. J. Mol. Sci., vol. 20, no. 22, p. 5640, 2019.

[8] K. K. Yang, Z. Wu, and F. H. Arnold, "Machine-learning-guided directed evolution for protein engineering," *Nat. Methods*, vol. 16, no. 8, pp. 687–694, 2019.

[9] D. M. Mason *et al.*, "Deep learning enables therapeutic antibody optimization in mammalian cells by deciphering high-dimensional protein sequence space," *bioRxiv*, p. 617860, 2019.

[10] Y. Xu *et al.*, "Deep dive into machine learning models for protein engineering," *J. Chem. Inf. Model.*, vol. 60, no. 6, pp. 2773–2790, 2020.

[11] D. Medina-Ortiz *et al.*, "Combination of digital signal processing and assembled predictive models facilitates the rational design of proteins," *arXiv [cs.CE]*, 2020.

[12] Z. Wu, S. B. J. Kan, R. D. Lewis, B. J. Wittmann, and F. H. Arnold, "Machine learning-assisted directed protein evolution with combinatorial libraries," Proc. Natl. Acad. Sci. U. S. A., vol. 116, no. 18, pp. 8852–8858,2.

[13] F. Cadet *et al.*, "A machine learning approach for reliable prediction of amino acid interactions and its application in the directed evolution of enantioselective enzymes," *Sci. Rep.*, vol. 8, no. 1, p. 16757, 2018.

[14] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[15] K. K. Yang, Z. Wu, C. N. Bedbrook, and F. H. Arnold, "Learned protein embeddings for machine learning," *Bioinformatics*, vol. 34, no. 15, pp. 2642–2648, 2018.

[16] S. Sinai, E. Kelsic, G. M. Church, and M. A. Nowak, "Variational auto-encoding of protein sequences," *arXiv [q-bio.QM]*, 2017.

[17] Smith, S. W. (1997). The scientist and engineer's guide to digital signal processing. San Diego, Calif: California Technical Pub.

[18] G. S. Randhawa, K. A. Hill, and L. Kari, "ML-DSP: Machine Learning with Digital Signal Processing for ultrafast, accurate, and scalable genome classification at all taxonomic levels," *BMC Genomics*, vol. 20, no. 1, p. 267, 2019.

[19] D. Mitra and M. Smith, "Digital signal processing in predicting secondary structures of proteins," in *Innovations in Applied Artificial Intelligence*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 40–49.

[20] Nwankwo, Norbert & Seker, Huseyin. (2011). Digital Signal Processing Techniques: Calculating Biological Functionalities. Journal of Proteomics & Bioinformatics. 4. 260-268. 10.4172/jpb.1000199.

[21] V. Veljković, I. Cosić, B. Dimitrijević, and D. Lalović, "Is it possible to analyze DNA and protein sequences by the methods of digital signal processing?," *IEEE Trans. Biomed. Eng.*, vol. 32, no. 5, pp. 337–341, 1985.

[22] F. Cadet *et al.*, "Application of fourier transform and proteochemometrics principles to protein engineering," *BMC Bioinformatics*, vol. 19, no. 1, p. 382, 2018.

[23] 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.*, vol. 31, no. 13, pp. 3692–3697, 2003.

[24] K.-C. Chou and Y.-D. Cai, "Predicting protein-protein interactions from sequences in a hybridization space," *J. Proteome Res.*, vol. 5, no. 2, pp. 316–322, 2006.

[25] S. A. K. Ong, H. H. Lin, Y. Z. Chen, Z. R. Li, and Z. Cao, "Efficacy of different protein descriptors in predicting protein functional families," *BMC Bioinformatics*, vol. 8, p. 300, 2007.

[26] S. Kawashima and M. Kanehisa, "AAindex: amino acid index database," *Nucleic Acids Res.*, vol. 28, no. 1, p. 374, 2000.

[27] J. Dong *et al.*, "PyBioMed: a python library for various molecular representations of chemicals, proteins and DNAs and their interactions," *J. Cheminform.*, vol. 10, no. 1, 2018.

[28] Y. Li, D. A. Drummond, A. M. Sawayama, C. D. Snow, J. D. Bloom, and F. H. Arnold, "A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments," *Nat. Biotechnol.*, vol. 25, no. 9, pp. 1051–1056, 2007.

[29] C. Chrysostomou, H. Seker, and N. Aydin, "Effects of windowing and zero-padding on Complex Resonant Recognition Model for protein sequence analysis," *Annu Int Conf IEEE Eng Med Biol Soc*, vol. 2011, pp. 4955–4958, 2011.

[30] M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the history of the fast Fourier transform," *Arch. Hist. Exact Sci.*, vol. 34, no. 3, pp. 265–277, 1985.

[31] R. B. Blackman and J. W. Tukey, "The measurement of power spectra from the point of view of communications engineering — Part I," Bell Syst. tech. j., vol. 37, no. 1, pp. 185–282, 1958.

[32] J. Guo, Y. Lin, and X. Liu, "GNBSL: a new integrative system to predict the subcellular location for Gram-negative bacteria proteins," *Proteomics*, vol. 6, no. 19, pp. 5099–5105, 2006.

[33] R. Liu and D. F. Gillies, "Overfitting in linear feature extraction for classification of high-dimensional image data," *Pattern Recognit.*, vol. 53, pp. 73–86, 2016.

[34] C. Zhang and Y. Ma, Eds., *Ensemble Machine Learning: Methods and Applications*, 2012th ed. New York, NY: Springer, 2012.

[35] R. D. Cramer III, "Partial Least Squares (PLS): Its strengths and limitations," *Perspect. Drug Discov. Des.*, vol. 1, no. 2, pp. 269–278, 1993.