Predicting the Effect of Point Mutations On Protein Structural Stability

ABSTRACT
Predicting how a point mutation alters a protein’s stability can guide drug design initiatives which aim to counter the effects of serious diseases. Mutagenesis studies give insights about the effects of amino acid substitutions, but such wet-lab work is prohibitive due to the time and costs needed to assess the consequences of even a single mutation. Computational methods for predicting the effects of a mutation are available, with promising accuracy rates. In this work we study the utility of several machine learning methods and their ability to predict the effects of mutations. We in silico generate mutant protein structures, and compute several rigidity metrics for each of them. Our approach does not require costly calculations of energy functions that rely on atomic-level statistical mechanics and molecular energetics. Our metrics are features for support vector regression, random forest, and deep neural network methods. We validate the effects of our in silico mutations against experimental ∆∆G stability data. We attain Pearson Correlations upwards of 0.69.

CCS CONCEPTS
• Computing methodologies → Machine learning algorithms;
• Applied computing → Bioinformatics; Molecular structural biology;

KEYWORDS
Machine Learning; Protein Structure; Mutation; Support Vector Regression; Random Forest; Deep Neural Network; Rigidity Analysis

1 INTRODUCTION
The amino acid sequence of a protein determines its structure and as a result, its function. Even 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, a disorder that causes cardiac and kidney complications [14].

Wet-lab experiments can be used to engineer a protein with a specific mutation, and the mutant directly assessed to infer the effect of that amino acid substitution. The wild type and mutant proteins can be denatured to determine their relative unfolding rates, from which the free energy of unfolding (∆G) can be calculated; it is an indicator of whether a particular mutation is stabilizing or destabilizing, and to what degree. Existing experimental data about various mutations performed in physical proteins is available in the ProTherm database [23].

Unfortunately, conducting mutagenesis experiments on physical proteins is expensive and time consuming, and thus experimental data about the effects of mutations is limited. Therefore, computational methods can be helpful in estimating the effects of a mutation on a protein structure, and several computational methods have been developed in the past, with various degrees of success.

2 RELATED WORK
In this section we survey the existing experimental and computational work for predicting the effects of amino acid substitutions.

2.1 Experiments On Physical Proteins
Wet-lab experiments provide the gold standard for directly measuring the effect of mutations on a protein structure, measured by ∆G with respect to the wild type. Matthews
et al. have generated many mutants of Lysozyme from the Bacteriophage T4 [2, 7, 12, 26–28]. They found that residues with high mobility or high solvent accessibility are much less susceptible to destabilizing substitutions. Although such studies provide precise, experimentally verified insights into the role of a residue based on its mutation, they are time consuming and cost prohibitive. Additionally, some mutations are so destabilizing that the mutant protein cannot be crystallized at all. Thus, only a small subset of all possible mutations can be studied explicitly.

2.2 Computational Approaches

To complement and inform mutation studies performed on physical proteins, computational methods have been developed over the years. These methods strive to predict the effects of mutations. Several have high prediction and accuracy rates in the 70-80% range.

Several computational methods for assessing the effects of mutations fix the atoms in the backbone of a protein and proceed to search for the best side-chain conformation, while others utilize rotamer side chain libraries to adjust a structure in response to an amino acid substitution [11, 19, 31]. Other approaches [24] rely on heuristic energy measurements to predict the stability of proteins in which side chains are perturbed. Yet another approach [15] estimates the folding free energy changes upon mutations using database-derived potentials. Prevost [32] used Molecular Dynamics simulations to study the effect of mutating Barnase, and concluded that the major contributions to the free energy difference arose from non-bonded interactions. 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 intensive.

2.3 Combinatorial, Rigidity Based Methods

A first generation of rigidity-based mutation analysis tools are available, but the extent of the types of in silico mutations that they can perform are limited. Rigidity Analysis [16] is a combinatorial technique for identifying the flexible regions of biomolecules. Figure 1 depicts the cartoon and rigidity analysis results of PDB file 1hvr of HIV-1 protease. Rigidity analysis, which identifies rigid clusters of atoms, is distinguished from most other methods because it is fast. It does not rely on homologous protein data, nor costly all-atom energy calculations.

In our previous work we used rigidity analysis to probe how a 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 in silico using KINARI-Mutagen [18]. On input of a Protein Data Bank (PDB) structure file, KINARI-Mutagen identifies 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.

