Alma Mater Studiorum — University of Bologna

DEPARTMENT OF PHARMACY AND BIOTECHNOLOGY

International Master Degree in Bioinformatics

Prediction of the Effect of Single Amino Acid Protein Variants Using Deep Mutational Scanning Data

Candidate:
Saul Pierotti

Internal Advisor:
Prof. Pietro Di Lena

External Advisor:
Prof. Arne Elofsson [1]

[1] Department of Biochemistry and Biophysics, Stockholm University

Thesis submitted 2nd July 2021

Abstract

Knowledge of the effect of mutations is crucial for improving our understanding of protein function. Moreover, the rapid increase in the availability of genomic data poses the challenge of linking uncharacterised genetic variants to phenotypes. Deep mutational scanning is an experimental mutagenesis technique that leverages the use of next-generation sequencing for the assessment of the effect of mutations. This approach is high-throughput compared to site-directed mutagenesis, but still, it is not sufficient for the characterization of the mutational landscape of the known proteomes. As a consequence, computational tools for the prediction of the effect of mutations are a valuable resource. In this work, I present a series of supervised regression models for the prediction of the quantitative effect of mutations trained on deep mutational scanning data. My models, despite not requiring structural information, perform similarly to Envision, a predictor that uses structural features. I compare gradient boosted trees and linear regression models, and I also explore several validation and testing strategies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Predicting the effect of mutations is of fundamental importance for a wealth of biological problems. The goal of precision medicine of providing targeted medical treatment (Goetz et al., 2018) requires an in-depth understanding of the effect of genomic differences, and current experimental assays are not yet capable of providing this information with high enough throughput (Reeb et al., 2020). This is why computational tools that can fill the void of uncharacterized mutations are of extreme interest. Traditionally these tools have been trained on manually annotated protein variants, incorporating the bias which is inherent in such a subjective dataset (Gray et al., 2018). The emergence of next-generation sequencing coupled with functional selection assays made it possible to assess the effect of mutations in a more unbiased fashion. This technique, which also offers a much higher throughput compared to previous approaches, is called deep mutational scanning (Fowler et al., 2014). Even though deep mutational scanning is not yet capable of solving the problem of characterizing most of the human genotypic variation, not even for the coding genome, it can provide valuable training data for computational tools. This work aims at taking advantage of deep mutational scanning data for training a family of predictors of the effect of single amino acid variants. Core advancements in variant effect prediction will be explored in the next sections, together with a description of the biological significance of mutations, experimental methods for obtaining them, and a high-level overview of the machine learning approaches used.

It is worth pointing out that, even though variant effect prediction has potential uses in human medicine, the focus of this work is not on the clinical applications. No implicit claim that the predicted scores can be interpreted as a proxy for the pathogenicity of a variation is made. Being the models trained on deep mutational scanning data, the predictions should be only interpreted as an approximation of deep mutational scanning fitness. This may or may not be correlated to pathogenicity, depending on the specific experimental selection strategy used.

1.1 Proteins

A protein is a linear chain of amino acid residues linked together by amidic bonds, which folds in a three-dimensional shape as a virtue of the interactions among residue side chains, backbone atoms, solvent molecules, and other components of biological organisms (membranes, other proteins, ions, small molecules). Proteins can be classified as soluble if they are free-floating in the cytosol or the extracellular environment, or membrane-bound if they are inserted in a biological membrane (the cell membrane or the membrane of an organelle or a vesicle). An intermediate category is peripheral membrane proteins, which are linked to a biological membrane through lipid anchors, or weak interactions with other integral membrane proteins.

Amino acids are small molecules presenting a central carbon atom, called C_α, bound to a carboxylic group, an amino group, a hydrogen atom, and a side chain. Sidechains can be of 20 different types (plus some additional non-standard configurations), and the identity of the side chain determines the identity of the amino acid. In proteins, the carboxylic group of an amino acid forms an amidic (also called peptidic) bond with the amino group of another amino acid, forming a chain of connected amino acid which constitutes the protein. When integrated into a protein chain, an amino acid does not present any more the carboxylic and amino groups, which are now embedded in amidic bonds, but only the residual side chain is distinguishable. For this reason, amino acids embedded in a protein chain are called residues. The linear chain of repeating amino groups, carboxylic groups, and C_α atoms forms the backbone of the protein. The torsion angles among backbone atoms define the three-dimensional configuration of the protein. In general, the bond between the nitrogen of the amino group and the C_α, and the bond between the C_α and the carboxylic carbon are free to rotate (defining torsion angles called ϕ and ψ, respectively), while the peptidic bond itself is rigid and planar because of resonance with the carboxylic oxygen. Alternative peptidic bond torsion angle configurations (called ω), however, are observed with proline.

The 20 standard residues can be categorized, according to the chemical property of their side chains, in apolar aliphatic, apolar aromatic, polar, positively charged and negatively charged. Apolar aliphatic residues are alanine (A), leucine (L), isoleucine (I), and methionine (M). Methionine is different from other aliphatic residues in that it presents a sulfur atom in a thioether bond along the side chain. Apolar aromatic residues are tyrosine (Y), phenylalanine (F), and tryptophan (W). Polar residues are serine (S), threonine (T), asparagine (N), glutamine (Q), and cysteine (C). Positively charged residues are lysine (K), histidine (H), and arginine (R). Arginine can, however, also effectively interact with aromatic residues thanks to the highly delocalised electronic configuration of its guanidino group. Negatively charged residues are aspartic acid (D), and glutamic acid (E). Proline (P), glycine (G), and cysteine (C) are a bit harder to categorize. Proline is the only amino acid to present a covalent bond between the side chain and a backbone atom different from C_α (it forms a bond with the aminic nitrogen). Glycine does not have a side chain at all, being replaced by a single hydrogen atom.

Cysteine presents a terminal thiol group on its side chain, and it can form covalent bonds with other cysteine residues (disulphide bridges) in oxidising conditions (particularly extracellularly).

The three-dimensional shape of a protein is responsible for its function and it is conceptualized in a hierarchy of interactions. The sole determinant of the three-dimensional shape of a protein in its native conformation is the identity and order of the residues composing it. The primary structure of a protein is the linear chain of residues, held together by covalent bonds. The secondary structure refers to stereotypical modes of interaction of the protein backbone, which gives rise to a limited number of well-distinguishable structural patterns: helices, strands, turns, and coils (Kabsch et al., 1983). The secondary structure of a residue is fully described by its backbone torsion angles (Ramachandran et al., 1963). The identity of the residues composing the primary structure of a protein determines which backbone torsion angles are favoured, and thus which secondary structures will be formed. The interactions among secondary structure elements, usually mediated by side-chain atoms, are responsible for the tertiary structure of a protein. The tertiary structure of a protein is its complete three-dimensional shape and it is responsible for protein function since it determines the interactions that the protein will be able to have with other molecules. Some proteins can interact in a stable manner and form complexes. The mode of interaction among protein subunits represents the protein quaternary structure.

The interactions responsible for protein structure, besides the covalent bonds responsible for the primary sequence, are almost all weak interactions. Hydrogen bonds have a primary role, especially among backbone atoms in the establishment of secondary structures (but they can also be present among side-chain atoms among polar and charged residues). Electrostatic interactions among charged side-chains are also important. Apolar residues experience hydrophobic forces when in a polar solvent. London dispersion forces among hydrophobic residues are also present. Disulphide bridges are among the few known covalent interactions responsible for higher protein organization, and they are exclusive of cysteine residues. Other more sporadic covalent interactions have also been observed (Ito et al., 1991, for example). Very recently, Wensien et al. (2021) reported the observation of a new type of covalent cross-link in proteins between lysine and cysteine, consisting of a NOS bridge.

In general, soluble globular proteins consist of a hydrophobic core, which is essential for the stability of the fold, and a surface that interfaces with the solvent. Interestingly, the presence of this hydrophobic core seems to be a universal feature of protein domains (Kalinowska et al., 2017).

1.2 Mutations and Mutagenesis

A mutation is any alteration in the sequence of a genome or an isolated nucleotide fragment.

Broadly, it is possible to distinguish mutations that affect single nucleotides (also called point mutations) and mutations that involve the insertion, deletion, or re-arrangement of larger portions of the genome. Insertions, deletions and re-arrangements have a profound effect on the protein product of the affected gene, but they are not the focus of this work.

In the context of protein-coding genes, a point mutation can be classified according to its effect at the residue level. Synonymous mutations alter the nucleotide sequence of the gene but leave the amino acid sequence of the protein unchanged. This is possible since the genetic code is degenerate and different codons can code for the same residue. This is not to say, however that synonymous mutations do not have any physiological effect: different codons for the same residue can be translated at different rates, affecting the amount of protein produced, and the nucleotide difference can affect splicing and the binding of proteins to the gene's DNA and mRNA (Hunt et al., 2014).

Non-synonymous mutations, on the contrary, affect the residue composition of the protein product. They do so by altering the wild-type codon to a codon that codes for a different residue type. The effect of non-synonymous mutations includes all the nucleotide-level alterations discussed for synonymous mutations, but it adds also the effect that the residue substitution has on the protein product. Substituting a residue in a protein can affect protein folding, stability, and function, aspects explored more in detail in Section 1.2.1. Because of their direct and causal effect on protein function, non-synonymous mutations on protein-coding genes, also called Single Aminoacid Variants (SAVs), are the main focus of this work.

The third category of point mutations in protein-coding genes is that of nonsense mutations. These mutations involve the conversion of a residue-coding codon (also called sense codon) to a stop codon (nonsense codon). The effect of nonsense mutations is often dramatic since they cause the premature termination of protein synthesis, causing the elimination of potentially vital portions of the protein.

Even though mutations happen spontaneously and they are one of the driving forces of evolution, they can also be caused willingly by researchers. This process is termed mutagenesis and involves experimentally generating genetic variants of a given sequence. When applied to protein-coding genes, mutagenesis can be invaluable in shedding light on the catalytic mechanism of enzymes (Peracchi, 2001), on the mechanics of protein folding and the residue interactions underpinning protein stability (Nisthal et al., 2019). Mutagenesis has also clear potentials for livestock and crops improvement and in genetic engineering (Davies, 1988; Holme et al., 2019; Kalds et al., 2019).

1.2.1 Effect of Residue Substitutions in Proteins

The effect of substituting a residue for another in a protein can vary dramatically according to the identity of the introduced residue, the local biophysical environment, and the mechanistic function of the protein.

Missense mutations in proteins typically result in large perturbations of stability and can lead to aggregation propensity (DePristo et al., 2005). The effect of a mutation on the stability of a protein can be measured in terms of the difference in folding free energy change (ΔG) between the wild-type and mutated protein (difference called $\Delta\Delta G$). Multiple mutations have an approximatively additive effect on stability, even though epistatic effects with a small number of other positions are common (Daopin et al., 1991; Green et al., 1993). Interestingly, most $\Delta\Delta G$ values are on the same order of magnitude as ΔG values (DePristo et al., 2005). Destabilizing mutations also result in increased aggregation, owing to the larger pool of unfolded proteins that they cause (Chiti, 2000; Ramirez-Alvarado et al., 2000). Mutations that influence aggregation independently from stability are also known (Mitraki et al., 1991; Pawar et al., 2005). Mutations that affect directly function are rarer since they need to occur on catalytic residues (DePristo et al., 2005).

1.2.2 Low- and Medium-Throughput Mutagenesis

The first attempts at genetically modifying a biological sequence were done in a non-targeted fashion, by exposing cells to a mutagenic agent such as ionizing radiations and then selecting among the many random mutants generated those deemed of interest on a phenotypic standpoint (Muller, 1927; Stadler, 1928).

Site-directed mutagenesis (reviewed in Shortle et al., 1981) was developed in the 1970s' by Clyde Hutchison, Marshall Edgell, and Michael Smith and paved the way for a wealth of experimental techniques aimed at precisely manipulating biological sequences. The development of these methods later earned the 1993 Nobel prize in Chemistry to Michael Smith, a prize shared with the inventor of the Polymerase Chain Reaction (PCR) Kary B. Mullis.

The site-directed introduction of point mutations can be achieved by annealing a mutant DNA oligomer to the wild-type sequence to be mutated and using it as a primer to perform elongation using DNA polymerase (Hutchison et al., 1978). The resulting mutant sequence can be then amplified by PCR and inserted in the organism of interest, obtaining the respective protein product and examining the phenotypic effect of the mutation. A more advanced technique to sample the mutational landscape of a protein applies site-directed mutagenesis repeatedly to a stretch of biological sequence, examining the effect of mutations at different positions. This technique, called single aminoacid scanning mutagenesis, usually performs substitutions towards a single residue and typically features alanine (reviewed in Morrison et al., 2001). Alanine is an ideal candidate for such studies since, compared to other residues and thanks to its simple methyl sidechain, it does show the effect of not having the wild-type functional groups in a given position while also it does not dramatically alter backbone flexibility.

1.2.3 Deep Mutational Scanning

Even though site-directed mutagenesis and alanine scanning greatly served the genetics and biophysics communities, recent developments in sequencing technologies made it possible to extend the technique even further.

Deep mutational scanning (Fowler et al., 2010; Fowler et al., 2014) is a high-throughput method for the characterization of large numbers of mutations in a given protein sequence. When all, or approximately all, the single amino acid variants of a sequence are produced and examined, the technique also takes the name of saturation mutagenesis.

The typical deep mutational scanning workflow involves the generation of a library of mutants, a functional assay for the selective enrichment of mutants that affect positively a property of interest, and a next-generation sequencing step to precisely quantify this selective effect.

The mutant libraries can be randomly produced or can contain a methodical selection of all the single residue mutations, all the double residue mutations, or higher-order combinations. The choice of the library is tightly coupled to the scientific question to be addressed. For example, double mutant libraries are essential in the study of pairwise residue interactions (as in Olson et al., 2014), while single mutant libraries are much easier to produce and assay, and are sufficient for the study of the mutational sensitivity of specific positions.

The first deep mutational scan used site-directed mutagenesis with doped oligomers (Knight, 2003) to randomly generate mutants for the WW domain of the protein Yap65 (Fowler et al., 2010). This library generation approach makes it almost impossible to obtain all the single mutants for a sequence, given its bias against multiple nucleotide substitutions. Site-saturation mutagenesis (Jain et al., 2014) is more apt at generating complete single mutant libraries. Since this technique requires an individual PCR reaction for each position to be mutagenized, however, it is quite time-consuming.

The functional assay is a fundamental part of any deep mutational scanning experiment, and the choice of which kind of assay to use is inevitably coupled to that of the expression vector and to the scientific question that motivated the experiment. More or less directly, protein stability is frequently one of the selected properties. For example, in binding assays, where antibodies or natural binding partners are used to selectively bind the protein of interest, the shape of the protein is of central importance for the selection, and this in turns depends on its thermodynamic stability.

Fluorescent or fluorescently-labelled proteins or antibodies can be used together with a Fluorescence-Activated Cell Sorter (FACS) for performing *in vitro* variant selection. Deep mutational scanning experiments that take advantage of a fluorecence-based *in vitro* selection step include Bhagavatula et al. (2017), Matreyek et al. (2017) and Sarkisyan et al. (2016).

Survival and competition assays performed with the mutant protein transfected in live cells can give a more complete picture of the effect of a given mutation in the organism, but at the same time the results will be confounded by the many ways that a protein can influence organismal fitness, and it can be harder to pinpoint a certain effect to precise biophysical changes in protein structure. These kinds of growth assays were, for example, used in Mishra et al. (2016) and Starita et al. (2015).

