Ensemble Voting Schemes that Improve Machine Learning Models for Predicting the Effects of Protein Mutations

Sarah Gunderson
Western Washington University
516 High Street
Bellingham, WA 98225
gunders7@wwu.edu

Filip Jagodzinski
Western Washington University
516 High Street
Bellingham, WA 98225
filip.jagodzinski@wwu.edu

ABSTRACT

Understanding how a mutation affects a protein's structural stability can guide pharmaceutical drug design initiatives that aim to engineer medicines for combating a variety of diseases. Conducting wet-lab mutagenesis experiments in physical proteins can provide precise insights about the role of a residue in maintaining a protein's stability, but such experiments are time and cost intensive. Computational methods for modeling and predicting the effects of mutations are available, with several Machine Learning approaches achieving good predictions. However, most such methods, including ensemble based approaches that are based on multiple classifier models instead of a single-expert system, are dependent on large datasets for training the model. In this work, we motivate and demonstrate the utility of several voting-based models that rely on the predictions of a Support Vector (SVR), Random Forest (RF), and Deep Neural Network (DNN) models for inferring the effects of single amino acid substitutions. The three models rely on rigidity analysis results for a dataset of proteins, for which we use wet lab experimental data to show prediction accuracies with Pearson Correlation values of 0.76. We show that our voting approaches achieve a higher Pearson Correlation, as well as a lower RMSE score, than any of the SVR, RF, and DNN models alone.

CCS CONCEPTS

•**Applied computing** → **Molecular structural biology;** *Bioinformatics;* •**Computing methodologies** → *Machine learning;*

KEYWORDS

Voting, Machine Learning; Protein Structure; Support Vector Regression; Random Forest; Deep Neural Network; Rigidity Analysis

ACM Reference format:
Sarah Gunderson and Filip Jagodzinski. 2018. Ensemble Voting Schemes that Improve Machine Learning Models for Predicting the Effects of Protein Mutations. In *Proceedings of ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, Washington DC, USA, August 29 - September 1 2018 (ACM-BCB'18),* 9 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

ACM-BCB'18, Washington DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The amino acid sequence of a protein determines its structure and, as a result, its function. A single amino acid substitution can alter a protein's shape, which can be the cause of a debilitating disease. For example, mutations of α-galactosidase cause Fabry Disease, which results in kidney and cardiac complications [19].

Wet-lab mutagenesis experiments for inferring the role of an amino acid are conducted to measure the effect of an amino acid substitution. The unfolding rates of a wild type and its mutant variant can be measured, and the relative change of the free energy of unfolding ($\Delta\Delta G$) is an indicator of whether a particular mutation is stabilizing or destabilizing. Existing experimental data about mutation studies in proteins is available in the ProTherm database [26]. Unfortunately, mutagenesis experiments on physical proteins are expensive and time consuming, and consequently experimental data about the effects of mutations is limited.

Computational models that aim to predict the effect of a mutation have been developed, with varying levels of prediction accuracies. A class of such models are Machine Learning-based approaches. Although a number of them have shown great promise, most are dependent on a large dataset for the training phase of the model's development. Ensemble based systems – also called multiple classifier or mixture of experts systems – synthesize the predictions of multiple models. A variety of ensemble methods have been shown to make improved predictions over their single classifier models. Unfortunately, the ensemble based methods, too, are hindered by their dependence on "large enough" datasets for training purposes.

Thus although progress has been made in developing algorithmic approaches for inferring the effects of mutations, opportunities for improvements exist. In this work, we have developed a multi-stage compute pipeline that involves training single-expert systems – that we call Level 0 (**L0**) models, followed by a second stage in which several voting schemes are configured and trained on the Person Correlation and RMSE properties of the L0 models. We evaluate our voting schemes to show that they make improved predictions over the predictions of any of the L0 models.

2 RELATED WORK

In the following sections, we survey the existing experimental and computational approaches for predicting the effects of amino acid substitutions. We also review ensemble based systems. We leave the details of the Machine Learning Models that we use – SVR, RF, and DNN – to Section 4.5.

2.1 Wet-lab Mutagenesis

Wet-lab experiments provide the gold standard for directly measuring the effects of mutations on a protein's stability, measured by the $\Delta\Delta G$ of the mutant with respect to the wild type. Matthews et al. have studied many mutants of Lysozyme from Bacteriophage T4 [3, 6, 15, 29–31]. It was found that residues with high mobility or high solvent accessibility are much less susceptible to destabilizing substitutions. The downside of experimental studies is that they are time consuming and expensive. Moreover, some mutations are so destabilizing that the mutant protein cannot be crystallized at all. Thus, only a small fraction of all possible mutations can be experimentally studied.

2.2 Computational Approaches

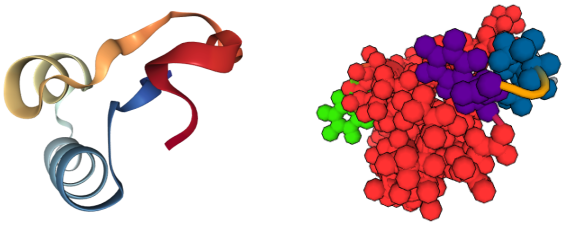
A variety of computational methods have been developed over the years to predict the effects of mutations on protein structure and stability. Several of these methods achieve high prediction and accuracy rates in the 70% range.

Several methods consider the backbone of a protein fixed, and perform a search for the best side-chain conformation. Many of them use rotamer libraries to search for the best side chain conformation upon an amino acid substitution [14, 25, 32]. Other studies used heuristic energy measures [27], database driven potentials [20] or Molecular Dynamics simulations [33]. Thus, progress has been made in predicting the effects of mutations on protein stability. However, many such methods rely on computationally intensive energy calculations and are therefore time consuming.

2.3 Combinatorial, Rigidity-Based Methods

Rigidity analysis was first used to explore the effects of mutations involved by calculating a rigid cluster's configuration entropy [34]. Rigidity Analysis [17, 21] is a combinatorial technique for identifying the rigid and flexible regions of biomolecules (Figure 1). Rigidity analysis, which identifies rigid clusters of atoms, is distinguished from most other methods by being very fast. It does not rely on homologous protein data, nor on costly all-atom energy calculations. See [17] for a detailed explanation of rigidity analysis.