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 [5, 17] 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 (WT, non-mutated protein). The rationale was that the LRC is an indicator of the protein’s rigidity or flexibility. Predictions were validated against experimentally derived ΔΔG unfolding measurements from the ProTherm [23] database. A negative ΔΔG value for a mutant structure reveals that the mutant is less stable than the wild type, and thus the amino acid substitution is destabilizing.

2.4 Machine Learning Based Approaches

Machine learning (ML) is a branch of artificial intelligence involving algorithms that allow programs to classify, group, and learn from data. The regression problem proceeds via the following steps: a) represent a set of known data points as a set of feature vectors labeled by the corresponding output value, b) train a model that best maps inputs to the correct output, c) use the model to make predictions on a set of new (e.g. held out) data points. In Section 5 we detail each of the Support Vector Regression, Random Forest, and Deep Neural Network methods we used.

Machine learning and statistical methods have been developed to help predict the effects of mutations and to infer which residues are critical. Cheng et al. [10] used Support Vector Machines to predict with 84% accuracy the sign of the stability change for a protein due to a single-site mutation. Also, data of amino acid replacements that are tolerated within families of homologous proteins have been used to devise stability scores for predicting the effect of residue substitutions [35], which has been extended and implemented into an online web server [36].

In our previous work [17], we used an SVM-based model that combines rigidity analysis and evolutionary conservation, in addition to amino acid type and solvent accessible surface

![Figure 1: Rigidity analysis of PDB file 1hvr identifies sets of atoms belonging to rigid clusters. The largest rigid cluster is shown orange, which spans both halves of the protein. The larger rigid clusters are (cluster size : count) 11:6, 12:5, 15:2, 16:2, 19:2, 23:1 and 1371:1.](image-url)
area, to a dataset of proteins with experimentally known critical residues. We achieved over 77% accuracy in predicting the sign of the change of stability for a single point mutation to Glycine and Alanine.

Brender, et al [9], have developed a scoring function that reasons about protein-protein interfaces. They used sequence- and residue-level energy potentials in conjunction with a Random Forest (RF) approach to achieve a Pearson correlation coefficient of approximately 80% between predicted and observed binding free-energy changes upon mutations. Jia, et al [20], have employed a variety of Machine Learning Tools to generate several models based on thermotability data for assessing the effects of single point mutations. They used 798 mutants from 51 different protein structures for which there is ∆ΔG data, and attained accuracy rates ranging from 78-85% among SVM, RF, NBC, KNN, ANN, and PLS approaches, with the Random Forest Approach having the highest accuracy. Li, et al [25], developed a model based on the Random Forest algorithm for predicting thermostability changes due to amino acid substitutions. In their approach they relied on 41 features, and achieved accuracies of 79.9%, 78.2%, and 78.7% for single, double, and multiple point mutation.

3 MOTIVATION
As discussed above, existing experimental methods still provide only partial information about the effects of mutations. Computational methods can complement this information, but many existing methods are time consuming, or alternatively, their accuracy could be improved. There is a need for fast and reliable methods that can efficiently analyze the effect of a mutation of an amino acid on the structure of a protein. As already discussed, machine learning based methods have been used in the past by us and other researchers, and they are a promising avenue to explore further.

In this work, we present fast and efficient machine learning and graph theory based methods for predicting the effect of a mutation on a protein structure. Through rigidity analysis, support vector regression, random forests and deep neural networks, we predict the effect of a mutation on the ∆ΔG of a protein. We validated our results by using experimental data from the ProTherm database. Our approach achieves strong performance in predicting the effect of a single point mutation on a protein structure, while being fast enough to run in a few minutes and sometimes seconds.

Several aspects of our work distinguish it from others. Firstly, none of our features in use by our machine learning models require calculating energetics of various biophysical phenomena. The calculation of our metrics does not require costly calculations based on statistical mechanics nor molecular energetics. Our features are strictly structure-based, which is a purposeful design decision to enable near real-time run-times. Secondly, the number of data points that we use is far more than most others have used. With 2,072 mutations for which we have experimentally derived data from ProTherm, our dataset far surpasses in size most others, many of which have fewer than 1,000. This dataset is far greater than we used in our previous work due to our recent expanded capabilities of generating mutations in silico [4]. Lastly, the majority of our features in use by our models are derived from quick calculations detailing the rigidity properties of mutant, wild type pairs of protein structures. With the exception of our past proof-of-concept work, nobody else has used rigidity metrics on a large scale to assess their use in enhancing models for predicting the effects of mutations.