Next-generation sequencing is a central part of the deep mutational scanning pipeline. Short-read and high throughput technologies such as Illumina (Bennett, 2004) are commonly used. A shortcoming of Illumina sequencing is the short read length, which makes it impossible to sequence through a typical human gene in one pass. This is particularly problematic in deep mutational scanning since many almost identical sequences (mutants) are sequenced together and must be told apart from each other. The typical solution to this problem is the use of DNA barcoding (Hiatt et al., 2010). With DNA barcoding, short (<20 bp) random sequences are cloned next to the mutagenized regions and are sequenced and quantified before and after selection in place of the actual mutations. Subsequently, a sequencing platform capable of longer reads such as PacBio (Eid et al., 2009) is used for associating each barcode to a specific mutation.

The scoring of the fitness of a variant in a deep mutational scanning experiment can be done in several ways. When only two timepoints of variant frequencies are present (i.e. before and after a selection step), the logarithmic ratio of the two frequencies can be used as a fitness score. When more time points are present, a linear regression model can be applied between mutant frequency and time, and the slope of the regression line will represent the fitness score.

The statistical model Enrich2 (Rubin et al., 2017) offers automation and standardization in the determination of fitness scores from raw sequencing data from deep mutational scanning experiments.

1.3 Variant Effect Prediction

Despite the potentials from deep mutational scanning, the goal of having a complete mutational effect atlas even just for the human proteome is still not in reach. For model organisms, this is a target that is not even in sight. Nonetheless, assessing the effect of mutations is a central theme in personalised medicine (Goetz et al., 2018).

The need for phenotypically scoring variants and the limitations of the available experimental techniques prompted the development of computational tools aimed at predicting the effect of unseen variants. The first generation of variant effect predictors was created to characterize the vast amount of novel genetic variants discovered with the advent of genome sequencing projects (Collins et al., 1998; Ng et al., 2001; Ramensky, 2002). These predictors were developed as classifiers, that produced a binary classification of mutations as pathogenic/non-pathogenic.

Some of the variant effect predictors that have been developed are simple statistical models, while others take advantage of more complex machine learning algorithms. Sorting Intolerant from Tolerant (SIFT, Sim et al., 2012) and SIFT for genomes (SIFT 4G, Vaser et al., 2015) use the likelihood of observing a given residue in a specific position in a multiple sequence alignment of homologous sequences as a proxy for assessing pathogenicity. They then produce binary predictions by imposing a threshold on the likelihood score. PolyPhen-2 (Adzhubei et al., 2010) is another binary variant effect predictor that uses sequence-based and structure-based predictive features with a naive Bayes classifier. Evolutionary Action (Katsonis et al., 2014) is also a statistical method, similarly to SIFT. It was trained exclusively on human Single Nucleotide Polymorphisms (SNPs), and thus it is optimized to predict only the effect of human variants. SNAP2 (Hecht et al., 2015) is an ensemble neural network-based classifier that uses structural features, evolutionary features, and biophysical features of the mutant and wild-type residues. In addition, this method uses a sliding window around the mutated position to obtain information on the sequence context of the mutation. Combined Annotation Dependent Depletion (CADD, Kircher et al., 2014; Rentzsch et al., 2021; Rentzsch et al., 2018) is a predictor that in addition to predicting the effect of single amino acid variants, it also predicts the effect of insertions, deletions, nonsense mutations, and mutations that alter splicing. It was first implemented as a support vector machine in (Kircher et al., 2014), and then as a logistic regression in (Rentzsch et al., 2018). The peculiarity of CADD is that instead of relying on annotated pathogenic variants, it is trained in distinguishing mutations that became fixed in the human lineage after the split with the last human-ape ancestor (termed proxy-neutral mutations) from a simulated set of *de novo* variants (termed proxy-deleterious). Envision (Gray et al., 2018) is a more recent, quantitative predictor that was directly trained on deep mutational scanning data. It features a gradient-boosting tree algorithm and it uses a combination of structural, evolutionary, and sequence features. Unsupervised models such as EVmutation (T. Hopf et al., 2017) and DeepSequence (Riesselman et al., 2018) have also been developed and perform surprisingly well. EVmutation is a statistical model that evaluates the epistatic effect between protein positions by examining multiple sequence alignments, but without ever being exposed to experimental variant effect scores. DeepSequence has a similar conception to that of EVmutation, but it models higher-order dependencies between protein positions by using a deep variational autoencoder.

The performances of variant effect predictors, especially in regression, is currently quite poor. It was observed that a simple position-specific scoring matrix can perform better than complex, machine learning-based predictors (Reeb et al., 2020). Poor performances, especially in the case of neutral mutations at evolutionarily conserved positions, could be due to a bias in the data used in training (Liu et al., 2013).

1.4 Machine Learning

Machine learning is a central tool in bioinformatics when the prediction of some quantity is of interest. Here I will briefly summarise the rationale behind the machine learning approaches used in this work. Instead of going into the mathematical details of the implementations used, which can, in any case, be explored by the interested reader in the respective original publications, this section wants to give an intuitive overview of the methods and practices that were used in the development of this project. In particular, an explanation of the effect of the hyperparameters optimised in this work is provided. For a more detailed overview of machine learning and its uses in computational biology, the reader can refer to Chicco (2017).

1.4.1 Overfitting

Overfitting is the failure of a trained model in generalizing to data not seen during training and optimization.

Overfitting is a typical problem that affects overparameterized models, where the complexity of the architecture is not justified by complexity in the relationship between independent and dependent variables. In extreme cases, a model will just memorize the training instances, without actually learning the relationship of interest between the features and the target variable. The opposite problem to overfitting is underfitting, which happens when a model is too simple to capture the complexity in the data.

To reduce overfitting is possible, depending on the circumstances, to choose a simpler model, increase the amount of data, or apply regularization. A good discussion of overfitting is available in Chicco (2017).

1.4.2 Validation and Testing

Validating a model means assessing its generalization performances. This is usually done by reserving a part of the data and excluding it from the training process. The model is trained on the so-called training set (the part of the data reserved for training) and then it is used to predict the data instances in the left-out fraction (called validation set). The goodness of a model can then be evaluated from the error in predicting instances in the validation set.

Validation can be used to assess if a model is overfitting the training dataset. In the case of overfitting, the validation error will tend to be sensibly higher than the training error.

A commonly used strategy is cross-validation, where the available data is split in a given number of sets of data points (called folds), and in turn one of these folds is left out for validation and the remaining ones are used for training. The process is repeated training new models until all the folds have been used for validation. The overall performances of the model can then be evaluated by aggregating all the validation predictions, or by calculating performance metrics for each fold.

If, as it is commonly done, the performances in the validation set are used for optimizing hyperparameters (in the broad sense, including selecting which features to use and which model to use), there is a potential for overfitting also the validation set. For this reason, it is good practice to leave out of training and validation an additional portion of the data, which is never used in model development. This test set will be used only to give an estimation of the true performances of the model with unseen data. Also for validation and testing, Chicco (2017) has a nice description of the strategies recommended in computational biology.

1.4.3 Hyperparameter Optimization

A hyperparameter of a machine learning model is a parameter whose value is not learned directly but is used to control the learning process. On the contrary, the optimal value of a model parameter is learned during training.

To select the optimal values for the hyperparameters of a model, it is necessary to try out numerous combinations of them. This will allow mapping a performance landscape in the hyperparameter space, from which the operator can in turn select a local maximum. It is possible to systematically probe all the possible combinations of a set of values for the hyperparameters, a process called grid search, or to randomly select a small fraction of the possible combinations of a much larger set of values (a process called random search).

A grid search can be computationally expensive because it suffers from the curse of dimensionality when many hyperparameters have to be optimized, while a random search can be more easily used also for large numbers of hyperparameters. In both cases, however, it is usually trivial to parallelize the hyperparameter search since testing each combination of values is an independent task. Random search outperforms grid search when only a small number of hyperparameters affects disproportionately the final performance of the model (in other words, when the optimization problem has a low intrinsic dimensionality).

Other approaches to hyperparameter tuning are possible but were not used in this work. They include Bayesian optimization, gradient-based optimization, and evolutionary optimization.

Once a satisfactory set of hyperparameter values has been found, it is possible to train a final predictor setting the hyperparameters to those values. Excessive hyperparameter tuning can lead to overfitting of the validation data, and thus an independent test data is always an essential requirement for unbiased estimation of model performances.

For more details on hyperparameter tuning see Bergstra et al. (2012) and Claesen et al. (2015).

1.4.4 Linear Regression

Linear regression is a simple model for the prediction of one (univariate) or more (multivariate) dependent variables from a set of independent variables. The prediction for the dependent variable is expressed as a linear combination of independent variables, with linear coefficients estimated from the data by minimising the sum of the squares of the errors. In this contest, the errors are the residuals between the predictions of the model and the actual values for the dependent variable.

The implementation used in this work does not have any free hyperparameter, since no regularization was used.

A more in-depth discussion of linear regression is available in Yan et al. (2009).

1.4.5 Gradient Boosted Trees

Gradient boosted trees (Freund et al., 1997; J. Friedman et al., 2000; J. H. Friedman, 2001) are a machine learning approach that recently reached great popularity thanks to their excellent results in extracting information from structured data. Compared to neural networks, which are the method of choice for unstructured data and when the spatial or sequential structure of the data is crucial for the prediction (for example, in computer vision and natural language processing), gradient boosted trees perform better with data in a tabular format, where a series of observations is associated with a series of features.

Gradient boosted trees are, like random forest, an ensemble method based on decision trees. Decision trees are weak learners with high variance, amenable to be combined in an ensemble. Differently from random forest, where the prediction of individual decision trees is averaged or in any case summarised horizontally, in gradient boosting each tree is trained in predicting the residuals from previous learners.

Gradient boosted trees can employ a variety of loss functions for regression, classification, and ranking (reviewed in Natekin et al., 2013). The choice of the loss function is, in itself, a hyperparameter of the model. For regression, common loss functions are the Mean Squared Error (MSE) and the Mean Absolute Error (MAE). MSE penalizes more large errors compared to MAE. When using regression trees in classification tasks (as some implementations do, instead of using directly classification trees), the output of the ensemble is subjected to a logistic or softmax function, and then a regression loss such as MSE or MAE is used. Gradient boosted trees that natively perform classification can use the binary or categorical cross-entropy as a loss function. Ranking loss functions include the pairwise loss (number of incorrectly ranked data pairs) and list-wise losses. In Learning To Rank problems, the LambdaMART (Burges, 2010) gradient boosting formulation is a common choice.

The two main hyperparameters in gradient boosted trees are the learning rate and the number of iterations. A small learning rate shrinks the contribution of successive trees as compared to the contribution of earlier ones. A learning rate of 1 makes the contributions of all trees equal. The number of iterations in the contest of gradient boosted trees is the number of successive trees trained to fit the residuals of the ensemble. Learning rate and number of iterations are closely intertwined in that a small learning rate will require more iterations to reduce the error to a similar level, but the result will be more stable than when using a larger learning rate. It can be said that a small learning rate and a large number of iterations can reduce overfitting at the cost of longer training times.

Regularization in gradient boosted trees can be enforced by applying a penalty term to the absolute values of the weights (L1 or lasso penalty) or their square (L2 or ridge penalty). The weights in this scenario are the factors that multiply the output of individual weak learners in the error function.

Subsampling is another common regularization technique in gradient boosted trees which consists of considering only a random fraction of the training data for each learner. Subsampling can be applied also column-wise (on the features), such that each weak learner uses only a random fraction of the features available. Besides reducing overfitting, subsampling can also speed up training by reducing the number of data points considered at each iteration.

The maximum depth is a hyperparameter that governs the maximum depth to which individual trees are allowed to grow. Increasing the maximum depth leads to more complex individual learners, and increases the risk of overfitting. Enforcing a minimum child weight also acts as a regularizing hyperparameter. The child weight is the number of instances in a leaf node in the case of equally-weighted data instances, and the sum of the instances weights otherwise. When this hyperparameter is set, the algorithm will give up partitioning any further a node when its weight goes below the threshold. Similarly to providing a minimum child weight, it is also possible to specify a minimum loss reduction required to continue splitting a node.

Additional hyperparameters besides the one described here are possible, depending on the loss function used and on the implementation.

Chapter 2

Materials and Methods

2.1 Software

The features used to train the predictive models presented in this work were produced using a variety of tools. Multiple sequence alignments and Hidden Markov models were obtained with the HH suite (Steinegger et al., 2019, version 3.3.0), and with the HMMER package (Eddy, 2011, version 3.3.2). The EVcouplings package (T. A. Hopf et al., 2018, version 0.1.1), EVmutation (T. Hopf et al., 2017, installed as part of EVcouplings), and plmc (T. Hopf et al., 2017, installed from `https://github.com/debbiemarkslab/plmc` in March 2021) were used for obtaining first- and second-order coupling terms among protein positions. The neural network trRosetta (Yang et al., 2020, installed from `https://github.com/gjoni/trRosetta` in March 2021 and using the trained model `2019_07`) was used for the prediction of protein contact maps. The neural network NetSurfP-2.0 (Klausen et al., 2019) was used for the prediction of relative solvent accessibility, secondary structure, torsion angles, and disorder.

DSSP (Define Secondary Structure of Protein, Kabsch et al., 1983, version 3.1.2) was used for the assignment of protein secondary structure, torsion angles, and residue solvent accessibility from experimental protein structures. DSSP assignments were not used in predictions but only for validating the output of NetSurfP-2 and for assessing how the fitness scores are affected by the solvent accessibility and the secondary structure of the mutated position.

The Python programming language (van Rossum, 2010) was used throughout the project for implementing the models and for intermediate processing steps. Different Python versions were used for different tasks according to the requirements of libraries and applications. Virtual environments and containers were used for managing the different versions of Python and the libraries. The last version of Python at the time of writing (3.9.2) was used whenever possible. TensorFlow (Abadi et al., 2015, version 2.4.1) and the Keras (Chollet et al., 2015) implementation `tensorflow.keras` were used for the development of neural network models. At the time of writing TensorFlow did not support Python 3.9, so for running scripts requiring TensorFlow Python was downgraded to version 3.8.8. Scikit-learn (Pedregosa et al., 2011, version 0.24.1) was used for data preprocessing, for cross-validation and testing, and for deploying some of the models. XGBoost (Chen et al., 2016, version 1.4.0rc1) was used to implement regressors based on gradient boosted trees. NetworkX (Hagberg et al., 2008, version 2.5.1) was used for representing the residue contacts in proteins as graphs, and for calculating properties of those graphs. The libraries Pandas (The pandas development team, 2021, version 1.2.3), NumPy (Harris et al., 2020, version 1.20.1), and Scipy (Virtanen et al., 2020, version 1.6.2) were used throughout the project. Some of the code was parallelized for running on supercomputer clusters using MPI for Python (also called mpi4py, Dalcin et al., 2011, version 3.0.3). Joblib (The joblib development team, 2020, version 1.0.0) was used for implementing shared-memory parallelization and for persisting and compressing Python objects. The interactive Python development environment IPython (Perez et al., 2007, version 7.19.0), run through JupyterLab (Kluyver et al., 2016, version 3.0.12) was used for prototyping and visualizations. Plots in the prototyping phase were produced using Matplotlib (Hunter, 2007, version 3.4.1) and Seaborn (Waskom, 2021, version 0.11.1).