In our initial previous work we used rigidity analysis to probe how an *in silico* mutation to glycine destabilizes a protein's structure. We compared the rigidity properties of the wild type structure to the rigidity properties of a mutant that we generated using KINARI-Mutagen [24]. On input of a PDB structure file, KINARI-Mutagen identified hydrogen bonds and hydrophobic interactions. The stabilizing interactions involving the atoms of the side chain being mutated to Glycine are removed from the protein's model. This is equivalent to computationally mutating a specific residue to Glycine, the smallest amino acid which has no side chain atoms that form stabilizing bonds. Later, we combined rigidity analysis with evolutionary conservation to find out if the two measures could give us richer, more complete information about important parts of the protein structure [2]. In subsequent work, we found out that even a simple, SVM-based machine learning scheme, gave better predictions [23]. In our most recent work, we developed Support Vector Regression, Random Forest, and Deep Neural Network models for predicting the effects of mutations [12]. In combination with an efficient computational approach for inferring the effects of mutations based on changes to the rigidity of a protein induced by a mutation [4], our most recent classifier systems was capable of predicting the effects of mutations with 71% accuracy.



(a) Cartoon (b) Rigidity Analysis

Figure 1: Cartoon and Rigidity analysis of PDB file 1crn. Atoms in the four largest rigid clusters are colored by cluster membership. The shown clusters are comprised of 387, 22, 22, and 12 atoms.

2.4 Ensemble Based Methods

Multiple classifier systems, which are also called ensemble systems, encompass a variety of approaches that synthesize and leverage information from multiple single-expert systems [35]. A common theme of ensemble approaches is the use of a diverse set of single-expert systems so that individual classifiers do not necessarily contribute the same type of information for a prediction or classification task. As such, ensemble systems work best when there exists ample data on which each individual expert system can be trained. Ideally, the data on which a single expert system is trained is different than the data in use by another single-expert system. In the absence of such volumes of data – for Deep Neural Networks for example, millions of data points are usually needed to achieve good results – a variety of approaches for diversifying the existing dataset are available, including Bagging, a machine learning algorithm that uses a forest of decision trees to increase diversity by bootstrapping replicas of a training dataset [9]. Bagging exhibits good performance in combining individual classifier results via a majority vote approach. Other approaches still, such as boosting, are algorithmic recipes for improving a weak learner, which minimally is better than random guessing, into a strong learner that can accurately classify a majority of instances [18]. A third class of ensemble methods, called mixture-of-experts, relies on a set of first level (L0) classifiers, which each generate a prediction. A second level (L1) classifier then considers the individual predictions of the first level classifiers via a variety of schemes, including random selection, winner takes all, or a weighted-majority approach were each prediction of a first level classifier is weighted based on past performance [22]. Combining the results of individual classifiers can involve a simple majority vote, or a more systematic weighted voting calculation. In a simple majority vote, the second level, or ensemble approach, might select that prediction that is achieved by the majority of single expert systems, which may or may not constitute more than 50% of the overall votes [8]. Probability-based approaches, which involve posterior probability calculations, are also available [13].

3 MOTIVATION

In this work, the available dataset for mutation experiments performed on physical proteins is composed of 2064 entries. This is not enough for the types of models needing millions of training data points. Unfortunately, this dataset is also not a good candidate for boosting or bagging (see Section 2.4), because a mere 3 proteins constitute more than 50% of all the mutation data available. Boosting or bagging in our case would bias or over fit a single-expert system as any reasonable data split of train, development, and testing would include a single protein whose mutation data would be contained in all three sets. This is not ideal, because such a system could not reasonably be expected to make predictions of the effects of mutations on any of the more than 100,000 proteins whose structures have been crystallography resolved [7].

To address these limitations, we have developed five voting schemes – second level classifiers – which generate predictions on a held-out testing set. **Our voting schemes are each a different numerical recipe that uses L0 predictions and L0 Root Mean Square Error (RMSE) and Pearson Correlation Coefficient (C) accuracy metrics, adjusted for noise caused by outlier predictions, to yield an L1 result**. Our L0 classifiers are the Random Forest, Support Vector Regression, and Deep Neural Network Machine Learning models.

4 METHODS

In this section, we detail our voting strategies, machine learning models used, and feature processing. The high-level overview of our compute pipeline is shown in Figure 2.

4.1 Voting Configuration

To train our voting schemes, we first trained our L0 models using data from the training set (see Section 4.4). The L0 models predict $\Delta\Delta G$ for each point in the train set (Box 1 in Figure 2). Each L0 model is then used to make predictions on the dev set (Box 2 in Figure 2). The accuracy (RMSE, C) of these predictions is calculated and retained. The L0 models' predictions on the dev set are used to find the best voting scheme hyperparameters.

To determine the best voting hyperparameters, we proceeded with an outlier configuration step. The purpose of this was to assess the performance of each L0 model based on its RMSE and C metrics. The outlier configuration involve a transformation of outliers (identified as L0 predictions whose error (L0_pred - truth) is greater than the RMSE of the whole data split), followed by a recalculation of a L0 model's RMSE and C values. These new RMSE and C metrics were then fed into the voting schemes. The last scheme, $VS_{combined-exp-wa}$ calculates how "good" different L0 models are by calculating a coefficient for each L0 based on its RMSE and C. This coefficient is modulated by an exponent. The voting configuration for each scheme was saved, and the best outlier transformation for each scheme was recorded. For the $VS_{combined-exp-wa}$ scheme, the best exponent value was also recorded. At this step, the voting scheme were configured. See Section 4.7 for a full description of the voting schemes.

4.2 Testing of Voting Schemes

For assessing the utility of the voting schemes (Box 3 in Figure 2), the L0 models made predictions on each entry in the test set, and their results were fed into the voting schemes. Each scheme used its saved voting configuration to weight the predictions of each L0 model. We calculated the RMSE and C of each L0 model on the test set, as well as the RMSE and C of each voting scheme on the test set, as a way to assess the improvement of the prediction using the voting schemes over just the L0 models.