4 DATA PREPARATION
Here we describe the source of our data, including how we processed the ProTherm ∆ΔG values, how we generated in silico mutants, and the split of the data into training, development and testing sets for building our machine learning models. We enumerate our features and explain how each is normalized.

4.1 ProTherm Data, in silico mutants
We downloaded the entire ProTherm plain-text database, and identified 2,072 entries for single mutations with ∆ΔG values. See Table 1 for a summary of the proteins and their mutations that we retained.

We used our in-house software, Protein Mutation High Throughput (ProMuteHT) [4], for quickly generating protein mutants in silico, with the mutations for which we had ProTherm stability data. That software is composed of several parts, including custom scripts and algorithms, integrated with off-the-shelf open access tools and libraries. It generates large-to-small (LTS) as well as small-to-large (STL) amino acid substitutions. For an LTS mutation ProMuteHT removes atoms from a PDB file to simulate a substitution. For example, mutating Leucine to Alanine involves removing from a protein structure file the CG, CD1 and CD2 atoms from the Leucine being mutated. When a small residue is mutated to a larger one, the STL module relies on the freely available SCWRL 4.0 [22] software that makes predictions about a side chain’s orientation. To account for the steric clashes that might arise due to replacing a small residue with a larger one, we used the NAMD [30] software to perform 500 energy minimization steps, requiring approximately 5 seconds.

The rigidity of each mutant and its wild type were analyzed using the publicly available rigidity software by Fox et. al [13]. The rigidity output data is of the form rigid cluster size : count, which offers the distribution of clusters and their sizes that were identified. We used the rigidity data for each mutant, wild type pair to calculate 6 different rigidity metrics. See [3] for a complete discussion explaining the motivation and utility of these metrics at inferring the effects of mutations.

4.2 Features and Data Split
From the ProTherm data and our rigidity calculations, we derived the following 60 features:

- WT SASA: how exposed to the surface a residue is.
Table 1: Summary of the data obtained from ProTherm.

<table>
<thead>
<tr>
<th>Parsed Numerical Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Proteins</td>
</tr>
<tr>
<td>Total Mutations</td>
</tr>
<tr>
<td>Mutated Residues that are Hydrophobic</td>
</tr>
<tr>
<td>Mutated Residues that are Aromatic</td>
</tr>
<tr>
<td>Mutated Residues that are Polar</td>
</tr>
<tr>
<td>Mutated Residues that are Charged</td>
</tr>
<tr>
<td>Mutated Residues with 0-30% SASA</td>
</tr>
<tr>
<td>Mutated Residues with 30-50% SASA</td>
</tr>
<tr>
<td>Mutated Residues with 50+ SASA% SASA</td>
</tr>
</tbody>
</table>

- WT Secondary Structure: four features indicating whether the mutation took place in a sheet, coil, turn or helix.
- Temperature and pH at which the experiment for calculating \( \Delta \Delta G \) was performed.
- Rigidity Distance (RD): one of \( \text{lm}, \text{sig1}, \text{sig2}, \text{sig3}, \text{sig4}, \text{sig5} \). See [3] for a full explanation.
- WT Rigid Cluster Fraction: 24 features giving the fraction of atoms in the WT that belong to rigid clusters of size 2, 3, \ldots, 20, 21-30, 31-50, 51-100, 101-1000 and 1001+, respectively.
- Mutation Rigid Cluster Fraction: 24 features giving the fraction of atoms in the mutation belonging to the same set of bins as above.
- Residue type: four binary features indicating whether the residue is 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).

Table 2: Feature summary.

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Norm.</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WT SASA</td>
<td>None</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>2-5</td>
<td>WT Secondary Structure</td>
<td>None</td>
<td>{0, 1},</td>
</tr>
<tr>
<td>6</td>
<td>Temperature</td>
<td>0-1</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>7</td>
<td>Potential of Hydrogen (pH)</td>
<td>pH</td>
<td>( \approx [-1, 1] )</td>
</tr>
<tr>
<td>8</td>
<td>Rigidity Distance (RD)</td>
<td>Standard</td>
<td>R</td>
</tr>
<tr>
<td>9-32</td>
<td>WT Rigid Cluster Frac.</td>
<td>None</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>33-56</td>
<td>Residue Rigid Cluster Frac.</td>
<td>None</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>57-60</td>
<td>Residue Type</td>
<td>None</td>
<td>[0, 1]</td>
</tr>
</tbody>
</table>

These features along with their normalization scheme and range are summarized in Table 2.