All the visualizations shown in this manuscript were produced with the R programming language (R Core Team, 2021, version 4.0.5) using chiefly the package ggplot2 (Wickham, 2016, version 3.3.3) and the tidyverse ecosystem (Wickham et al., 2019, version 1.3.1). The packages cowplot (Wilke, 2020, version 1.1.1) and scales (Wickham et al., 2020, version 1.1.1) were used for tweaking the appearance of the visualizations. Biostrings (Pagès et al., 2020, version 2.56.0) was used to obtain the values for the BLOSUM100 scoring matrix (Henikoff et al., 1992) shown in Figure 3.3b. The machine learning package caret (Kuhn, 2021, version 6.0–88) was used for obtaining the classification evaluation metrics shown in Table 3.2. Directional (Tsagris et al., 2021, version 4.9) and circular (Agostinelli et al., 2017, version 0.4–93) were used for obtaining respectively correlation statistics among circular quantities and between circular and linear quantities. The only circular quantities used in this work are the backbone torsion angles. Reshape2 (Wickham, 2007, version 1.4.4) was used for data reshaping. Boot (Canty et al., 2021, version 1.3–28) was used for obtaining bootstrapped confidence intervals for the model performances. The package rlang (Henry et al., 2021, version 0.4.11) was used for functionalising some of the R code. The RStudio development environment was also used heavily (RStudio Team, 2021, version 1.4.1103 “Wax Begonia”).

2.2 Databases

Protein sequence databases were used extensively in this project. UniProt (Bateman et al., 2020, release `2021_02`) was used as an authoritative source for the wild-type sequence of the mutagenised proteins. Uniclust30 (Mirdita et al., 2016, release `2020_06`) was used for the construction of multiple sequence alignments with the hhblits tool (HH suite). UniRef100 (Suzek et al., 2014, release `2021_4`) was used by jackhmmer (HMMER package) in the first step of the EVcouplings pipeline. Experimental protein structures stored in the Protein Data Bank (PDB, Burley et al., 2018, accessed in March 2021) were used for assessing the quality of predicted structural features. InterPro (Blum et al., 2020, release 84.0) was used for determining protein domain boundaries when trimming the query sequences. Structure Integration with Function, Taxonomy and Sequence (SIFTS, Dana et al., 2018, accessed in March 2021) and the Protein Data Bank in Europe-Knowledge Base (PDBe-KB, Varadi et al., 2019, accessed in March 2021) were used for mapping PDB structures to UniProt positions.

2.3 Computational Infrastructure

Light computational work and simple manipulations were performed locally on a Linux machine. Due to the storage requirements for large protein sequence databases, the multiple sequence alignments were performed on a remote Linux server at SciLifeLab (Stockholm, Sweden), kindly provided by Prof. Arne Elofsson. GPU-heavy work such as neural network training was also performed on the same server, which was equipped with two Nvidia GeForce RTX 2080 GPUs. The most computationally demanding steps (random search of hyperparameter space for the models) were performed on the supercomputer cluster Kebnekaise, of the High-Performance Computing Center North (HPC2N) located in Umeå, Sweden. Access to the infrastructure was provided under the project SNIC2020–5–300 of Prof. Elofsson.

2.4 Dataset

The deep mutational scanning datasets used in this study were obtained from the training set of the variant effect predictor Envision (Gray et al., 2018). For convenience, fitness measurements for each mutation were extracted from the supplementary files available in Gray et al., 2018 and not from the respective dataset publications. Envision used a normalized version of the raw fitness scores in its training, but for this work, I opted for using the raw scores of each mutation.

To maximise the comparability of the results of this study with those of Envision, the deep mutational scanning datasets that were excluded from the training of Envision were filtered out also in this work. The retained data consisted of nine independent experiments on eight different proteins. They employed vastly different methodologies for the evaluation of variant fitness and included proteins of bacterial, yeast, rat, and human origins. Each dataset consisted of a set of single amino acid variants associated with a fitness score. A score of 0 was assigned to the wild type, with positive values indicating variants more abundant than the wild type after selection, and negative values variants less abundant than the wild type. A summary of the datasets is reported in Table 2.1. The fitness scores were obtained from the sequencing counts for each mutation according to Equation (2.1), where m_a and m_b are the read counts for the mutant sequence after and before selection, and wt_a and wt_b are the read counts for the wild-type sequence, after and before selection.

$$Fitness = \log \frac{m_a/m_b}{wt_a/wt_b} \tag{2.1}$$

2.4.1 Correction for Dataset `kka2_1:2`

It was observed in preliminary analyses that the dataset `kka2_1:2` (Table 2.1) contained duplicated mutations with different fitness scores in the aggregated version of the dataset from Gray et al. (2018). Curiously, these duplicated entries involved only and all of the mutations towards isoleucine, tyrosine, serine, asparagine, and glutamate. It was not possible to determine the cause of the duplicated entries, which are absent from the original dataset from Melnikov et al. (2014). In a private communication, the corresponding author of Gray et al. (2018) informed me that the inclusion of these duplicated entries is most probably accidental. For this reason and to avoid possible biases in training towards these specific mutations, the entries relative to this dataset were sourced directly from the original publication, Melnikov et al. (2014).

Melnikov et al. (2014) reports different categories of fitness scores, measured under different selection conditions. The scores that were included in the training set for this work are the ones obtained with kanamycin selection at half of the minimum inhibitory concentration. The pre-computed fitness scores were used, and not the raw sequencing counts. All the statistics for `kka2_1:2` used in this study reflect the data sourced from Melnikov et al. (2014) and not the data included in Gray et al. (2018).

2.4.2 Other Minor Inconsistencies

It was noted that for dataset `hsp90` the mutations from position 212 to 222 are missing, while they are present in the original publication (Mishra et al., 2016). It was also noted that for dataset `E1_Ubiquitin` the mutations from position 40 to 48 are missing. In the original publication, it is mentioned that these exact positions were used for method development but are nonetheless included in the fitness measurements. It was not possible to determine the source of these two discrepancies, but given the small number of mutations involved no action was taken. This was also done to maintain the dataset as similar as possible to the one used for training Envision, to allow for more meaningful comparisons.

2.4.3 Filtering

The datasets were purged of the mutations to or from non-standard or unknown residues. The declared original residue for each mutation was confronted with the residue in that position on the UniProt sequence. Mismatching entries were excluded from further processing. See Table 2.2 for a summary of the number of retained data points for each dataset.

Table 2.1: Summary of the deep mutational scanning datasets used in this work. The column "Dataset name" reports the name used throughout this work to refer to each dataset and is identical to the names used in Gray et al., 2018.

Dataset name	Source organism	UniProt ID	Reference
`beta-lactamase`	*Escherichia coli*	`P62593`	Firnberg et al., 2014
`WW_domain`	*Homo sapiens*	`P46937`	Fowler et al., 2010
`PSD95pdz3`	*Rattus norvegicus*	`P31016`	McLaughlin et al., 2012
`kka2_1:2`	*Klebsiella pneumoniae*	`P00552`	Melnikov et al., 2014
`hsp90`	*Saccharomyces cerevisiae*	`P02829`	Mishra et al., 2016
`Ubiquitin`	*Saccharomyces cerevisiae*	`P0CG63`	Roscoe et al., 2013
`Pab1`	*Saccharomyces cerevisiae*	`P04147`	Melamed et al., 2013
`E1_Ubiquitin`	*Saccharomyces cerevisiae*	`P0CG63`	Roscoe et al., 2014
`gb1`	*Streptococcus* sp. group G	`P06654`	Olson et al., 2014

Table 2.2: Number of mutations in each dataset before and after filtering. Table that summarises the number of mutation originally present in each deep mutational scanning dataset, and the number of mutations that were retained after filtering.

Dataset name	Total number of mutations	Number of retained mutations
`beta-lactamase`	5436	5397
`WW_domain`	377	373
`PSD95pdz3`	1577	1577
`kka2_1:2`	5280	5280
`hsp90`	4417	4231
`Ubiquitin`	1403	1267
`Pab1`	1276	1220
`E1_Ubiquitin`	1198	1142
`gb1`	1045	1026

Table 2.3: Trimming strategy for the multiple sequence alignments (MSA) queries. Table that details the range of positions in each mutagenized protein for which mutations were present in the respective dataset and how the query sequences were trimmed before being used in database searches for building multiple sequence alignments. All the positions are indicated according to the UniProt numbering of the respective sequences.

Dataset name	Mutagenized range	MSA query trimming	Trimming strategy
`beta-lactamase`	1 to 286	full protein	not trimmed
`WW_domain`	170 to 203	167 to 207	InterPro `IPR036020`
`PSD95pdz3`	311 to 393	303 to 426	InterPro `IPR036034`
`kka2_1:2`	1 to 264	full protein	not trimmed
`hsp90`	2 to 231	2 to 231	InterPro `IPR036890`
`Ubiquitin`	2 to 76	1 to 76	InterPro `IPR000626`
`Pab1`	126 to 200	126 to 203	InterPro `IPR000504`
`E1_Ubiquitin`	2 to 70	1 to 76	InterPro `IPR000626`
`gb1`	229 to 282	226 to 283	InterPro `IPR000724`

2.4.4 Normalization of Fitness Scores

For the single protein models (Section 2.7.1), the fitness scores were used in their original, non-normalized format. For the general models (Section 2.7.2), the fitness scores were used in their original format for models that could be easily adapted in using a ranking loss function, while they were normalized for models that relied on a regression loss function as the minimum squared error.

The normalization strategy used aimed at making the scores more easily comparable across datasets, and at reducing the excessive influence that datasets with a large score range would otherwise have on the loss function. The scores were first grouped by dataset and quantile normalized to the range 0 to 1 using 100 quantiles. Subsequently, for each dataset from the normalized scores, the quantile value corresponding to a score of 0 was subtracted. This had the effect of centring the data in each dataset such that positive scores would correspond to an increase in stability compared to the wild-type, and vice-versa for negative scores.

2.5 Features

All the predictors were trained using the same set of features. These included only information that could be derived from the sequence of the proteins and comparison with homologous sequences, avoiding the need for structural data. The approach proposed in this work leverages the use of unsupervised models, supervised predictors for structural data, sequence information, and position-specific scoring matrices.

2.5.1 Mutation Identity

For each data point, the original and mutated amino acid were provided to the models in one of two different ways. One-hot encoding was used for gradient boosted trees, linear regressors and support-vector machines. Each residue was represented by a vector with 20 elements, one for each possible standard residue. Given their rarity, mutations to or from non-standard or unknown residues were excluded from the dataset and thus the encoding of only standard residues was sufficient for the scope of this work. For neural networks, an ordinal encoding was used as input to an embedding layer. The dimensionality of the residue embedding was treated as a hyperparameter.

2.5.2 Multiple Sequence Alignments

Multiple sequence alignments are central to this predictor since they constitute the input for the computation of many of the features used. In general, alignment methods based on Hidden Markov Models were adopted because of their superior speed in large database searches and their increased sensitivity in the identification of remote homologies as compared to more traditional methods such as PSI-BLAST (see the introduction of Steinegger et al., 2019, and references therein). The tool hhblits from the HH suite was used for most of the alignments. An exception was the alignments used for the inference of evolutionary couplings, for which the pipeline included in the EVcouplings package was used (see Section 2.5.4). The hhblits tool was used with default parameters (which correspond to two iterations) and enforcing a minimum query coverage of 70% against the Uniclust30 database.

The query sequences used correspond to the respective UniProt canonical sequences for each of the mutagenised proteins. Most of the deep mutational scanning datasets used in this work focused on only one domain of a protein. For this reason, both the full-length UniProt sequence and the sequence trimmed to the mutagenised domain were used, obtaining two distinct multiple sequence alignments for each protein (excluding experiments where the full protein was mutagenised, where a single multiple sequence alignment was obtained).

In most cases the multiple sequence alignments used in input for the extraction of features were the ones corresponding to the full-lenght query sequence, with three exceptions:

(i) The datasets `Ubiquitin` and `E1_Ubiquitin`. Ubiquitin is encoded in the yeast genome in four different genes, and some of those are polyproteins with multiple domain repeats. The first ubiquitin repeat of the yeast gene UBI4 was used as query sequence for this protein in all instances.

(ii) The dataset hsp90. It covers the ATPase domain of the yeast molecular chaperone HSP82. The full-length alignment of this protein tended to recruit many sequences unrelated to the ATPase domain but homologous to the rest of the HSP82 sequence. For this reason, the trimmed version of the query was used.

(iii) Preliminary visual comparison of the contact maps obtained with trRosetta with experimental contact maps retrieved from structures from the Protein Data Bank showed that more accurate predictions could be obtained by using multiple sequence alignments trimmed to single domains instead of full-length alignments. Thus, trRosetta was always given in input the multiple sequence alignments trimmed to the mutagenised domains.

The protein positions considered in each case are reported in Table 2.3. The trimming step was always done at the level of the query sequence before the generation of the multiple sequence alignments. In all cases, the alignments were further processed by trimming out all the columns which corresponded to gaps in the query sequence.

2.5.3 Position-Specific Scoring Matrices

The position-specific scoring matrices (PSSMs) were obtained from the emission probabilities of the Hidden Markov models trained from the hhblits multiple sequence alignments produced as described in Section 2.5.2. The Hidden Markov models were trained with the tool “`hmmbuild`” from the HMMER suite. This tool was preferred to the corresponding tool “`hhmake`” from the HH suite (which was used for obtaining the multiple sequence alignments in the first place) since the latter does not include pseudo counts in the model file and thus needed further processing for producing usable PSSMs.

From the model file, the PSSMs were extracted by taking the emission probabilities of the match states for positions of the query corresponding to match states, and by taking the emission probabilities of the insert states for positions of the query corresponding to insert states. A Python script produced in house was used for the purpose.

The entire PSSM vector for each position was used as a feature, together with the PSSM scores for the wild-type residue and the mutated residue, and the difference between the two.

2.5.4 Evolutionary Couplings

EVmutation is an unsupervised model for the prediction of mutation effect. It is available as part of the EVcouplings package, which implements a full pipeline from sequence to the prediction of mutation effect (among other functions).

Internally, first EVcouplings produces a multiple sequence alignment of the query using an available installation of the tool "`jackhmmer`" in the system (part of the HMMER suite). After obtaining the multiple sequence alignment, it calculates the coupling between positions using plmc, a method for the inference of undirected graphical models that describe first-order couplings. Finally, EVcouplings calls EVmutation and predicts second-order coupling terms between multiple sequence alignment columns. The output of this stage is a table containing every possible single mutation in the query and the associated plmc and EVmutation scores. Both scores were used as features in the predictor.

For all stages, the default parameters from the sample EVcouplings configuration file were used. For the jackhmmer search, the parameters used included five iterations, a relative sequence and domain bit score threshold of 0.5 bits per residue, and UniRef100 as a database.

2.5.5 Predicted Structural Features Obtained with NetSurfP-2.0

The neural network NetSurfP-2.0 is a state-of-the-art predictor for protein secondary structure, residue solvent accessibility, torsion angles, and disordered residues. The hhblits-based variant of the network was used, which uses hhblits-derived multiple sequence alignments. The multiple sequence alignments given in input were obtained with hhblits as described in Section 2.5.2. All the output values of the network were used as features. These include the relative and absolute solvent accessibilities, the three- and eight-class secondary structure class probabilities, torsion angles, and disorder probability.

2.5.6 Connectivity Graphs

Protein contacts were predicted with trRosetta. The contacts were then used for defining connectivity graphs of the proteins, from which various features were extracted.

The inputs given to trRosetta were the multiple sequence alignments obtained with hhblits from the sequence of the mutagenised proteins. As explained in Section 2.5.2, the query sequences were trimmed to the respective mutated domains before running hhblits. trRosetta produces in output predicted interresidue angles and predicted interresidue distances for each pair of positions in the input sequence. The predicted interresidue angles were not used in this work. The predicted distances are represented by trRosetta binned in 36 bins, each collecting distances from 2 Å to 20 Å in bins of 0.50 Å, with an additional bin for no contact. The network is trained to predict the distances among the respective Cβ for each pair of residues (Cα for glycine, which does not have a Cβ atom). The last layer of the network has a softmax activation, and thus the numerical value stored in each output neuron can be interpreted as the probability that a pair of residues are at a distance included in the respective bin.