4.3 Data preparation and Feature Extraction

For our previous work [16], we curated a dataset of mutant proteins for which we retrieved experimental mutation data from the ProTherm database. We used that dataset of 2063 unique entries about single amino acid substitutions performed in physical proteins. The effect of a mutation on the protein's structural stability can be correlated with its effect on a protein's rigidity. In our previous work we measured the effect of the mutation by recording the change in the size of the Largest Rigid Cluster (LRC) of the mutant versus the wild type [2, 23], and more recently have used not only the size of the largest rigid cluster, but the count of different rigid clusters and their sizes to infer the effect of a mutation [12]. We developed a variety of rigidity distance metrics for comparing the rigidity of a mutant to the rigidity of a wildtype to infer the effect of an amino acid substitution. From that past work, we have retained our best performing rigidity metric, the *lm*, or *l*inear *m*etric:

$$lm : \sum_{i=1}^{i=LRC} i \times linearWeight_i \times [WT_i - Mut_i]$$

where WT_i refers to the count of rigid clusters in the wildtype made up of i atoms, and Mut_i refers to the count of rigid clusters made up of i atoms in the mutant. The linear weighting adjusts each contribution of the sum based on the size of the rigid cluster. The motivation is that a mutation that reduces the size of the LRC is more destabilizing than a mutation that affects small rigid clusters only (See [4] for a full description).

Using both experimental data from ProTherm, and the results of the rigidity analysis, we retained the following features:

- Solvent accessible surface area (SASA): 2 real-valued features indicating how exposed to the surface a residue is (both absolute and percentage).
- Secondary Structure: 4 binary features indicating whether each mutation is part of a sheet, coil, turn or helix.
- Temperature and pH at which the experiment for calculating $\Delta\Delta G$ was performed.
- Rigidity distances *lm* metric (see above and [4]).
- Rigid Cluster Fraction: 48 features giving the fraction of atoms in the WT and Mutant that belong to rigid clusters of size 2, 3, ... , 20, 21–30, 31–50, 51–100, 101–1000, and 1001+, respectively.
- Residue type: 8 categorical features indicating whether the mutant wildtype and mutant target are Charged (D, E, K, R), Polar (N, Q, S, T), Aromatic (F, H, W, Y), or Hydrophobic (A, C, G, I, L, M, P, V).

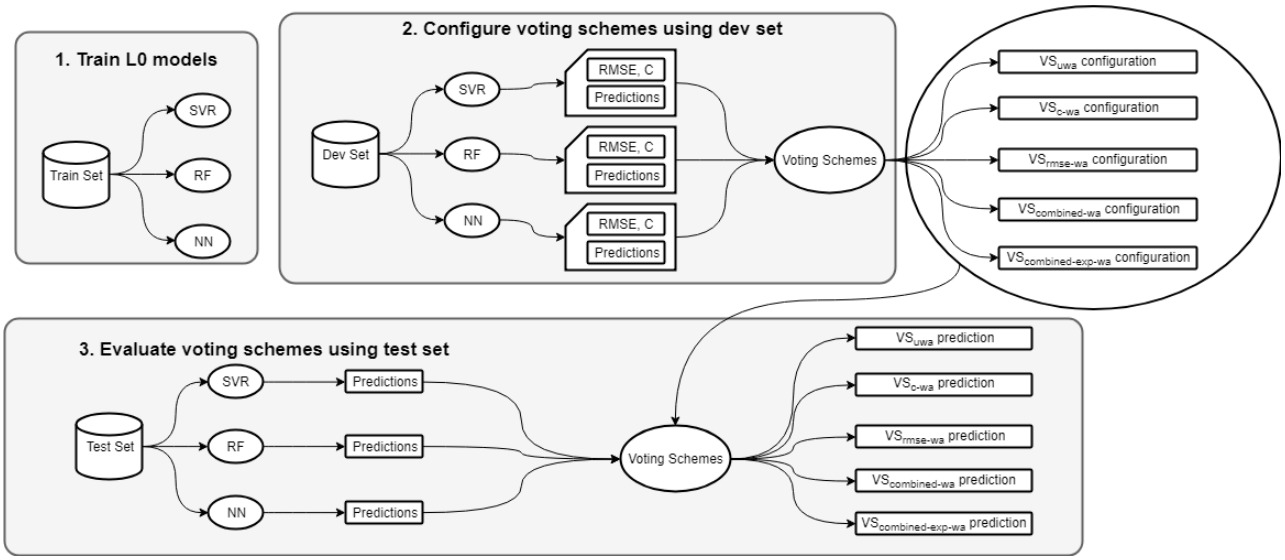


Figure 2: System Design. The three grey boxes represent our three stages which included training the L0 models, configuring and Voting Schemes (L1 models), followed by evaluation of the effectiveness of the voting using a test set.

4.4 Train, Dev, and Test Data Splits

Our data was split into training, development and test sets, for the purposes of L0 model training, voting scheme "configuration", and voting scheme evaluation, respectively. The union of train and development sets was used to tune the hyperparameters of L0 models. The statistics for these sets can be found in Table 1. Because approximately 50% of the dataset of 2063 entries from the ProTherm database was for mutations performed on 4 proteins only, we could not use traditional boosting methods to generate diverse test, dev, and training sets. We were not able to have each set be composed of mutations for distinct proteins, but were able to generate 2 different data splits in which all of the proteins in the dev and train sets were unique from the proteins in the test set.

Table 1: Data splits and set sizes

Dataset	Training	Development	Test	Total
Split 1	1754	167	142	2063
Split 2	1650	206	207	2063

4.5 L0 Models

We describe the three major machine learning methods employed in this work: support vector regression, random forests and deep neural networks. We used a variety languages, including the R programming language's *mlr* package; *mlr* is a wrapper for learning algorithms (*mlr* that "integrates" learning methods from other packages).

4.5.1 Random Forests. Random forests (RF) are ensemble models that combine the results of numerous decision trees in regression or classification tasks. A decision tree is a simple machine learning model that maps inputs to predictions by traversing a tree structure, in which a question about a feature is asked at each node. Trees are learned by applying splitting criteria at each node, such as information gain. A random forest ensembles many decision trees that are trained on multiple random sub-samples of the training set and then use averaging to improve the accuracy of the prediction and reduce model variance [10]. Each tree is trained on a set sampled from the full training set, generally with replacement, equal in size to the full set.