Normalization “0-1” refers to the mapping

\[
\frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (1)
\]

while “pH” normalization refers to a slight variant that maps pH of 0 to -1, 7 to 0 and 14 to 1:

\[
\frac{x_i - 7}{7} \quad (2)
\]

Standardization refers to normalizing the feature to be zero-mean and unit-variance:

\[
\frac{x_i - \mu}{\sigma} \quad (3)
\]

We estimated the feature mean \( (\mu) \) and standard deviation \( (\sigma) \) on the training set, and used these same values to transform the development and test sets. Our \( \Delta \Delta G \) labels were also standardized prior to training using the training set statistics. The \( \Delta \Delta G \) values fall in the range of \([-5.02, 4.55]\) after standardization.

Our 2,072 data points were randomly split into a training set (1,438 data points), a development set (324 data points) and a test set (310 data points), under the constraint that all instances of each WT appear in a single dataset. This constraint was used so that we can assess how well our methods generalize to new protein wild types, and means that our reported results are more pessimistic than they would be under an unconstrained random split.

5 METHODS: SVR, RF, DNN

In this section, we provide more details about three popular machine learning methods, Random Forest, Support Vector Regression and Deep Neural Networks, that we used for predicting \( \Delta \Delta G \). That is, we solve a regression problem: our objective is building a function that minimizes the difference between calculated (predicted) and actual observed target values for all data points in the training set.

5.1 SVR

Support Vector Machines are supervised learning models that are widely used for solving classification and regression problems. In case of Support Vector Regression (SVR), the aim is to minimize the generalization error bound in order to achieve strong generalization [6]. The generalization error bound is the combination of the training error and a regularization term that controls the complexity of hypothesis space [34]. SVR works based on generating a regression function in a high dimensional feature space where the input data are implicitly mapped using a kernel function. The kernel function can be linear or nonlinear (polynomial, sigmoid or radial basis).

The SVR model that we used in this work for \( \Delta \Delta G \) prediction were implemented using Scikit-learn [29], which is a free machine learning library for the Python programming language. Scikit-learn offers a variety of easy to use clustering, classification and regression algorithms implementations.

We used grid search to tune the parameters of the model using our training and development sets. For the SVR model we tuned three main parameters; the penalty parameter of the error term (regularization constant) \( C \), the kernel function and the kernel coefficient \( \gamma \). Using the development set, we evaluated the prediction accuracy using values chosen from range of 0.1 to 1000 for \( C \), examined RBF, linear and sigmoid as the kernel functions and changed \( \gamma \) values from 0.0001 to
1. The lowest error was achieved with $C$ equal to 150, when the RBF kernel was used and $\gamma$ was set to 0.1.

5.2 Random Forest
Random forests (RFs) work by utilizing the power of decision trees. A decision tree infers decision rules from the training data to carry out a regression or classification task. The decision rules are generated based on the value of a feature that yields the best split of the training data using a metric such as information gain or Gini impurity index. A random forest is an ensemble learning method that fits a number of decision trees on multiple random sub-samples of the training set and then use averaging to boost the accuracy of the prediction and control overfitting [8]. Each tree uses a sample size the same as the original training set. The method allows these samples to be the same as the original training set (Replicate) or to be drawn with replacement (Bagging).

The random forest model was also implemented using Scikit-learn. To tune the model parameters, we examined the prediction error while changing the number of estimators (trees in the forest) from 10 to 1000. The accuracy of the model did not show improvements when beyond 150 estimators were used. For resampling methods, we used bagging (Bootstrap Aggregation), where each tree in the forest was trained using random subsamples from the training set chosen with replacement. Bagging showed over 30 percent improvement in accuracy compared to Replicate. For the rest of parameters such as maximum depth of the tree and minimum number of samples required to split an internal node we used Scikit-learn default values, but plan to include these parameters in our future tuning experiments.