To obtain a contact map from the trRosetta output, first, the binned distances had to be converted to a single number representing the most probable distance. Firstly, the pairs of residues for which the probability of not being a contact was bigger than the summed probabilities of the remaining 36 distance bins were assigned an infinite distance. Secondly, for those pairs where the above condition was not verified, the most probable distance was obtained as the sum of the bin distances, weighted by the probability of each bin. The bin probabilities were rescaled to the total probability of the 36 distance bins before this operation (the sum of the distance bins probabilities, excluding the non-contact bin). The distance considered for each bin was the central one: for instance, for the bin that spanned 2 Å to 2.50 Å, a distance of 2.25 Å was considered. Finally, the residues at less than 8 Å predicted distance apart were considered contacts.

The contact maps were converted to undirected graphs using the Python library NetworkX. For each residue (node in the connectivity graph), the following metrics were calculated with NetworkX and used as features for the models produced in this work: closeness centrality, betweenness centrality, degree centrality, load centrality, harmonic centrality, clustering coefficient. Closeness centrality was obtained as described in Freeman (1978). Betweenness centrality was obtained as described in Brandes (2001). Degree centrality was obtained by counting the number of edges connected to a node and normalizing this value by $n - 1$, where n is the total number of nodes in the graph. Load centrality was obtained as described in Newman (2001). Harmonic centrality was obtained as described in Boldi et al. (2014). The clustering coefficient implementation used is described in Onnela et al. (2005). Further implementation details can be obtained from the NetworkX documentation at `https://networkx.org/documentation`.

2.5.7 Imputation of Missing Features

The features obtained from EVcouplings were missing for some mutations. This is due to the handling of regions with a poor quality multiple sequence alignment done by the EVcouplings package. For predictors that can deal with missing features (gradient boosted trees), no imputation was done. For predictors that could not deal with missing features (linear regression, support vector machines, neural networks), missing features were imputed using the median feature value of the respective training sets.

2.6 Experimental Validation of Predicted Structural Features

To validate the predicted structural features used in model training, when possible they were compared to the corresponding experimental properties. It should be noted that features derived from experimental protein structures, however, were never used in model training. The features that were validated in this way consisted of the NetSurfP-2 predictions and the trRosetta contact maps.

Table 2.4: PDB structures and chain identifiers used for the validation of predicted features. Table that shows the PDB and chain identifiers used for the validation of the NetSurfP-2 and trRosetta predicted structural features, which were in turn used to train the models presented in this work. Note that the chain identifiers refer to the author chain and not the PDB chain.

Dataset name	PDB ID	Author chain ID
`beta-lactamase`	`1BTL`	`A`
`WW_domain`	`4REX`	`A`
`PSD95pdz3`	`1BE9`	`A`
`kka2_1:2`	`1ND4`	`A`
`hsp90`	`1AH6`	`A`
`Ubiquitin`	`3OLM`	`D`
`Pab1`	`6R5K`	`D`
`E1_Ubiquitin`	`3OLM`	`D`
`gb1`	`2IGD`	`A`

Firstly, experimental protein structures were sourced from the PDB. The PDB structure and chain identifiers used for each dataset are reported in Table 2.4. For the datasets `Ubiquitin` and `E1_Ubiquitin`, the mutagenised domain correspond to the first ubiquitin domain of the yeast UBI4 gene (UniProt ID: `P0CG63`). A structure is not available for this portion of the protein, but it is available for the third ubiquitin domain of the same protein. For this reason, the residues in the PDB structure were first mapped following the SIFTS mappings and then shifted backwards of 304 residues, to overlap with the mutagenised domain. In the dataset gb1, the first Immunoglobulin G (IgG) binding domain of Protein G from *Streptococcus sp.* (UniProt ID: `P06654`) was mutagenised. However, a structure is available only for the second IgG-binding domain of the same protein. For this reason, the residues in the PDB structure were first mapped following the SIFTS mappings and then shifted backwards of 70 residues, to overlap with the mutagenised domain.

The software DSSP was used to assign secondary structures, torsion angles, and solvent accessibility to the residues in the PDB structures. These properties were mapped to the UniProt sequences using SIFTS mappings and the adjustments described in the previous paragraph.

Since the secondary structure assigned by DSSP is categorised in eight classes, for the evaluation of the three-class NetSurfP-2 secondary structure predictions these had to be reduced to a three-class system. The following mapping was used: 3_{10}-helix, α-helix, and π-helix were mapped to helix; β-strands and β-bridges were mapped to strands; the remaining classes were mapped to coil. For the evaluation of the eight-classes secondary structure predictions of NetSurfP-2 instead, the DSSP assignments were used in their original format.

Since DSSP reports only the Accessible Surface Area (ASA) and not the Relative Solvent Accessibility (RSA) for each residue, the relative solvent accessibility had to be calculated to evaluate the quality of the NetSurfP-2 RSA predictions. The DSSP RSA was obtained by dividing the DSSP ASA by the maximum solvation values for each residue type as reported in Ahmad et al. (2003). These values correspond to the ASA of the residue type in an extended tripeptide (Ala-X-Ala) configuration. The NetSurfP-2 predicted ASA was instead compared directly to the DSSP ASA. Note that NetSurfP-2 predicts directly only the RSA, and the ASA is instead calculated from it using the maximum solvation values reported above.

For the evaluation of the trRosetta distograms, distance maps were obtained from the PDB structures by evaluating all the possible pairwise residue distances. The PDB distance maps were then mapped to the respective UniProt sequences following the SIFTS mappings and additionally the adjustments described in this section for the datasets `Ubiquitin`, `E1_Ubiquitin`, and `gb1`. The trRosetta performances were evaluated as the fraction of experimental contacts that are present in the top L/n medium and long-range predicted trRosetta contacts, where n is 5, 2 and 1 and L is the length of the primary sequence from the PDB. Medium range contacts were defined as contacts between residues at 12 or more positions apart in the primary sequence. Long-range contacts were defined as contacts between residues at 24 or more positions apart in the primary sequence. An experimental contact was defined as a pair of residues at a distance of 8 Å or less. Experimental distances were measured between C_β atoms for all residues except glycine, where the C_α atom was taken as reference.

2.7 Validation and Testing Strategy

The general framework followed throughout the project is that of using cross-validation for model optimization on a part of the dataset, and then evaluating the generalization performances on a different test set. In general, the test set was separated from the training dataset before performing cross-validation.

2.7.1 Single Protein Models

Single protein models are models trained on a single deep mutational scanning experiment to predict a holdout proportion of the mutations in the same dataset. Independent models were produced for each of the nine deep mutational scanning datasets used in this work.

Each dataset was first shuffled and then split in half. One half was split again into five cross-validation folds. Once a satisfactory combination of hyperparameters was found, the model was trained again on the full half used for validation. The model was then evaluated on the portion left out for testing.

Two different validation and testing strategies were used, both conforming to the description given above. In the first case, which I refer to as “naive validation and testing”, the mutations in a given dataset were just distributed randomly among training and testing sets, and among the different validation folds. In the second case, care was taken to avoid different mutations of the same residue (same UniProt position) ending up both in the training and testing sets or in different cross-validation folds.

2.7.2 General Models

The general models trained on the aggregated dataset were cross-validated in a Leave-One-Protein-Out (LOPO) fashion, similarly to what was done in Gray et al. (2018).

Differently than in that work, however, in my implementation half of the mutations in the left-out protein at each iteration were used for hyperparameter tuning, while the other half was set aside for testing.

The LOPO cross-validation approach consists of training the model iteratively on all the proteins except one and then using the remaining protein for validation and testing. It should be noted that the left-out portion of the aggregated dataset was selected according to the UniProt identifier, and not according to the deep mutational scanning experiment. This implies that when more than one experiment targeted the same protein, such as in the case of ubiquitin, all of those experiments ended up in the same fold.

The separation of validation and testing data on the holdout proteins was done in such a way to avoid that different mutations affecting the same protein positions ended up in the same set.

Hyperparameter tuning and test performance evaluation were conducted in an aggregated manner for all the proteins, not in a protein-specific way.

2.8 Model Architectures and Optimization

Different models were tested for the general configuration described in Section 2.7.2, while for the single protein models only gradient boosted trees were used. This section describes the optimization strategy used for each model type, and other custom configurations used.

The optimization of hyperparameters was done by cross-validation and random search. The number of sampled configurations varied for different models. Many rounds of random search were usually performed testing different hyperparameters or narrowing down the range for specific hyperparameters until a satisfactory combination was found.

2.8.1 Linear Regression

A simple linear regressor without any free hyperparameter and regularization was deployed using the implementation available in the Scikit-learn Python library (the class `sklearn.linear_model.LinearRegression` was used). Linear regression was used only for the general model. To make the results from different datasets more comparable, the fitness scores were quantile-normalized before being fed to the predictor as explained in Section 2.4.4.

2.8.2 Gradient Boosted Trees

Gradient boosted trees were implemented with the XGBoost Python library. For more details on the implementation and available parameters, the reader can refer to the library documentation at `https://xgboost.readthedocs.io`.

To speed up training, the histogram version of the implementation was used (`tree_method` set to `hist`). For single protein models (Section 2.7.1), the loss function used was the minimum squared error (`objective` set to `reg:squarederror`). For the general model, a pairwise ranking loss was used (`objective` set to `rank:pairwise`).

The hyperparameters that were optimized in gradient boosted trees where the following (in parenthesis the name of the parameter in the XGBoost implementation): maximum depth (`max_depth`), minimum child weight (`min_child_weight`), fraction of subsampled datapoints (`subsample`), fraction of subsampled columns (`colsample_bytree`), learning rate (`eta`), minimum loss reduction required to make a further partition on a leaf node of the tree (`gamma`), L2 regularization term on weights (`lambda`), L1 regularization term on weights (`alpha`), number of iterations (`num_rounds`).

It was observed in preliminary trials that the larger the number of iterations the better the predictions (with diminishing returns for very large values). Because of this, the number of iterations was fixed to the largest value that allowed to train a model in a reasonable amount of time. This value was determined to be $1 \cdot 10^3$ iterations for the random search, and it was increased to $1 \cdot 10^4$ iterations for testing (when estimating both the final validation and testing performances).

The random search was started with a very broad range for each hyperparameter and then repeated once the most promising values were determined. This was iterated several times until no further improvement could be reached. For some of the models, the regularization terms were optimized separately from the other hyperparameters, since for many combinations of the regularization terms the model failed to yield a measurable performance (constant predictions). This led to discarding most of the results, and increasing the search space such that a satisfactory number of observations could be obtained was not practical with the available resources. In those cases first, the regularization terms were left to default values, and the remaining hyperparameters were optimized. Once optimal values were found, the regularization terms were optimized while the other hyperparameters were set to the optimal values. Finally, the learning rate and the number of iterations were examined again by plotting learning curves. If necessary, the learning rate was adjusted.

For each random search iteration, around 2000 configurations were sampled. The actual number varied according to the number of available nodes in the supercomputer cluster used for the computation. The random search was parallelized with MPI for Python, such that different combination of hyperparameters could be tested at the same time. This allowed me to take advantage of the large number of cores available in the supercomputer cluster, and the embarrassingly parallel characteristic of the random search problem. The number of sampled configurations was adjusted to maximise resource usage and avoid that some nodes waited for others to complete their computations (it was adjusted to be an integer multiple of the number of cores available).

2.9 Performance Evaluation

Performance was evaluated in terms of correlation between the predicted and ground truth values. For single protein models, the Pearson correlation coefficient was used.

In cross-validation, the performance was not calculated for each fold but on the aggregated predictions. That is, the model was trained with all but one fold and used to predict the remaining one. The predicted fitness values were stored, and the model was trained again leaving out a different fold. The predictions on this fold were concatenated to the ones obtained before. This was repeated until the data points on all the folds had a prediction associated with them. Then, performance metrics were evaluated on these aggregated predictions. This evaluation strategy was preferred to a fold-specific metric since having a larger number of predictions allowed for greater confidence in the observed correlations.

When testing, the training part of the dataset was first cross-validated with the given combination of hyperparameters and the aggregated predictions for all the folds were evaluated. The model was trained again on the full training set and used to predict the test set. The performance of the test predictions was evaluated, and the validation and test scores were reported. The comparison of validation and test scores allowed me to estimate if the model was overfitting the validation set.

2.10 Comparison with Other Variant Effect Predictors

A set of state-of-the-art quantitative variant effect predictors were selected: the supervised gradient boosting tree model Envision (Gray et al., 2018) and the unsupervised EVcouplings epistatic and independent models (T. A. Hopf et al., 2018).

A notable quantitative model that was not included in this comparison is the unsupervised variational autoencoder DeepSequence (Riesselman et al., 2018). It was evaluated the possibility to include it in this comparison and also to feed its predictions as a feature for my models, but the extensive computational resources required for training the protein-specific DeepSequence models prevented me from doing so.

For Envision, since the model was trained on the same training set adopted in this work, using directly the predictions from the Envision webserver would have been an unfair comparison. For this reason, the cross-validation performances of Envision reported in Gray et al. (2018) are shown. Since the original work does not report the Kendall correlation coefficients, only the Pearson and Spearman correlations are reported. The performances obtained by Envision with rescaled fitness scores were chosen.

Independent and epistatic EVcouplings predictions were already used as features, and the procedure followed for their acquisition is detailed in Section 2.5.4. The respective predictions were directly compared to the experimental, unscaled fitness scores.

For my predictors, the performances are the ones measured on the respective testing sets. To allow for a more meaningful comparison of the scores, the performances for my model were evaluated only on the mutations for which a prediction could be obtaned with EVcouplings. This criterion was not applicable for Envision since I did not have access to the actual predictions, but only to the declare performances.

2.11 Evaluation of Feature Importance

To assess which features contributed more to the performance of the models developed in this work, their permutation importance was evaluated.

Given that some of the features used are collinear, they were not evaluated independently but by grouping them into nine groups. This was needed since, if one feature is removed but another highly correlated feature is still present, the model would still be able to use the remaining feature to obtain the same information. The same would be true in reverse when the second feature is removed and the first one left in. This would lead to the faulty conclusion that both features are not important, while actually, they are important but only when the correlated feature is not present.

The identity of the wild-type and mutant amino acid were evaluated independently and not grouped. Also, the NetSurfP-2 predicted disorder was evaluated independently. The Q_3 and Q_8 secondary structure predictions obtained with NetSurfP-2 were grouped. All of the trRosetta-derived centrality metrics were grouped. All the PSSM scores for the 20 standard residue types, the PSSM scores for the wild-type and mutant residues, and their difference were grouped. All the EVcouplings-derived features were grouped. The last two categories were composed of the NetSurfP-2 predicted torsion angles, and of the relative and absolute NetSurfP-2 predicted solvent accessibility.

For each feature group, the following procedure was used for evaluating the permutation importance. First, the baseline performance $\hat{p}$ of the model on the test set was determined, as explained in the sections relative to each of the models. Then, one feature group at a time, the columns corresponding to that group of features in the test set were randomly permutated. The same model was used again to predict the fitness scores using this corrupted feature set, measuring a group-specific performance p_{jk} for feature group j.

The process was repeated $K = 5$ times for each feature group using different random permutations of the columns. Each repetition is referred to with the subscript k in this notation. The importance i_j for feature group j was then evaluated according to Equation (2.2).

$$i_j = \hat{p} - \frac{1}{K} \sum_{k=1}^{K} p_{jk} \tag{2.2}$$

In words, the performance on the corrupted dataset for a given feature group across the 5 repetitions was averaged and the average was subtracted from the original score of the model. Thus, a feature importance of 0 indicates that the feature group is irrelevant for the model since when the values of the respective features are randomized the performance of the model is not altered. An importance score equal to the performance of the original model indicates that the model entirely depends on that feature group since when it is randomized the performance drops to 0. A negative importance score indicates that the model performs better than the baseline when the feature group is randomized. The feature importance was evaluated independently for each of the nine datasets used in this work.