The random forest model we used is implemented by the *randomForest* library in the R programming language [28], specifically regr.RandomForest in *mlr*. We tuned our random forest model using 5-fold cross-validation on the union of the training and development sets over a range of hyperparameters. The specs of the hyperparameter search space included between 30 and 1000 trees ranging from 2 to 62 candidate features for each split, a minimum terminal node size of 1 to 5, and with `true` or `false` value for both the importance and proximity parameters. The best hyperparamters discovered included 148 trees with 18 candidates sampled per split, a minimum terminal node size of 2, and with `True` for importance and `False` for proximity.

4.5.2 Support Vector Regression. Support Vector Machines (SVMs) are supervised machine learning models that are effective for cases with large feature dimensions and small training sets. While most often used for classification, SVMs can be used for regression. This is often referred to as support vector regression (SVR) [5]. These models are trained to minimize regularized empirical risk in order to balance fitting the trends in the data and not over fitting noise. While SVR is by default a linear method, it can model non-linear relationships through the use of a kernel function, which implicitly, non-linearly maps the features into a (potentially infinite dimensional) feature space. Common non-linear kernels include the polynomial, sigmoid and radial basis kernels.

The SVR model that we used in this work for $\Delta\Delta G$ prediction was implemented using the *kernlab* package, specifically regr.ksvm in *mlr* [11].

For the SVR we tuned the following parameters by random search and 5-fold cross validation on the union of the training and development sets: SVM type, C (regularization term for Lagrange formulation), kernel and kernel coefficients ν, ϵ, σ, degree, scale, and offset. The hyperparameter search space included nu-svr and eps-svr for svr type, 2^{-4} to 2^4 for C , vanilladot, rbfdot, polydot, and tanhdot for kernel, 0.15 to 0.3 for ν, 2^{-5} to 2^5 for σ (for the rbf kernel), 0.0001 to 0.25 for ϵ, a degree of 2 for the polydot kernel, scale values of 0.01 to 1.5 for the tanhdot kernel, and offset values of 0.01 to 1.5 for the tahndot kernet. The best hyperparameters were achieved with svr type nu-svr, the rbfdot kenrel, a σ of 0.0583039419265523, 0.246922406381 for ϵ, and 0.205569167 for ν.

4.5.3 Deep Neural Network. Deep Neural Networks (DNNs) are supervised machine learning models that are composed from a sequence of parametric, differentiable "layers" and trained end-to-end. A shallow neural network has only two parametric layers (a "hidden" layer and an output layer). The output y is as follows:

$$y = W_{(L)}^T h_{(L)} + b_{(L)} \quad (1)$$

$$h_{(\ell)} = g\left(W_{(\ell)}^T h_{(\ell-1)} + b_{(\ell)}\right) \text{ for } i = 1, \ldots, L-1, \quad (2)$$

where the input x is written as $h_{(0)}$. There are L parameter matrices, W, and L bias vectors, b. The function g is known as an activation function and must be non-linear for the overall model to be non-linear. In this work we consider $g(z) = \tanh(z)$ and $g(z) = \text{ReLU}(z) = \max(0, z)$. The model parameters are learned using minibatch stochastic gradient descent to minimize the average training set squared error (MSE). We used the open-source machine learning library TensorFlow [1] for all DNN operations.

We performed 1000 iterations in which we trained on the train set and evaluated on the dev set our DNN model. The hyperparameter search space used a fixed patience of 500, 1000 epochs, tanh activation function, adam optimizer, using minibatch sizes of 8, 16, 32, 64, 128, 256, and 512, along with a learning rate of 0.001 to 0.1. There were 10 to 100 hidden units, between 1 and 9 layers, and an initial weight range of 0.001 to 0.1. To accommodate for noise in the model, we averaged the result of 9 runs. The best hyperparameter values were a learning rate of 0.0058542, 67 hidden units, the tahh activation function, the adam optimization algorithm, a minibatch size of 128, only a single hidden layer, and an initialization of 0.03403. For our random search method we utilized NVIDIA GTX 1080 Ti GPUs to find the best hyperparameters.

4.6 Voting Schemes - Outlier Transformations

Our voting schemes utilized the RMSE and Pearson Correlation (C) values for the L0 models. RMSE is the square root of the mean error over an entire set, while the Pearson Correlation is a classical correlation metric, which we calculated using python's scipy.stats.pearsonr method.

In order to get a good idea of L0 model accuracy in a general case, we de-emphasized outlier predictions when calculating RMSE and C scores for L0 models when "configuring" the voting schemes. We accomplished this by excluding outlier predictions or scaling them towards their ground truth value. Outlier predictions are identified in the following equation, where $h(x)$ is the prediction an L0 model on point x, $y(x)$ is the true $\Delta\Delta G$ value of point x, h_{RMSE} is the RMSE value of that L0 model, and T is the given outlier threshold.

$$\frac{h(x) - y(x)}{h_{RMSE}} > T$$

If more than 15% of predictions were identified as outliers, data from that particular outlier threshold was not used. Outlier thresholds tested were between 1.4 and 2.4, inclusive, incremented by intervals of 0.1. The constant scalar values tested were between 0.3 and 0.8, inclusive, incremented by intervals of 0.1. Our last scheme also modulate the exponent value, between -2 and 1.9 (inclusive), excluding 0.

4.7 Voting Schemes

With three different machine learning models trained to predict $\Delta\Delta$G values, we sought to combine the strengths of these algorithms to make an improved prediction. In order to leverage the additional information contained in the RMSE and C scores for the L0 models, we developed 5 voting methods. Much like a stacked ensemble, these methods use the predictions from multiple L0 models and their RMSE and C values to "vote" on a winning $\Delta\Delta$G value which may or may not be present in the input set. The 5 voting schemes are describe below.