5.3 DNN
A Deep Neural Network (DNN) is a supervised machine learning model in which the input undergoes multiple "hidden" layers of non-linear transformation before a prediction is made. This generalizes the standard, shallow neural network which has only a single hidden layer. Predictions, $y$, in our DNN are made according to the following:

$$y = W_T^{(L)}h_{(L)} + b_{(L)}$$

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

where $h_{(0)}$ denotes the input $x$, and our model parameters are matrices $W_{(0)}, \ldots, W_{(L)}$ and vectors $b_{(0)}, \ldots, b_{(L)}$. The hidden activation function is denoted $g$; we explored two options: $g(z) = \tanh(z)$ and $g(z) = \text{ReLU}(z) = \max(0,z)$. Our model parameters are learned using first order optimization algorithms to minimize training set mean squared error (MSE).

We implemented our DNN using TensorFlow [1], an open-source machine learning toolkit. Our hyperparameters were tuned using a combination of random search and the Spearmint [33] tool. Spearmint is a Bayesian hyperparameter optimization tool, incrementally exploring the hyperparameter space with the objective of maximizing expected improvement.

We tuned several hyperparameters on the developed set: number of hidden layers (1 - 4), number of units per layer (10 - 100), learning rate (0.0001 - 0.5), weight initialization range (0.0001 - 0.1), activation function (tanh, ReLU), and the optimizer used for backpropagation (stochastic minibatch gradient descent, adam [21]).

6 RESULTS
The SVR, RF and DNN models were trained initially using the training set and the hyperparameters were tuned to optimize development set performance as described in Section 5. After the optimal parameters were identified, in a final round of training for SVR and RF, the samples in training set and development set were combined and used as the training set. For DNN, whose results were highly dependent on the initial random weight initialization, we omitted this step of combination and retraining, and simply used the best model trained on the training set alone. We repeated our experiments using the features mentioned in Section 4 for six configurations where in each configuration we included one RD measure for training. The prediction accuracy of the models on the test set was then evaluated using two metrics. We calculated Root Mean Square Error (RMSE) and the Pearson correlation coefficient between predicted and expected (actual) $\Delta\Delta G$ values. The prediction accuracy of the three models for each RD measure is presented in Table 3. As shown in the table, the RF model consistently outperformed the SVR and DNN models by having lower RMSE and higher correlation coefficients. The DNN performance typically falls in-between the other two. This relative ordering is fairly consistent regardless of the RD value used as a feature to train the models. Averaged over the six experiments (last rows of Table 3), the RF gives the best performance and the SVR gives the worst. While the RF RMSE and correlation variation over different RDs is rather small, RF generated the least RMSE (0.820) and highest correlation (0.694) when $\text{sig}2$ was used in the feature set. With SVR and DNN $\text{sig}1$ and lm showed the best performance among the RDs, respectively.

7 DISCUSSION
To assess the utility of our three models in predicting the values of $\Delta\Delta G$ due to point mutations, we compared the Pearson Correlation Coefficients of our Random Forest model (our highest scoring average) against equivalent coefficients for 12 other approaches that we found in the literature [20]. Our Pearson Correlation Coefficient value of 0.689 would rank our RF approach 9th of 12, with Prethemut, ProMaya, and ELASPIC having attained higher correlation coefficient values of 0.72, 0.74, and 0.77, respectively. Understandably, any such comparison must be taken with caution, for example due to different data set sizes, different cross validation approaches, as well as data preprocessing. And although for this work we focused on a regression model rather than attempting a binary classification of the data, it is not uncommon in the literature for binary classification models to exclude neutral ($0\pm0.5 \Delta\Delta G$ kCal/mol) mutants. Any such
We developed and present several machine learning based methods to predict the effects of mutations on the stability of a protein. In particular, our method predicts the change to the free energy of unfolding upon mutation ($\Delta \Delta G$), using a combination of graph based rigidity analysis and features like solvent accessible surface area (SASA), temperature, pH, and the type of mutated amino acid. We trained and tested our methods on an extensive dataset taken from the ProTherm database, which contains experimental information about amino acid substitutions and their effect on a protein’s stability.

## 9 CONTRIBUTIONS

Roshanak conducted the RF and SVR analysis, Max performed the DNN analysis, and Rebecca preprocessed and aggregated the ProTherm data. Brian, Nurit, and Filip supervised the work. All authors contributed to the analysis of the results and the writing of the manuscript.

## ACKNOWLEDGEMENTS

The work is supported in part by NSF grant CCF-1421871 (NH). BH and MS gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

## REFERENCES