2.12 Determination of the Statistical Significance of the Results

For the evaluation of the statistical significance of the performance differences among the general models and EVcouplings, a permutation test was performed. The significance of the performance differences with Envision was not evaluated since for this predictor I did not have access to the predictions themselves, but only to the correlation scores reported by the Envision authors. The linear models, gradient boosted tree models, and EVcouplings were compared in a pairwise fashion for each of the nine datasets, for a total of 27 comparisons.

The permutation test was devised so to test the null hypothesis that there is no difference in correlation among each pair of models. Since the performances were evaluated in terms of the Spearman correlation coefficient and since the models used different normalizations for the experimental fitness scores, first the predicted and the experimental fitness scores were transformed into ranks. 10 000 permutations of the predicted ranks were obtained for each pair of models. A single permutation consisted of exchanging a variable number of predicted ranks among the two models. Each pair of predictions had a probability of 0.50 of being swapped. The original predictions were included in the 10 000 permutations, so 9999 permutated predictions were obtained. For each set of permutated predictions, the difference in their Spearman correlation with the experimental ranks was calculated. To obtain two-tailed p values, the fraction of Spearman correlation differences among permutated predictions that had an absolute value greater or equal to that of the actual predictions were determined.

The significance of the performance differences was tested at a p value threshold of 0.05. Since multiple significance tests were performed, a Bonferroni correction (Dunn, 1961) was applied to the threshold. The corrected significance threshold is $\alpha = \frac{0.05}{27} = 0.00185185$.

A permutation test similar to the one described above was not possible for the comparison of validation and testing strategies. Since the testing strategies were different the testing sets were not identical for the models, and so a permutation of the predictions would have been not useful. Instead, I opted for providing a confidence interval for the measured performances by bootstrapping the predictions (Davison, 1999). For each dataset and each model, the test set prediction were resampled with replacement 10 000 times. Performances were calculated on the bootstrapped data, defining a 95 % confidence interval.

Chapter 3

Results

3.1 Dataset Overview

The aggregated data used in this study, after preprocessing and filtration (Section 2.4), consisted of 21 842 mutations. Nine independent experimental datasets were included, covering eight different proteins from bacteria, yeast, rat, and humans. Table 2.1 reports a summary of each dataset, while Table 2.2 reports the number of mutations in each of them before and after filtering. Each dataset consists of a set of single amino acid variants associated with a fitness score.

The fitness scores have a distribution which is extremely dataset-specific (Figure 3.1a) and the number of mutations in each dataset is quite variable (Figure 3.1b). Synonymous mutations tend to have scores that are close to 0 (indicating neutral effect), but some variability is observed. This could be potentially due to differences in codon preferences. The distribution of fitness scores is heavily not normal, with a large number of outliers. A more or less pronounced bimodal distribution is observed for all datasets (Figure S.2), with the highest peak close to 0 and a second, smaller peak towards negative values (detrimental effect). The distributions are strongly asymmetrical, skewed towards negative effects. The datasets are of different sizes, with `beta-lactamase`, `hsp90`, and `WW_domain` having many more mutations than the other ones.

A comparison of fitness scores for identical mutations in the datasets `Ubiquitin` and `E1_Ubiquitin` (which refer to the same protein but examined with different experimental approaches) is reported in Figure 3.2b. A relatively low correlation between the fitness score from these two experiments is observed (Pearson r=0.58). There is an agreement between the experiments on the effect of most quasi-neutral mutations. Synonymous mutations are consistently assigned neutral effects in both datasets. A cluster of mutations with a unanimously disruptive effect can be also observed. Mutations with intermediate effect have more variability in their experimental score.

The unbiased nature of deep mutational scanning experiments is testified by the fact that the frequency of occurrence of mutations towards each of the 20 standard residues is very similar (Figure 3.2a). The overall composition of the aggregated data in terms of wild-type residues for each mutation is consistent with the residue composition of the wild-type sequence of the mutagenized protein regions (Figure 3.2c). Thus, no observable bias is present in the number of mutations originating from different wild-type residues. This holds also on a per-experiment basis (Figure S.3). The aggregated residue composition of the wild-type sequences is similar to that of the whole UniProt (Figure 3.2c), even though there is consistent variability among the nine constituent experiments (Figure S.3). This is expected since single proteins can vary significantly in residue composition (for instance, membrane proteins tend to have a larger fraction of apolar residues than soluble proteins).

The median effect of mutations is overwhelmingly negative (destabilizing) for mutations towards and from all the 20 standard residue types and in all datasets. Proline has the strongest disruptive effect, consistently with the perturbating effect that this residue has on secondary structures (Figure 3.2d). Cysteine and tryptophan are the wild-type residues that are hardest to replace. A breakdown of the median scores per residue and per dataset is reported in Figure S.4.

The number of sampled mutations per position along the primary sequence was examined in Figure S.5. Most datasets show almost full coverage of the mutational landscape for the respective mutagenized regions. Synonymous mutations are not reported in each dataset, but for those that do report them, they are uniformly distributed along the primary sequence (except for `Pab1`, where synonymous mutation scores are available only for a limited region). The dataset `WW_domain` has a lower mutational landscape coverage compared to the other experiments.

3.2 Features Overview

This section explores the relationship between the features used by the models presented in this work and the experimental fitness scores from deep mutational scanning. Numerical quantification of the strength of the relationship between each of the features and the fitness scores is reported in Table 3.1. For the features that are themselves predicted quantities, the agreement between them and their experimental counterpart is reported in Table 3.2 and Table 3.3.

3.2.1 Mutation Identity

Figure 3.3a shows the average scaled fitness score for mutations towards and from any possible combination of residue types.

The identity of a mutation is, as expected, informative for the associated fitness score. Mutations towards proline are universally more damaging than mutations towards any other residue type. Mutations from tryptophan and to a lesser extent cysteine are particularly damaging, confirming the observations made in Section 3.1. The aliphatic residues

Table 3.1: Relationship between features and fitness scores. Table showing various metrics that delineate the dependency between the features used in this work and the corresponding experimental fitness scores. Fitness scores were scaled before computing any of the reported statistics to minimize the effect of having a potentially different experimental range for each dataset. Scaling was done on a per-dataset basis bringing the scores to the range 0 to 1 and subtracting the scaled value for a raw score of 0. For numerical features, the Pearson, Spearman and Kendall correlation coefficients between the features and the fitness scores are shown. For torsion angles, the linear-circular correlation (Mardia et al., 1999) is shown. For categorical features, the test statistics and the p values for the Kruskal-Wallis one-way analysis of variance test (Hollander et al., 2013) are shown.

Feature	Pearson r	Spearman ρ	Kendall τ
PSSM mutation score	-0.29	-0.28	-0.19
Netsurf predicted RSA	0.34	0.37	0.25
Netsurf predicted ASA	0.32	0.35	0.24
Netsurf predicted disorder	0.06	0.18	0.12
EVcouplings epistatic model	0.46	0.50	0.34
EVcouplings independent model	0.44	0.44	0.30
EVcouplings frequency	0.19	0.35	0.24
EVcouplings conservation	-0.32	-0.33	-0.23
Closeness centrality (trRosetta predicted contacts)	-0.16	-0.17	-0.11
Betweenness centrality (trRosetta predicted contacts)	-0.20	-0.29	-0.19
Degree centrality (trRosetta predicted contacts)	-0.12	-0.13	-0.09
Load centrality (trRosetta predicted contacts)	-0.20	-0.29	-0.19
Harmonic centrality (trRosetta predicted contacts)	-0.19	-0.20	-0.14
Clustering coefficient (trRosetta predicted contacts)	0.23	0.25	0.17

	Linear-circular correlation
Netsurf predicted ϕ torsion angle	0.01
Netsurf predicted ψ torsion angle	0.02

	Kruskal-Wallis χ^2	p value
Wild-type residue	1482.40	$< 2.20 \cdot 10^{-16}$
Mutated residue	708.53	$< 2.20 \cdot 10^{-16}$
Netsurf predicted Q_3 secondary structure	215.33	$< 2.20 \cdot 10^{-16}$
Netsurf predicted Q_8 secondary structure	351.97	$< 2.20 \cdot 10^{-16}$

Table 3.2: Agreement between the structural features predicted by NetSurfP-2 and their experimental counterparts. Table showing the agreement between the features predicted by the neural network NetSurfP-2 and their experimental counterparts obtained with DSSP. For numerical features, the Pearson correlation coefficient is shown. For categorical features, the multi-class accuracy is shown. For torsion angles, the circular variant of the Pearson product-moment correlation is shown (Jammalamadaka, 2001). Since the experimental values are defined only for a subset of mutations, the reported scores are evaluated on the set of mutations that have both a predicted and an experimental value for the feature in question.

Feature	Evaluation metric	Score
Relative solvent accessibility	Pearson r	0.79
Accessible surface area	Pearson r	0.80
Q_3 secondary structure	Q_3 accuracy	0.85
Q_8 secondary structure	Q_8 accuracy	0.72
ϕ torsion angle	Circular correlation	0.73
ψ torsion angle	Circular correlation	0.87

Table 3.3: Precision of trRosetta in predicting residue contacts. Table showing for each dataset the fraction of the top L/n contacts predicted by trRosetta that are present in the respective experimental PDB structures, subdivided by medium-range and long-range. L is the length of the PDB protein sequence. s is the linear distance among the residues in the primary sequence such that if i is the position of the first residue and j the position of the second residue in a contact along the primary sequence of the protein, then $s = |j - i|$.

	Medium range ($s \geq 12$)			Long range ($s \geq 24$)		
Dataset	Top $L/5$	Top $L/2$	Top L	Top $L/5$	Top $L/2$	Top L
`beta-lactamase`	1.00	0.92	0.86	0.96	0.93	0.76
`WW_domain`	0.95	0.90	0.83	0.90	0.87	0.75
`PSD95pdz3`	0.96	0.92	0.80	0.92	0.81	0.70
`kka2_1:2`	1.00	1.00	0.96	1.00	1.00	0.89
`hsp90`	1.00	1.00	0.96	1.00	1.00	0.89
`Ubiquitin`	0.98	0.92	0.82	1.00	0.90	0.70
`Pab1`	0.80	0.72	0.67	0.87	0.74	0.60
`E1_Ubiquitin`	0.82	0.86	0.77	0.91	0.75	0.54
`gb1`	1.00	0.85	0.46	0.63	0.40	0.22

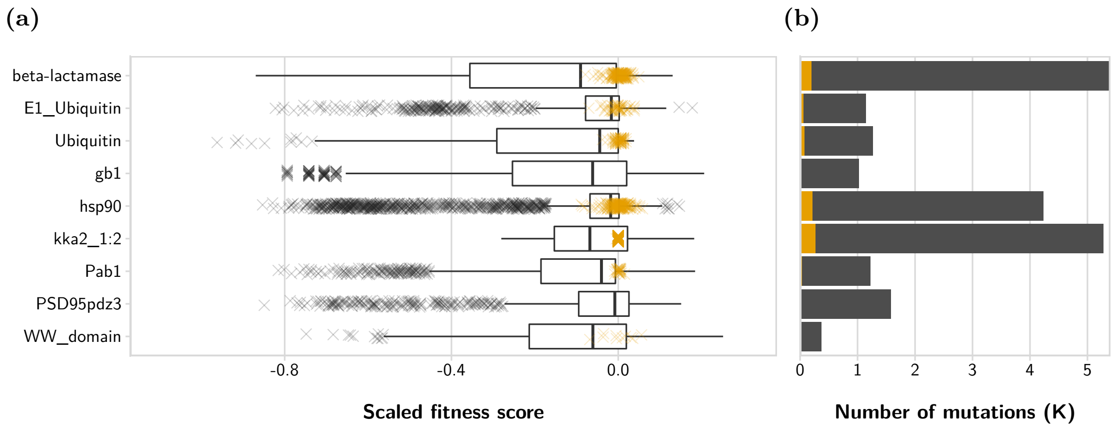


Figure 3.1: Dataset distributions and number of mutations. Nine deep mutational scanning studies were used in this work. To make comparisons more meaningful, the fitness scores were scaled independently for each dataset to the range 0 to 1 and centred such that a neutral mutation would have a scaled score of 0. This scaling was only done for the visualizations and was not used in model development. **(a)** Distribution of fitness scores. Orange crosses are synonymous mutations. Black crosses are outliers. **(b)** Number of mutations in each dataset (in thousands). The orange portion shows the number of synonymous mutations. The vertical axis is shared with **(a)**. For a more detailed representation of the distribution of fitness scores see Figure S.2.

valine, leucine, isoleucine, and methionine can quite freely substitute for each other. The same is true also for the aromatic residues tyrosine, tryptophan, and phenylalanine. Polar and charged residues tend to be easy to substitute, but damaging when inserted in apolar positions. No mutation type has an average positive effect. Synonymous mutations are, as expected, the least damaging ones.

I reasoned that the striking difference in mean fitness score for mutations from polar and charged residue as compared to mutations from apolar residues could be due to the tendency of the formers to occupy positions at the protein surface. This hypothesis was confirmed by filtering the mutations according to the relative solvent accessibility of the position. The difference in fitness scores for mutations from polar and apolar residues disappears when only mutations with relative solvent accessibility greater than 0.15 or only mutations with relative solvent accessibility smaller than 0.15 are considered (data not shown).

When the data is partitioned by dataset (Figure S.6), the observed trends still generally hold. However, it can be noted that some datasets have more pronounced effects than others. This is likely due to the presence of outliers that influence the scaling procedure used for the fitness scores. Mutations from tryptophan are exceptionally damaging in the dataset `gb1`.