VS_{uwa}: An unweighted average of all models' predictions for a given mutation. For m models each with an output $h_i(x) \in \mathbb{C}$, where $i = 1, 2, \ldots, m$, our voting prediction is:

$$\text{VS}_{\text{uwa}}(x) = \frac{1}{m}\sum_{i=1}^{m} h_i(x) \quad (3)$$

$\text{VS}_{\text{c-wa}}$: A weighted average of all models' predictions for a given mutation, adjusting each model's prediction based on the strength of its Pearson Correlation Coefficient, C, relative to the model with the best C. Again assume we have a set of models $h_i(x)$ for $i = 1, 2, \ldots, m$ but let $h_*(x)$ denote the output from the best performing model, c_i denote the C for model i and let c_* denote C for the best performing model, then our voting prediction is:

$$\text{VS}_{\text{c-wa}}(x) = \frac{1}{m}\sum_{i=1}^{m}\left(h_i(x) + (h_*(x) - h_i(x))\left(1 - \frac{c_i}{c_*}\right)\right) \quad (4)$$

$\text{VS}_{\text{rmse-wa}}$: A weighted average of all models' predictions for a given mutation analogous to $\textbf{VS}_{\textbf{c-wa}}$, except using RMSE instead of R. In this case, $h_*(x)$ is the prediction of the best model (according to RMSE), r_i is the RMSE for model i and r_* is the RMSE of the best model. Then our voting prediction is:

$$\text{VS}_{\text{rmse-wa}}(x) = \frac{1}{m}\sum_{i=1}^{m}\left(h_i(x) + (h_*(x) - h_i(x))\left(1 - \frac{r_*}{r_i}\right)\right) \quad (5)$$

$VS_{\text{combined-wa}}$: Weighted average of all models' predictions for a mutation incorporating both the C and RMSE performance. Letting $\gamma_i = c_i/r_i$ and γ_* denote the best (max) γ_i, our prediction is:

$$VS_{\text{combined-wa}}(x) = \frac{1}{m}\sum_{i=1}^{m}\left(h_i(x) + (h_*(x) - h_i(x))\left(1 - \frac{\gamma_i}{\gamma_*}\right)\right) \quad (6)$$

$VS_{\text{combined-exp-wa}}$: A weighted average of all models' predictions for a given mutation incorporating both the C and RMSE performance, but weighting one of those metrics more using ξ. Letting $\gamma_i = c_i/r_i$ and γ_* denote the best (max) γ_i, our prediction is:

$$VS_{\text{combined-exp-wa}}(x) = \frac{1}{m}\sum_{i=1}^{m}\left(h_i(x) + (h_*(x) - h_i(x))\left(\frac{\gamma_i}{\gamma_*}\right)^{\xi}\right) \quad (7)$$

Several hyperparameters were introduced to improve these voting scheme baselines in the event that outlier predictions significantly affect a model's C or RMSE score. In addition to the standard computation of a model's R and RMSE, we recalculated each model's RMSE and R after excluding outliers or scaling outliers by a constant, and only then commenced the voting schemes. Outliers were identified by comparing a threshold hyperparameter to the ratio of a prediction's loss to the model's RMSE. The best hyperparameter configuration for each scheme was used to generate the results of each voting scheme.

5 RESULTS

The prediction accuracy of the L0 and voting schemes on the testing set were evaluated by two metrics: Root Mean Square Error (RMSE) and the Pearson correlation (C) between the predicted and actual $\Delta\Delta G$ values.

Irrespective of the improvement of the voting schemes over the L0 models, our implementation of the L0 models yielded Pearson Correlation Coefficient (C) values of 0.756 and 0.687 for RF and DNN, respectively, on the test set. This is an improvement over our past work, in which we attained a Pearson Coefficient Correlation value of 0.71 [12]. Our current implementation also has an improved (lower) RMSE score for the RF model over our past work (1.124 versus 1.34). Complete results are shown in Table 2.

Table 2: Pearson correlation Coefficient, C, and RMSE scores for the L0 models (RF, DNN, SVR) and 5 voting schemes

Model	RMSE	C
DNN	1.305	0.687
RF	1.124	0.756
SVR	1.592	0.396
VS_{uwa}	1.182	0.727
$VS_{\text{c-wa}}$	1.155	0.749
$VS_{\text{rmse-wa}}$	1.165	0.744
$VS_{\text{combined-wa}}$	1.118	0.766
$VS_{\text{combined-exp-wa}}$	1.120	0.765

To assess the improvement of the voting schemes over the L0 models, we tallied the Pearson Correlation Coefficient, C, for all five voting schemes and the three L0 models for both the dev and test sets (Figures 3 and 4). For both the dev and test sets, the voting schemes outperformed the best L0 model (RF).

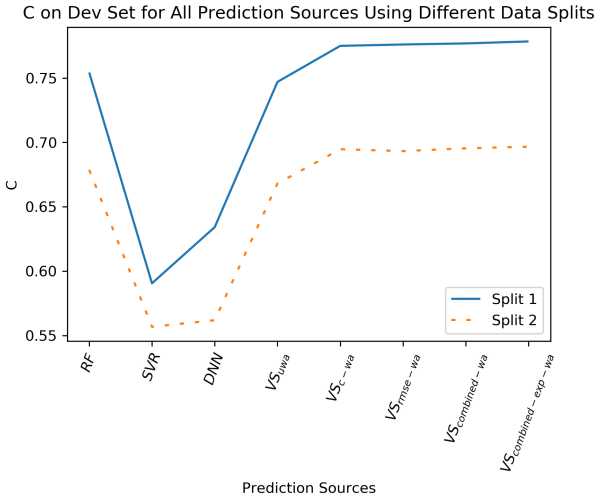


Figure 3: The Pearson Correlation Coefficient (C) value for each of the three L0 models and five voting schemes shows that for the dev set, the voting schemes had an improved performance over the L0 models using both splits 1 and 2.

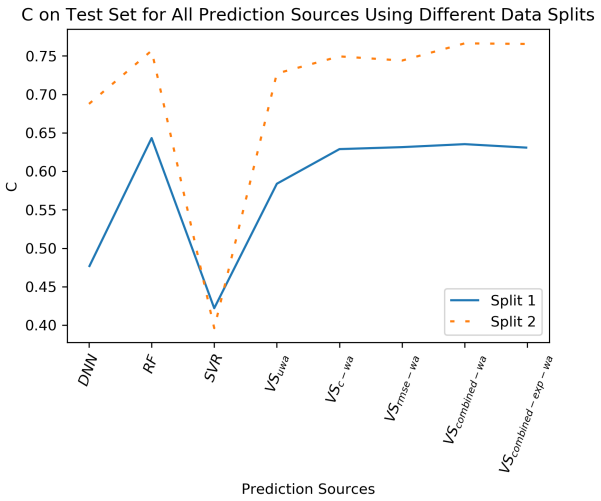


Figure 4: The Pearson Correlation Coefficient (C) value for each of the three L0 models and five voting schemes shows that for the test set, the voting schemes had an improved performance over the L0 models using both splits 1 and 2.

To assess the improvement of the voting schemes over the L0 models in terms of error, we tallied the RMSE metric for all five voting schemes and the three L0 models for both the dev and test sets (Figures 5 and 6). For both the dev and test sets, the voting schemes outperformed the best L0 model (RF).

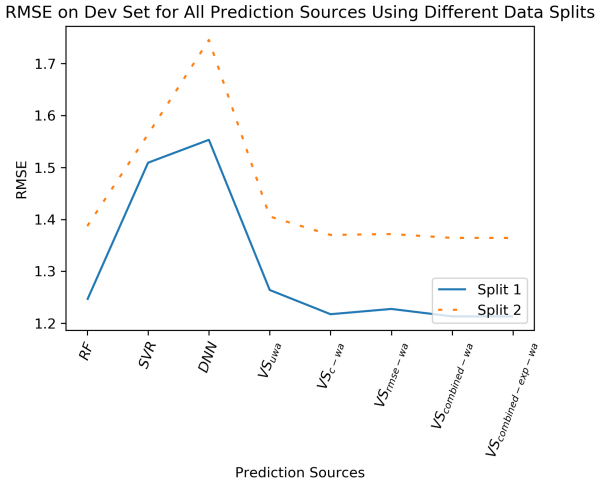


Figure 5: The Root Mean Square Error (RMSE) value for each of the three L0 models and five voting schemes shows that for the dev set, the voting schemes had an improved performance over the L0 models.

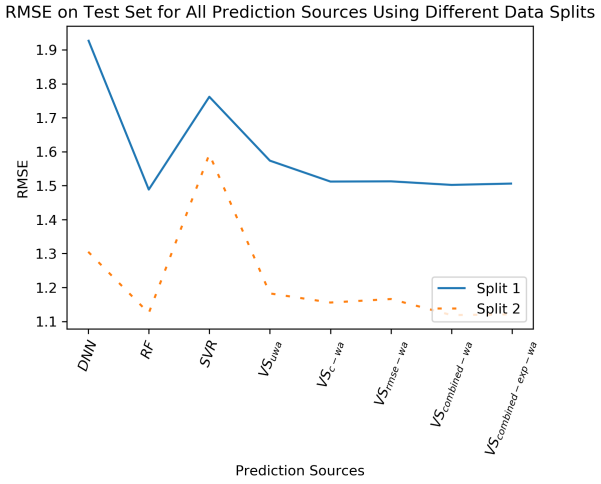
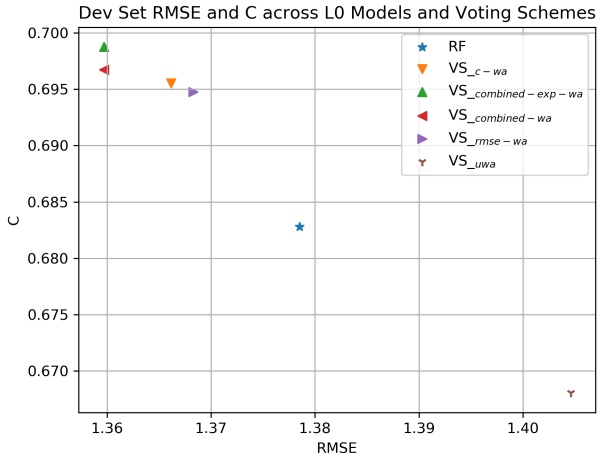


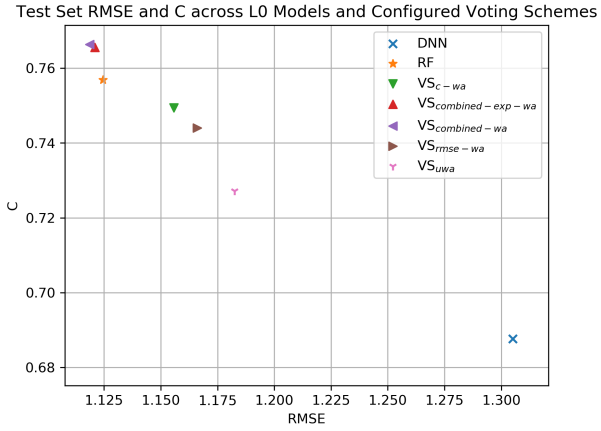
Figure 6: The Root Mean Square Error (RMSE) value for each of the three L0 models and five voting schemes shows that for the test set, the voting schemes had an improved performance over the L0 models.

To tally the improvement of the voting schemes over the L0 models in terms of both RMSE and C values (a high C and a low RMSE are best, while a low C and high RMSE is the worst), we plot both the RMSE and C scores for the 5 voting schemes as well as the best L0 models for the dev set (Figure 7) and test set (Figure 8). For both of those sets, the best voting scheme (routinely either $VS_{\text{combined-wa}}$ or $VS_{\text{combined-exp-wa}}$), outperformed the best L0 models.



Note: DNN and SVR data outside of graph bounds.

Figure 7: The Root Mean Square Error (RMSE) as well as the Pearson Correlation Coefficient C values for the best L0 model (RF) compared to the five voting schemes shows that all but theVS_{uwa} outperformed the L0 model on the dev set.



Note: SVR data outside of graph bounds.