The relationship between fitness score and mutation identity closely resembles the scores from a small-distance substitution matrix such as the BLOSUM100 (Henikoff et al., 1992,

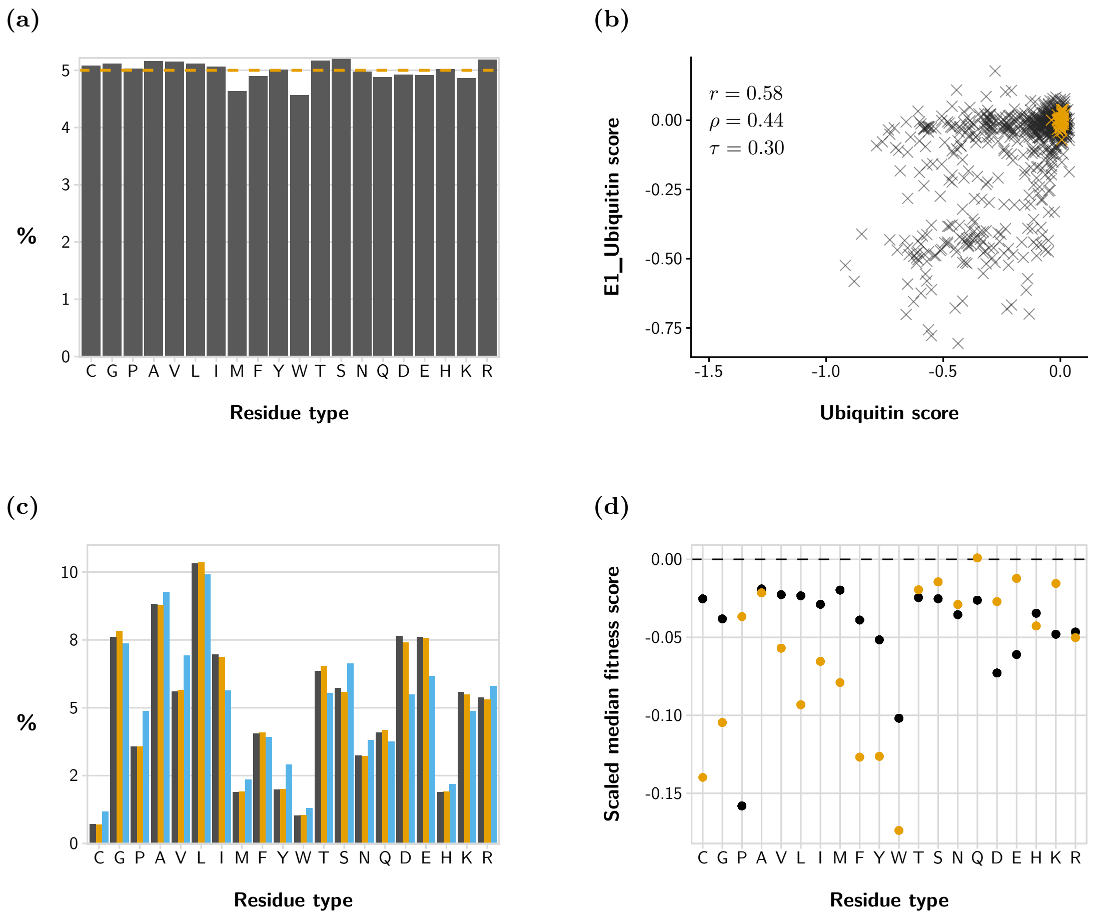


Figure 3.2: Aggregated statistics and correlation among different experiments. (a), **(c)**, **(d)** show aggregated data from the nine datasets used in this work. Before aggregating the data, the fitness scores were scaled independently for each dataset to the range 0 to 1 and centred such that a neutral mutation would have a scaled score of 0. The same data presented individually for each dataset is available in appendix (Figure S.1, Figure S.3, and Figure S.4). **(a)** Fraction of mutations towards each of the 20 standard residues. The orange dashed line marks the expected fraction of mutations in case of uniform frequency. **(b)** Correlation among the fitness scores from two different experiments on ubiquitin. Orange crosses are synonymous mutations. Black crosses are missense mutations. The Pearson (r), Spearman (ρ), and Kendall (τ) correlation coefficients are shown. **(c)** Aggregated composition of the dataset in terms of wild-type residues for each mutation (in grey) compared to the aggregated residue composition of the wild-type sequences of the mutagenized protein regions (in orange) and the residue composition of the whole UniProt (in blue). The UniProt composition refers to the release `2020_01`, sourced from `https://www.uniprot.org/statistics/`. **(d)** Median score of mutations from and to each of the 20 standard residues. Orange dots represent the median score of mutations from the residue. Black dots represent the median score of mutations towards the residue. The black dashed line marks a fitness score of 0, corresponding to a neutral effect.

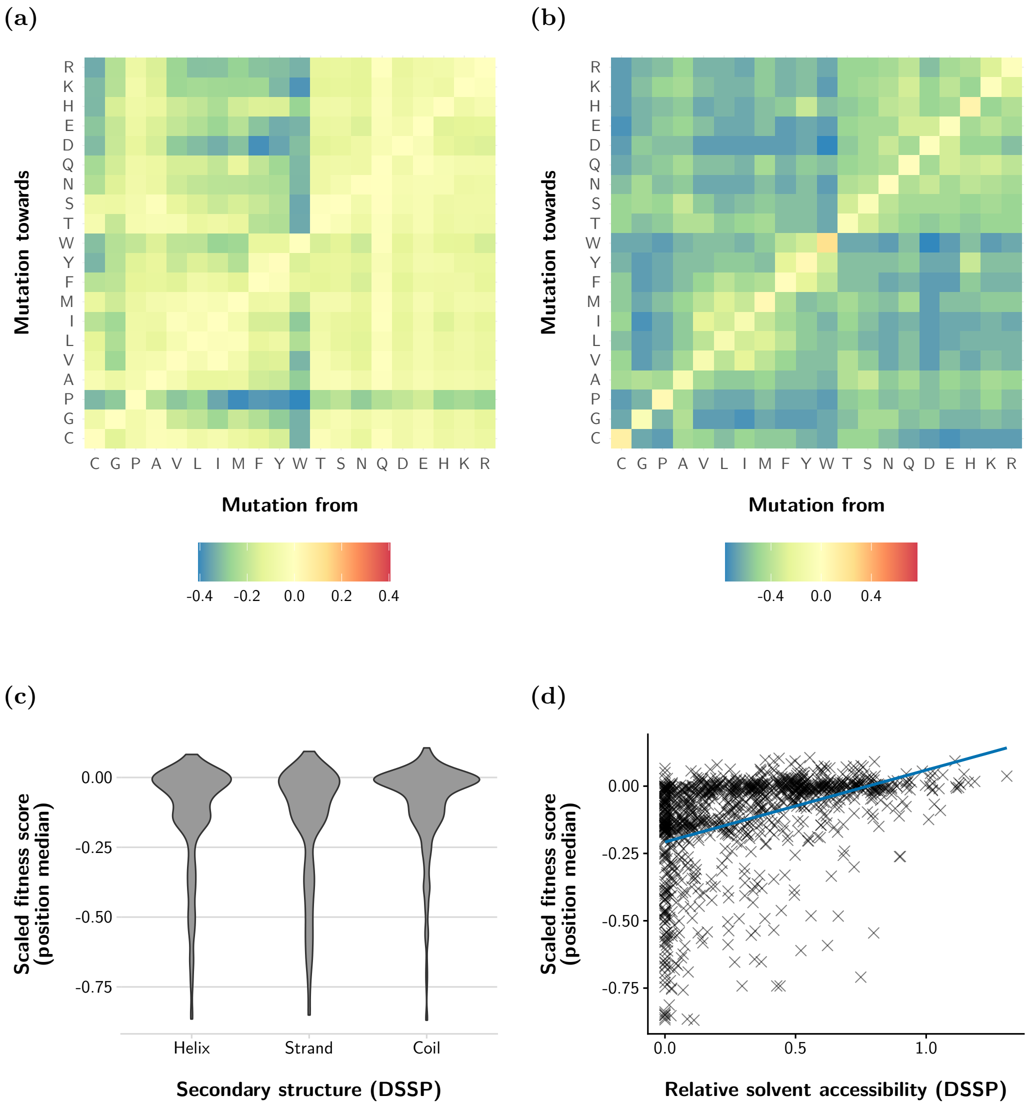


Figure 3.3: Fitness scores by secondary structure, mutation type, and solvent accessibility. For all the plots, fitness scores were scaled to make them more comparable across datasets. Scaling was done on a per-dataset basis bringing the scores to the range 0 to 1 and subtracting the scaled value for a raw score of 0. **(a)** Heatmap showing the average scaled fitness score for mutations towards and from any of the 20 standard residue types. **(b)** Heatmap showing the BLOSUM100 scoring matrix substitution scores rescaled to the range 0 to 1 and centred by subtracting the average rescaled score for a synonymous substitution. **(c)** Violin plot showing the distribution of median scaled fitness scores according to the secondary structure of the mutated position. Only the subset of protein positions with an experimental secondary structure assignment is shown. The assignments were obtained as explained in Section 2.6. **(d)** Scatter plot showing the dependency between the experimental relative solvent accessibility of a mutated position and the median scaled fitness score.

Figure 3.3b). For example, a cluster of permissive scores is present in both heatmaps for mutations among aliphatic residues, aromatic residues, and polar residues. Of course, being the BLOSUM100 a symmetric matrix, the directional patterns observed in the experimental scores are not mirrored. These include the particularly unfavourable mutations towards proline and from cysteine and tryptophan.

3.2.2 Position-Specific Scoring Matrices

The emission probabilities of the hidden Markov models obtained from the multiple sequence alignments of each of the proteins used in this work were used to extract Position-Specific Scoring Matrices (PSSMs). The scores of each mutation from the respective PSSMs have a weak negative correlation to the experimental fitness score (Table 3.1). The directionality of the relationship is negative because the emission probabilities are represented by HMMER with their negative natural logarithm.

3.2.3 Solvent Accessibility

The solvent accessibility used as a feature was not the experimental one, but the predicted quantities obtained with the neural network NetSurfP-2. This was done to make the models independent from the availability of protein structures. Both the Relative Solvent Accessibility (RSA) and the Accessible Surface Area (ASA) were used as features. The Pearson correlation between the predicted and experimental values is 0.79 and 0.80 respectively (Table 3.2), suggesting reliable predictions.

Relative solvent accessibility has a Pearson correlation with the fitness scores of 0.34 (Table 3.1). The positive correlation implies that exposed residues tend to have higher (i.e. less damaging) fitness scores. This is coherent with previous observations (Savojardo et al., 2021).

3.2.4 Secondary Structure

The secondary structure classification used in the models was not the experimental one, but the predicted one obtained with the neural network NetSurfP-2. The actual features used was not the categorical classification, but the raw activation for the output layer of the neural network NetSurfP-2. Both the eight-classes and the three-classes classification were used as features.

The distribution of fitness scores is significantly different according to the secondary structure of the mutated position (Table 3.1). Figure 3.3c reports the aggregated distribution of the median fitness scores (per dataset and position) by secondary structure. Figure S.7 reports the same but detailed by dataset. Mutations that affect residues in a coil conformation tend to be less damaging than mutations affecting strands or helices, coherently with the less stringent chemical requirements for this type of secondary structure. Very strong variability in the relationship between secondary structure and fitness score is observed among datasets, but in general, coils tend to be more permissive and strands more stringent in the type of mutations that are permitted.

The predicted secondary structures from NetSurfP-2 are in good agreement with the experimentally determined ones, where the latter are available (Table 3.2).

3.2.5 Torsion Angles

The relationship between torsion angles of the backbone and fitness scores is negligible (Table 3.1). Also in this case the predicted quantities from NetSurfP-2 were used and not the experimental torsion angles. Nonetheless, the agreement between the two is very strong (Table 3.2).

3.2.6 Disorder

NetSurfP-2 includes in its output also a probability of each protein position being disordered. This probability was trained by considering disordered residues which did not appear in the respective crystal structures of the proteins in the training set of the NetSurfP-2 neural network.

The Pearson correlation of this predicted disorder probability with the fitness scores is weak, while the Spearman correlation is stronger (Table 3.1). The low Pearson correlation can be due to the sigmoid activation for the neuron responsible for this feature in the output layer of NetSurfP-2, which imposes a strong bi-modality in the resulting distribution.

Fitness score and disorder probability are positively correlated, indicating that, as would be expected, disordered residues tend to be more permissive towards mutations.

3.2.7 Evolutionary Couplings

Among all the features used in this work, the epistatic evolutionary couplings obtained with the unsupervised model EVcouplings have the highest correlation with the experimental fitness scores (Table 3.1). The independent couplings are also strongly correlated with the fitness scores, as well as the frequency of the mutation in the multiple sequence alignment of the mutagenized protein, and the conservation of the position. These last two features are also part of the output from the EVcouplings predictor.

As expected, the correlation between the conservation of a position and the corresponding fitness scores is negative, indicating that conserved positions have more dramatic effects when mutated as compared to less conserved positions. On the contrary, the correlation between the other EVcouplings features and the fitness scores is positive, since this predictor was developed for predicting mutational effects.

3.2.8 Connectivity Graphs

To make the models independent from the availability of experimental protein structures, instead of relying on experimental residue contacts the neural network trRosetta was used to obtain predicted inter-residue distances. From these distances, a connectivity graph was obtained for each protein by imposing a threshold of 8 Å distance between C_β atoms to assign contacts (C_α for glycine). For each residue in the protein then, which was represented by a node in the contact graph, a series of node metrics were calculated and used as features.

All the node centrality metrics used have a negative correlation with the fitness scores, indicating that more central residues tend to have more negative fitness scores. This is expected since more central residues tend to be in the core of the protein, where mutations have the most destabilizing effects.

The clustering coefficient on the other hand has a positive correlation with the fitness scores, indicating that residues that are a part of densely connected sub-networks tend to be more permissive towards mutations. This is also intuitive since residues with a very low clustering coefficient could be mediators of key and non-redundant contacts that keep the protein folded. For instance, residues with a low clustering coefficient could be those that make different protein domains interact. However, further research is needed to elucidate the exact reason for this trend.

The predicted contact maps obtained with trRosetta are in excellent agreement with the ones determined from the experimental protein structures. The precision of the top $L/2$ long-range predicted contacts, a common metric used in protein structure prediction, is between 0.40 and 0.94 for the proteins used in this work (Table 3.3).

3.2.9 Missing Features

The features obtained from EVcouplings were missing for some mutations. This is due to the handling of regions with a poor quality multiple sequence alignment done by the EVcouplings package. For the other features, no missing values were present. The handling of missing features is described in Section 2.5.7. The number of mutations that did not have associated EVcouplings features amounted to 1884 on a total of 21 513 mutations, or 8.76 %. The number of missing features by dataset is described in Table 3.4.

3.3 Correlation among Features

The correlation among the features used by the models presented in this work is shown in Figure 3.4.

The EVcouplings predictions are strongly correlated to each other. The correlation is positive among all the EVcouplings features except the position conservation, which is negatively correlated to the other EVcouplings features. This makes logical sense since

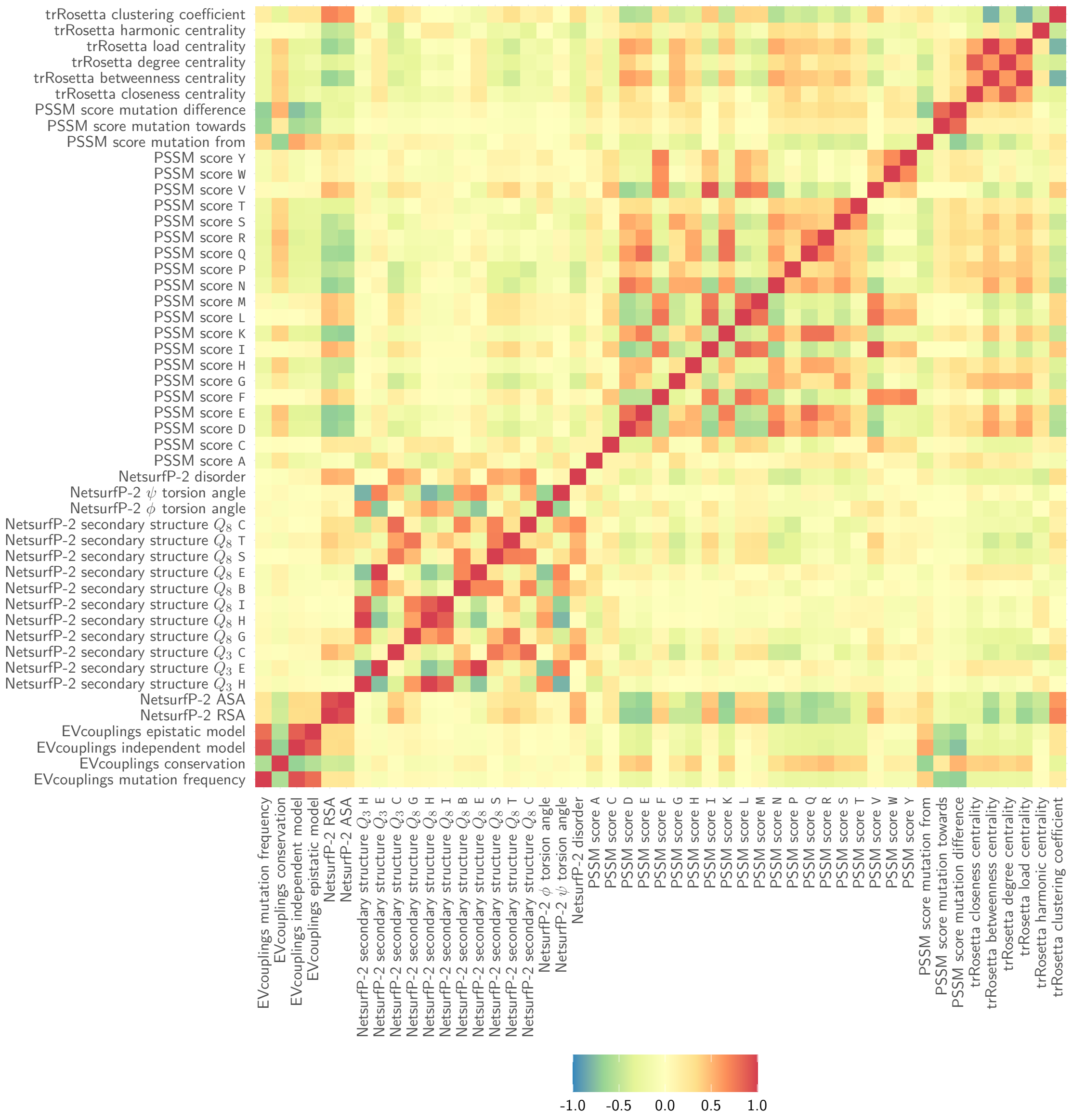


Figure 3.4: Heatmap of the correlation among features. Heatmap showing the Spearman correlation coefficient among the features used for training the models presented in this work. For the NetSurfP-2 secondary structure predictions, the class probabilities (network outputs) are used for evaluating the correlations. The definition of the one-letter code for the secondary structure classes can be found at Kabsch et al. (1983). The feature “PSSM score mutation from” refers to the PSSM score of the wild-type residue in each position. The feature “PSSM score mutation towards” refers to the PSSM score of the mutant residue in each position. The feature “PSSM score mutation difference” is the difference of “PSSM score mutation towards” and “PSSM score mutation from”.