Figure 8: The Root Mean Square Error (RMSE) as well as the Pearson Correlation Coefficient C values for the best L0 model (RF) and second best L0 model (DNN) compared to the 5 voting schemes shows that the two best performing voting schemes outperformed the L0 models.

To explore the sensitivity of the outlier configuration stages for the voting schemes, we systematically swept the outlier threshold (tv) and constant scalar value (cv) parameters during the development stages of the five voting schemes. In Figure 9 we see that when outliers were excluded, the VS_{uwa} performed the worst as judged by the Pearson Correlation Coefficient, while $VS_{\text{combined-wa}}$ performed the best. When the outlier values were scaled, VS_{uwa} had the worst performance of the five voting schemes, but also was

the least sensitive, while $VS_{\text{combined-wa}}$, although it did have the best Pearson Correlation Coefficient C values (yellow, upwards of 0.69), it also had the lowest C values for certain combinations of cv and tv. Our last voting scheme, $VS_{\text{combined-exp-wa}}$, used different ξ exponent values for the RMSE and C values of L0 models during the outlier configuration stage of model development, and a systematic analysis of its performance during a sweep of the ξ values reveals that certain cv, tv pairs (for example 0.3, and 2.3, respectively) yield high Pearson Correlation Coefficient C values (near 0.69), but that for other values (for example cv=0.3 and tv=2.2) its performance diminished, down to a Pearson Correlation Coefficient of near 0.57.

6 CONCLUSIONS AND FUTURE WORK

We have implemented three Machine Learning models (Level 0, L0) for inferring the effects of single amino acid substitutions on the stability of proteins. We have achieved prediction accuracies near 75%, which is an improvement over the 71% we realized in our most recent work. To improve on the predictions of the Machine Learning models, we have implemented five voting schemes (L1 models) which we trained using the RMSE and Pearson Correlation Coefficient values of the L0 models. Our voting schemes make improved predictions over the L0 models when tested on a held-out test set. For future work, we aim to explore further the sensitivity of the voting schemes, and incorporate hyperparameter values of the L0 models in the training component of the L1 predictors.

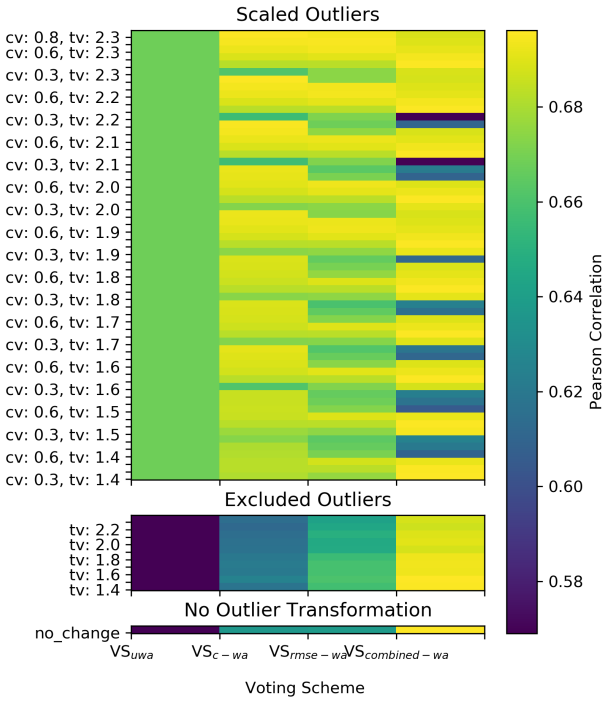


Figure 9: Heatmap of Pearson Correlation Coefficient values for voting schemes other than $VS_{\text{combined-exp-wa}}$ as a function of a systematic parameter sweep of the outlier threshold (tv) and constant scalar (cv) values during the outlier configuration stage of model development.

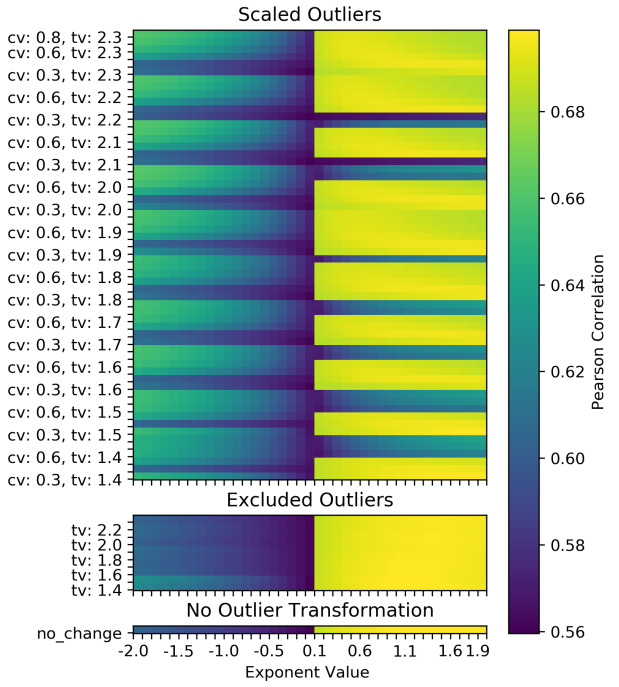


Figure 10: Heatmap of $VS_{\text{combined-exp-wa}}$ Pearson Correlation results on the dev set using different ξ exponent values given RMSE and C values for various outlier configurations. tv = outlier threshold value, and cv = constant scalar value.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Bahar Akbal-Delibas, Filip Jagodzinski, and Nurit Haspel. 2013. A conservation and rigidity based method for detecting critical protein residues. *BMC structural biology* 13, 1 (2013), S6.

[3] Tom Alber, Sun Dao-Pin, Keith Wilson, Joan A Wozniak, Sean P Cook, and Brian W Matthews. 1987. Contributions of hydrogen bonds of Thr 157 to the thermodynamic stability of phage T4 lysozyme. *Nature* 330, 6143 (1987), 41.

[4] Erik Andersson, Rebecca Hsieh, Howard Szeto, Roshanak Farhoodi, Nurit Haspel, and Filip Jagodzinski. 2016. Assessing how multiple mutations affect protein stability using rigid cluster size distributions. In *Computational Advances in Bio and Medical Sciences (ICCABS), 2016 IEEE 6th International Conference on*. IEEE, 1–6.

[5] Debasish Basak, Srimanta Pal, and Dipak Chandra Patranabis. 2007. Support vector regression. *Neural Information Processing-Letters and Reviews* 11, 10 (2007), 203–224.

[6] Jeffrey A Bell, Wayne J Becktel, Uwe Sauer, Walter A Baase, and Brian W Matthews. 1992. Dissection of helix capping in T4 lysozyme by structural and thermodynamic analysis of six amino acid substitutions at Thr 59. *Biochemistry* 31, 14 (1992), 3590–3596.

[7] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. 2006. The protein data bank, 1999–. In *International Tables for Crystallography Volume F: Crystallography of biological macromolecules*. Springer, 675–684.

[8] Philip J Boland. 1989. Majority systems and the Condorcet jury theorem. *The Statistician* (1989), 181–189.

[9] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[10] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[11] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3 (2011), 27.

[12] Ramin Dehghanpoor, Evan Ricks, Katie Hursh, Sarah Gunderson, Roshanak Farhoodi, Nurit Haspel, Brian Hutchinson, and Filip Jagodzinski. 2018. Predicting the Effect of Single and Multiple Mutations on Protein Structural Stability. *Molecules* 23, 2 (2018), 251.

[13] Richard O Duda, Peter E Hart, and David G Stork. 2012. *Pattern classification*. John Wiley & Sons.

[14] Roland L Dunbrack Jr and Martin Karplus. 1994. Conformational analysis of the backbone-dependent rotamer preferences of protein sidechains. *Nature Structural and Molecular Biology* 1, 5 (1994), 334.

[15] A Elisabeth Eriksson, Walter A Baase, Xue-Jun Zhang, Dirk W Heinz, MPBE Blaber, Enoch P Baldwin, and Brian W Matthews. 1992. Response of a protein structure to cavity-creating mutations and its relation to the hydrophobic effect. *Science* 255, 5041 (1992), 178–183.

[16] Roshanak Farhoodi, Max Shelbourne, Rebecca Hsieh, Nurit Haspel, Brian Hutchinson, and Filip Jagodzinski. 2017. Predicting the effect of point mutations on protein structural stability. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 247–252.

[17] Naomi Fox, Filip Jagodzinski, Yang Li, and Ileana Streinu. 2011. KINARI-Web: a server for protein rigidity analysis. *Nucleic acids research* 39, suppl_2 (2011), W177–W183.

[18] Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*. Springer, 23–37.

[19] Scott C Garman and David N Garboczi. 2002. Structural basis of Fabry disease. *Molecular genetics and metabolism* 77, 1 (2002), 3–11.

[20] Dimitri Gilis and Marianne Rooman. 1997. Predicting protein stability changes upon mutation using database-derived potentials: solvent accessibility determines the importance of local versus non-local interactions along the sequence1. *Journal of molecular biology* 272, 2 (1997), 276–290.

[21] Donald J Jacobs, Andrew J Rader, Leslie A Kuhn, and Michael F Thorpe. 2001. Protein flexibility predictions using graph theory. *Proteins: Structure, Function, and Bioinformatics* 44, 2 (2001), 150–165.

[22] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.

[23] Filip Jagodzinski, Bahar Akbal-Delibas, and Nurit Haspel. 2013. An evolutionary conservation & rigidity analysis machine learning approach for detecting critical protein residues. In *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*. ACM, 779.

[24] Filip Jagodzinski, Jeanne Hardy, and Ileana Streinu. 2012. Using rigidity analysis to probe mutation-induced structural changes in proteins. *Journal of bioinformatics and computational biology* 10, 03 (2012), 1242010.

[25] Joel Janin, Shoshanna Wodak, Michael Levitt, and Bernard Maigret. 1978. Conformation of amino acid side-chains in proteins. *Journal of molecular biology* 125, 3 (1978), 357–386.

[26] MD Shaji Kumar, K Abdulla Bava, M Michael Gromiha, Ponraj Prabakaran, Koji Kitajima, Hatsuho Uedaira, and Akinori Sarai. 2006. ProTherm and ProNIT: thermodynamic databases for proteins and protein–nucleic acid interactions. *Nucleic acids research* 34, suppl_1 (2006), D204–D206.

[27] Christopher Lee and Michael Levitt. 1991. Accurate prediction of the stability and activity effects of site-directed mutagenesis on a protein core. *Nature* 352, 6334 (1991), 448.

[28] Andy Liaw, Matthew Wiener, and others. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.

[29] Masazumi Matsumura, Wayne J Becktel, and Brian W Matthews. 1988. Hydrophobic stabilization in T4 lysozyme determined directly by multiple substitutions of Ile 3. *Nature* 334, 6181 (1988), 406.

[30] Blaine HM Mooers, Walter A Baase, Jonathan W Wray, and Brian W Matthews. 2009. Contributions of all 20 amino acids at site 96 to the stability and structure of T4 lysozyme. *Protein Science* 18, 5 (2009), 871–880.

[31] H Nicholson, E Söderlind, DE Tronrud, and BW Matthews. 1989. Contributions of left-handed helical residues to the structure and stability of bacteriophage T4 lysozyme. *Journal of molecular biology* 210, 1 (1989), 181–193.

[32] Jay W Ponder and Frederic M Richards. 1987. Tertiary templates for proteins: use of packing criteria in the enumeration of allowed sequences for different structural classes. *Journal of molecular biology* 193, 4 (1987), 775–791.

[33] Martine Prevost, Shoshana J Wodak, Bruce Tidor, and Martin Karplus. 1991. Contribution of the hydrophobic effect to protein stability: analysis based on simulations of the Ile-96—-Ala mutation in barnase. *Proceedings of the National Academy of Sciences* 88, 23 (1991), 10880–10884.

[34] S Radestock and H Gohlke. 2008. Exploiting the Link between Protein Rigidity and Thermostability for Data-Driven Protein Engineering. *Engineering in Life Sciences* 8, 5 (2008), 507–522.

[35] Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review* 33, 1-2 (2010), 1–39.