Table 3.4: Number of mutations with missing features. Table that shows the number of mutations in each dataset that could not be assigned EVcouplings predictions. EVcouplings predictions were the only feature that presented some missing values.

Dataset name	Number of mutations with missing EVcouplings features	Total number of mutations	Percentage (%)
`beta-lactamase`	569	5397	10.54
`WW_domain`	10	373	2.68
`PSD95pdz3`	0	1577	0
`kka2_1:2`	625	5280	11.84
`hsp90`	464	4231	10.97
`Ubiquitin`	110	1267	8.68
`Pab1`	49	1220	4.02
`E1_Ubiquitin`	57	1142	4.99
`gb1`	0	1026	0

more conserved positions are expected to have more negative real, and hence predicted, fitness scores.

The NetSurfP-2 RSA and ASA are trivially correlated to each other since ASA is calculated from the RSA and not predicted by NetSurfP-2. The two solvent accessibility features have a weak positive correlation with the EVcouplings score (except the conservation, which shows a negative correlation), weak correlation with the secondary structures, negative correlation with the PSSM scores for polar and charged residues, and positive correlation with the PSSM scores for apolar residues.

NetSurfP-2 secondary structure probabilities are trivially correlated to each other, being different outputs of the same softmax activation layer. Virtually no correlation with the EVcouplings scores is observed, and very low correlation with the other features. As it would be expected, three-classes and eight-classes prediction are strongly correlated.

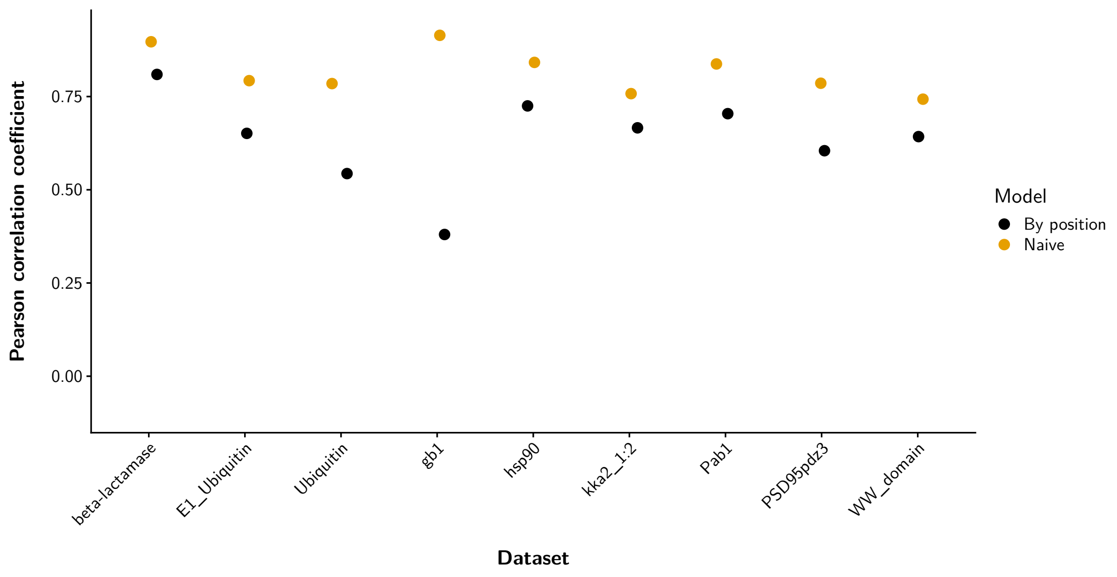
NetSurfP-2 torsion angles and disorder probabilities follow similar correlation patterns as the secondary structures.

The node centrality metrics from the predicted trRosetta contacts are strongly related to each other, except for the harmonic centrality which has a weaker correlation. They are negatively correlated to solvent accessibility, as it would be expected, and positively correlated to most of the PSSM scores. The clustering coefficient follows inverse correlation patterns as compared to the centrality metrics.

3.4 Comparison of Model Testing Strategies

The aim of this section is the evaluation of the influence of the validation and testing strategy used on the perceived model performances. To isolate the effect of the validation strategy from that of the model type, only models based on gradient boosted trees are

(a)



(b)

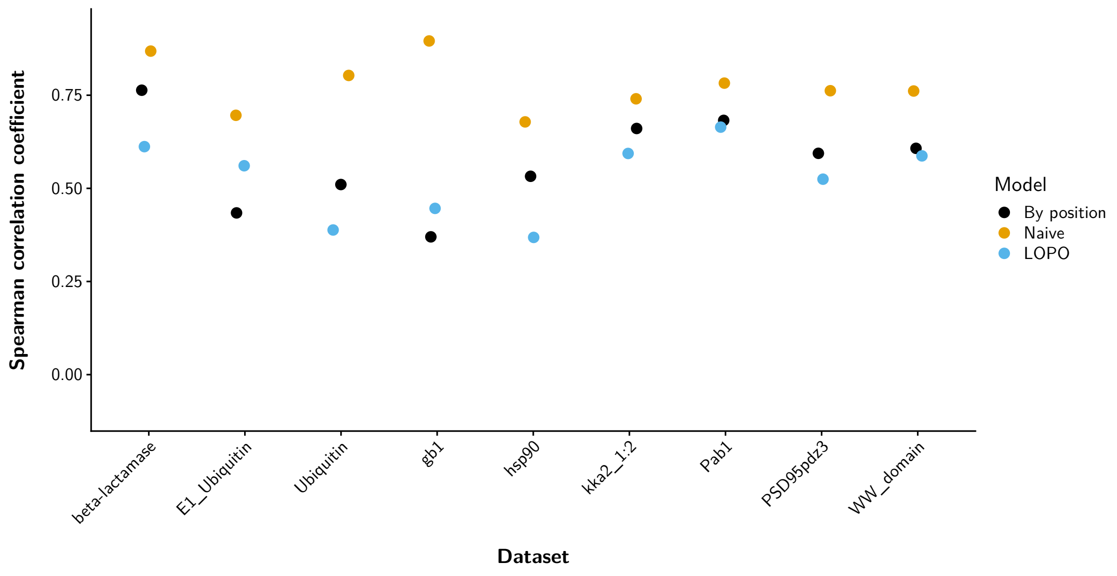


Figure 3.5: Comparison of model testing strategies. Pair of plots showing the correlation among experimental and predicted fitness scores in the testing set for gradient boosted tree models, subdivided by dataset. **(a)** Comparison of single protein models trained under the naive paradigm and by segregating protein positions. **(b)** Comparison of the naive single protein models and protein models trained by segregating protein positions against the gradient boosted tree general model (LOPO). Since the general model was trained using a ranking loss function, the Spearman correlation coefficient was preferred in this comparison.

Table 3.5: Confidence intervals for the performances of the gradient boosted tree models. Table that shows the 95 % confidence intervals (C.I.) determined by bootstrapping for the performances of the models based on gradient boosted trees developed in this work. The performances were measured in terms of the Pearson and Spearman correlation coefficients among experimental fitness scores and predicted fitness scores.

Dataset name	Model	95 % C.I. (Pearson)	95 % C.I. (Spearman)
`beta-lactamase`	Naive	0.89 to 0.91	0.86 to 0.88
`beta-lactamase`	By position	0.79 to 0.83	0.75 to 0.78
`beta-lactamase`	LOPO	—	0.68 to 0.72
`WW_domain`	Naive	0.67 to 0.82	0.70 to 0.84
`WW_domain`	By position	0.57 to 0.73	0.52 to 0.72
`WW_domain`	LOPO	—	0.56 to 0.73
`PSD95pdz3`	Naive	0.74 to 0.83	0.73 to 0.80
`PSD95pdz3`	By position	0.55 to 0.67	0.54 to 0.65
`PSD95pdz3`	LOPO	—	0.53 to 0.63
`kka2_1:2`	Naive	0.74 to 0.78	0.72 to 0.76
`kka2_1:2`	By position	0.65 to 0.69	0.64 to 0.68
`kka2_1:2`	LOPO	—	0.60 to 0.64
`hsp90`	Naive	0.82 to 0.87	0.65 to 0.71
`hsp90`	By position	0.69 to 0.76	0.50 to 0.57
`hsp90`	LOPO	—	0.38 to 0.45
`Ubiquitin`	Naive	0.75 to 0.83	0.78 to 0.83
`Ubiquitin`	By position	0.49 to 0.60	0.46 to 0.57
`Ubiquitin`	LOPO	—	0.30 to 0.43
`Pab1`	Naive	0.80 to 0.87	0.75 to 0.82
`Pab1`	By position	0.65 to 0.76	0.64 to 0.73
`Pab1`	LOPO	—	0.60 to 0.70
`E1_Ubiquitin`	Naive	0.75 to 0.85	0.65 to 0.75
`E1_Ubiquitin`	By position	0.59 to 0.72	0.36 to 0.51
`E1_Ubiquitin`	LOPO	—	0.50 to 0.63
`gb1`	Naive	0.90 to 0.93	0.88 to 0.92
`gb1`	By position	0.31 to 0.46	0.29 to 0.45
`gb1`	LOPO	—	0.32 to 0.47

considered in this section. Three different approaches to testing and optimization were adopted: in two cases independent models were trained for each dataset (single protein models), while in the remaining case a single model was trained on all the available data.

The single protein models were trained either randomly setting aside half of the mutations in the respective datasets for testing (referred to as “naive” models), or by segregating protein positions, such that different mutations that affected the same residue would be grouped (referred to as models “by position”).

For the general models trained on all the nine datasets used in this work, a Leave-One-Protein-Out (LOPO) approach to validation and testing was used. Briefly, one of the eight proteins included in the datasets was left out of the training set and used for validation and testing. Half of the mutations in the left out protein were used for validation and half for testing, splitting the sets by segregating protein positions. The process was repeated changing which protein was left out until all proteins had been tested. This cross-validation and testing approach was based on grouping mutations by protein, not by dataset. This difference is important since two of the nine datasets used concerned the same protein, ubiquitin. The performance evaluation however was reported on a per-dataset basis, for compatibility with the single protein models and for assessing dataset-specific performance patterns. More details on the methods used for validation and testing are reported in Section 2.7.

The three model groups were optimized independently with a random search procedure (Section 2.8). The implementations of the three models were identical, except that for the single protein models a minimum squared error loss was used, while for the general model a pairwise ranking loss was used. This was made necessary by the high diversity in range and distribution of fitness scores among different datasets. For the same reason, the Pearson correlation coefficient was preferred when comparing only single protein models, and the Spearman correlation coefficient was preferred when comparing general models.

The single protein models trained by segregating protein positions in different splits for validation and testing performed sensibly worse than the ones trained with the naive approach (Figure 3.5a). This was expected since the naive models are likely to overfit the validation and testing sets because of the presence of different mutations affecting the same protein position in both training and testing. Likely, knowledge of the mutational pattern towards certain residues at a specific position gives valuable clues about the fitness of different mutations in the same position.

The single protein models trained by segregating protein positions perform on average better than the general gradient boosted tree model, but the difference is smaller than with the naive single protein models (Figure 3.5b). This is remarkable since it shows that a general model, trained to predict mutation effects from a range of different proteins, can reach similar performances to a model specifically trained for a single protein. The main determinant of performances in fitness prediction seems to be the availability in the training set of mutations affecting the same residues used for testing, not merely the presence during training of mutations affecting the same protein, or optimization of the model towards a specific dataset. In light of these results, it seems reasonable to caution against the overestimation of performances in models that use a naive approach to testing.

For a more detailed representation of the test results for the three model groups compared here see Figure S.10, Figure S.11, and Figure S.10. Confidence intervals for the performance estimations are reported in Table 3.5.

3.5 Comparison of Different Model Architectures with Previous Predictors

This section solely concerns the general models trained in a LOPO fashion, and how they do compare with previously published predictors. Figure 3.6 shows the performance of two state-of-the-art mutation effect predictors compared to that of the general models developed as part of this work. The statistical significance of the performance differences among the predictors was calculated according to Section 2.12 and is reported in Table 3.6.

The models that I tested in a LOPO fashion include gradient boosted trees, a simple linear regression, support vector machines, and neural networks. However, support vector machines and neural networks in preliminary testing phases failed to improve on the linear regression performances and thus were excluded from further optimization and this comparison. Gradient boosted trees, on the other hand, were included in the comparison even though they did not provide significant improvements on the linear regression. This was decided to allow for better comparison with the single protein models, which were also based on gradient boosted trees. In addition, gradient boosted trees have the advantage compared to linear regression to allow for the use of a ranking loss and they are insensitive to feature scaling.

In light of the test results, the use of a more complex model such as gradient boosted tree does not seem justified when a much simpler model such as unregularized linear regression can attain similar performances, while at the same time diminishing training times and minimising the risk for overfitting. Nonetheless, gradient boosted trees could be preferred to dispense from the need for scaling features and normalizing labels.

The comparison of the models developed in this work to two state-of-the-art predictors, Envision and EVmutation, shows comparable performances. Envision is also a gradient boosted tree-based model, but it is based on a different feature set that includes also structural data. EVmutation is an unsupervised model whose output is also included as a feature in the models that I developed. Other notable variant effect predictors were excluded from the comparison for their focus on binary classification instead of quantitative scores, or the excessive computational requirements.

My models generally improve on the raw EVmutation performances, showing that they are not simply echoing the EVmutation output. The improvement is particularly striking

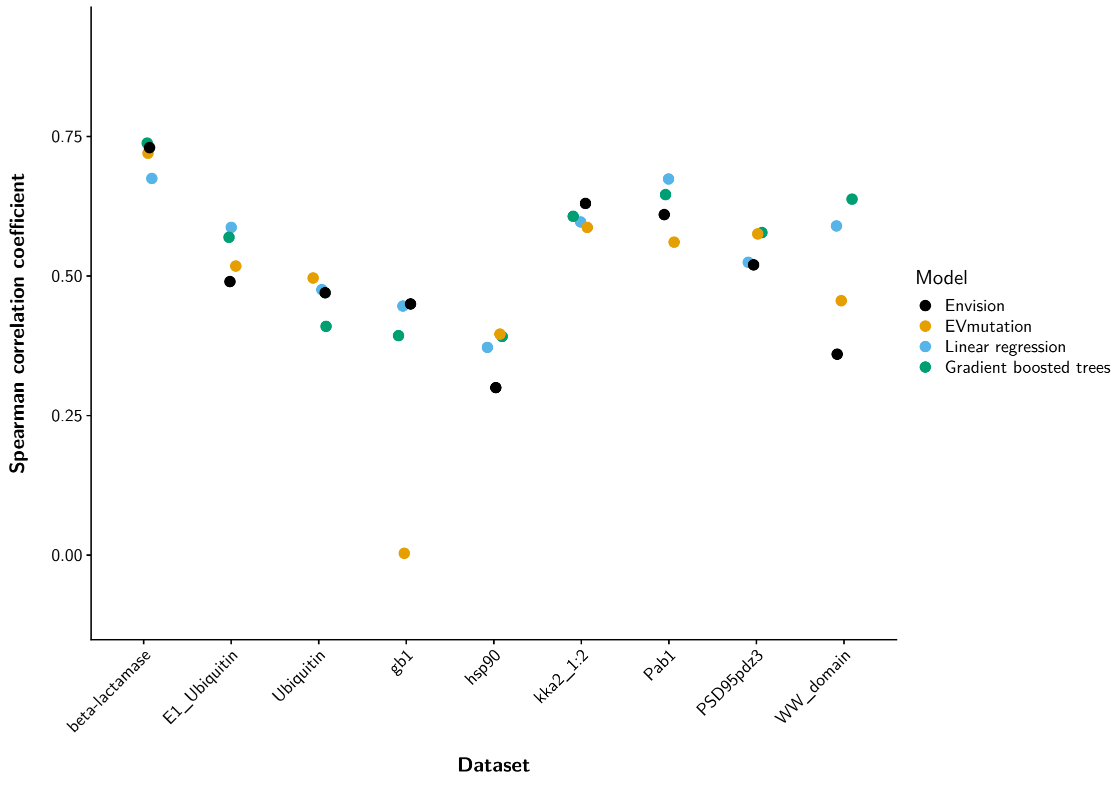


Figure 3.6: Comparison of performances for various quantitative predictors of single amino acid variant effect. Plot showing the Spearman correlation coefficients between the predicted effect and the experimental measurements obtained with deep mutational scanning for the nine datasets used in this work. The predictions of two state-of-the-art methods are shown alongside the performances of the methods developed in this work.

for the dataset `gb1`, where EVmutation predictions are very poor. In general, EVmutation seems to depend on the quality of the multiple sequence alignment more than the models that I developed. Indeed, the dataset `gb1` has a poor multiple sequence alignment because of the lack of enough homologous sequences in UniProt.

Compared to the similar model Envision, my models perform better on some of the datasets and worse on others. It should be noted, however, that my models do not use any structural feature. Also, Envision performance are cross-validation scores measured on the same set used for hyperparameter tuning, while the performances of my models are measured on the test set. Thus, it is likely that the perceived performances of Envision are an overestimation.

For a comparison of the validation and test scores of my models, refer to Figure S.9.

Table 3.6: Significance of the performance differences for various quantitative predictors of single amino acid variant effect. Table that shows the significance of the test-set performance differences observed among the models trained in LOPO fashion and EVmutation. Significant p values are denoted with an asterisk (Bonferroni-corrected $\alpha = \frac{0.05}{27} = 0.00185185$). The null hypothesis is that the performance difference among each pair of models is equal to 0. Envision was excluded from this table since I did not have access to the model predictions but only to the performances declared by the authors of the method.

Dataset name	Model 1	Model 2	p value
`beta-lactamase`	Linear regression	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`beta-lactamase`	Linear regression	EVmutation	$1 \cdot 10^{-4}$ *
`beta-lactamase`	EVmutation	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`WW_domain`	Linear regression	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`WW_domain`	Linear regression	EVmutation	$1 \cdot 10^{-4}$ *
`WW_domain`	EVmutation	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`PSD95pdz3`	Linear regression	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`PSD95pdz3`	Linear regression	EVmutation	$1 \cdot 10^{-4}$ *
`PSD95pdz3`	EVmutation	Gradient boosted trees	0.51
`kka2_1:2`	Linear regression	Gradient boosted trees	0.00
`kka2_1:2`	Linear regression	EVmutation	0.01
`kka2_1:2`	EVmutation	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`hsp90`	Linear regression	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`hsp90`	Linear regression	EVmutation	$1 \cdot 10^{-4}$ *
`hsp90`	EVmutation	Gradient boosted trees	0.24
`Ubiquitin`	Linear regression	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`Ubiquitin`	Linear regression	EVmutation	$1 \cdot 10^{-4}$ *
`Ubiquitin`	EVmutation	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`Pab1`	Linear regression	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`Pab1`	Linear regression	EVmutation	$1 \cdot 10^{-4}$ *
`Pab1`	EVmutation	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`E1_Ubiquitin`	Linear regression	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`E1_Ubiquitin`	Linear regression	EVmutation	$1 \cdot 10^{-4}$ *
`E1_Ubiquitin`	EVmutation	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`gb1`	Linear regression	Gradient boosted trees	$1 \cdot 10^{-4}$ *
`gb1`	Linear regression	EVmutation	$1 \cdot 10^{-4}$ *
`gb1`	EVmutation	Gradient boosted trees	$1 \cdot 10^{-4}$ *

3.6 Gradient Boosted Tree Optimization

The random hyperparameter search adopted for the gradient boosted trees-based models showed similar patterns of dependency on the hyperparameters for the single protein models and the models trained in a LOPO fashion. In general, more iterations corresponded to better results for all the models, but with diminishing returns. From the learning curves (Figures 3.7 and 3.8) it can be seen how small learning rates consistently lead to better performances, but the number of iteration required to reach plateau is greater for small learning rates. If the learning rate is not too large ($\leq 1 \cdot 10^{-2}$) it is always possible to reach optimal performances, but smaller learning rates require more iterations to do so. For the single protein models and with a learning rate of $1 \cdot 10^{-3}$ or $1 \cdot 10^{-4}$, a peculiar learning curve is observed where the performances seem to plateau around 100 iterations but then increases again. The reason for this is not clear. This pattern is not observed for the learning curves of the LOPO general model, which appear less regular. In the general LOPO models, the performances start to deteriorate after $1 \cdot 10^{4}$ iterations.

Other hyperparameters besides the learning rate and the number of iterations show a very modest impact on performances. The L1 and L2 regularization terms proved to be very damaging if set to large values, as can be expected. Subsampling the rows and columns seemed to be effective in reducing overfitting while at the same time making training faster. Enforcing a minimum loss reduction, a minimum child weight, and a maximum tree depth had little impact, besides being damaging if set to extremely large values.

For a comparison of testing and validation performances refer to Figure S.12. For a more detailed report of the influence of the various hyperparameters refer to Figure S.13, Figure S.14, and Figure S.15.

3.7 Feature Importance

The importance of the features used by the models developed in this work was evaluated according to the grouped permutation importance (Section 2.11). Nine feature groups were defined, according to the collinearity of the features. The importance was independently evaluated for each dataset.

In the gradient boosted tree models, the importance profile is similar for models trained with the two single protein strategies (Figure S.16 and Figure S.17), and for the models trained in a LOPO fashion (Figure 3.9). The EVcouplings feature group is by far the most important. In some datasets (`E1_Ubiquitin`, `Ubiquitin`, `WW_domain`, `Pab1`, `PSD95pdz3`, `hsp90`) also the PSSM score are strong contributors to performance, while in other datasets they are less important (`beta-lactamase`, `kka2_1:2`), or even detrimental (`gb1`). It is interesting how the PSSM scores appear to be detrimental for the `gb1` dataset in the single protein models trained by segregating protein positions and in the LOPO general models, while they are important in the naive single protein models. In the LOPO models, the PSSM scores are also more important than in the single protein models for

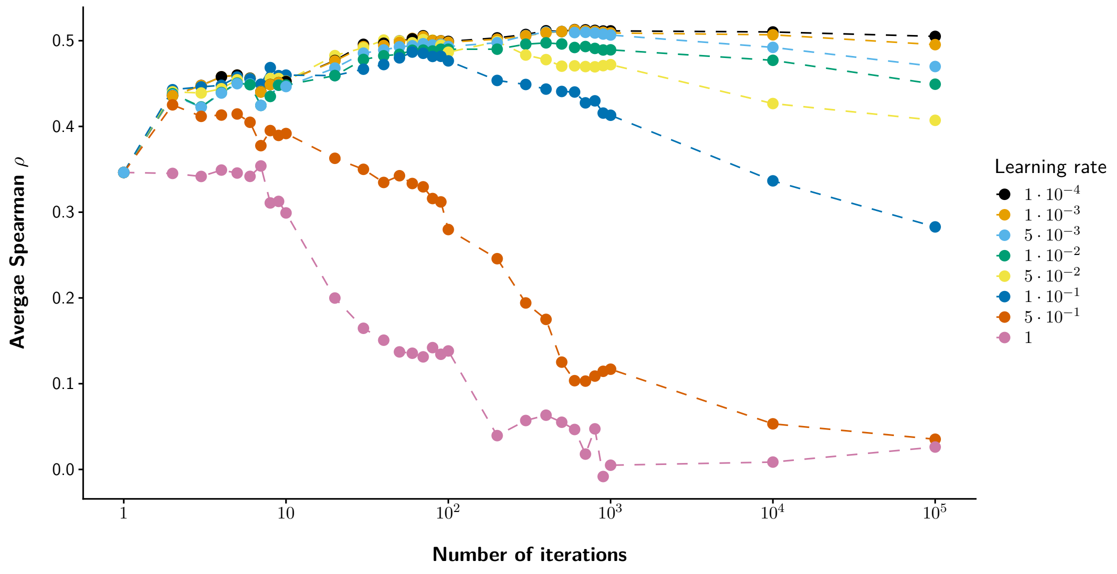
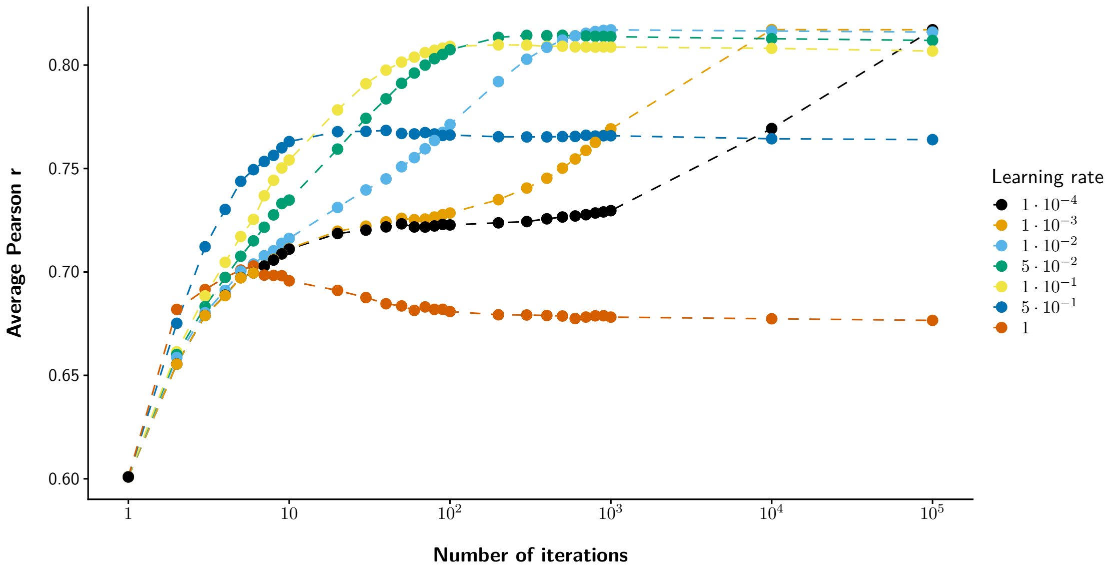


Figure 3.7: Learning curves for the general gradient boosted tree models. Pair of plots showing the average Spearman correlation coefficient between true and predicted fitness scores on the validation set for the gradient boosted tree models trained in a LOPO fashion as a function of the number of iterations used in training. Each colour corresponds to a different learning rate. The dotted lines connect performance points obtained with the same learning rate, defining a family of learning curves. The averaging of the correlation coefficient is across the nine datasets used in this work. The other hyperparameters besides the learning rate and the number of iterations were kept constant at the optimal values determined during the random search.

the dataset `beta-lactamase`. The trRosetta-derived features seem very important for the LOPO models on the dataset `gb1`, and in general, they are relatively important in most datasets. They are less important for single protein models, but still, they contribute strongly to performances in the dataset `gb1`. The NetSurfP-2 solvent accessibility is important for the LOPO models on dataset `PSD95pdz3`, and moderately important for other models and on other datasets. The remaining feature groups seem to be less critical.

For the linear regressor (Figure 3.10), the importance of features is more equilibrated, with a less strong dependency on the EVcouplings feature group. Besides EVcouplings, the NetSurfP-2 predicted secondary structures and the PSSM scores are the most important groups. Follow the trRosetta features and the NetSurfP-2 solvent accessibility. The feature dependence of the linear model seems to be less variable across datasets.

(a)



(b)

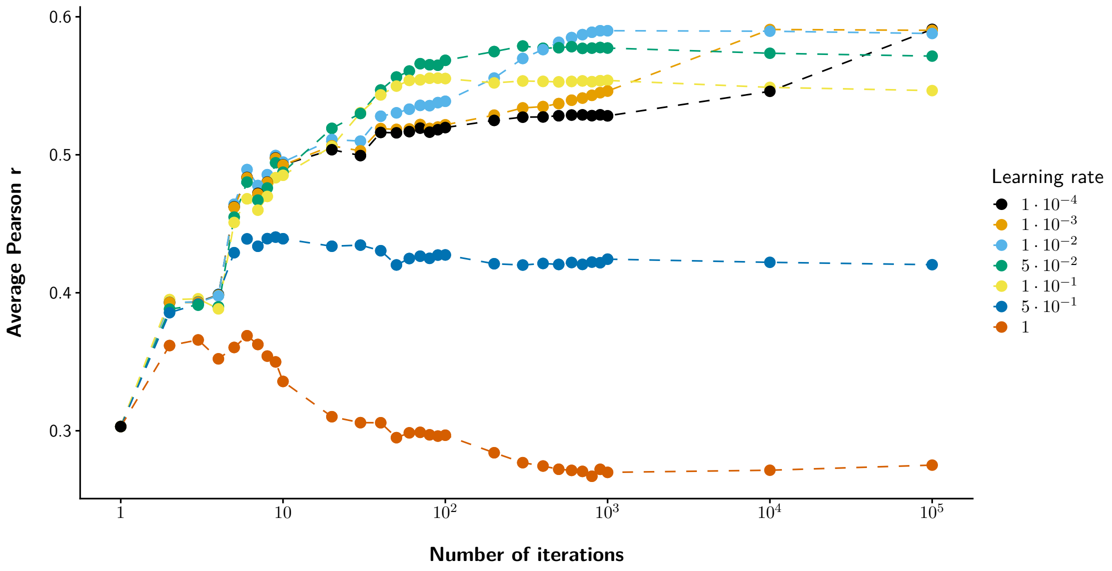


Figure 3.8: Learning curves for the single protein models. Pair of plots showing the average Pearson correlation coefficient between true and predicted fitness scores on the validation set for the single protein models as a function of the number of iterations used in training. Each colour corresponds to a different learning rate. The other hyperparameters besides the learning rate and the number of iterations were kept constant at the optimal values determined during the random search. **(a)** Learning curves for models trained under the "naive" paradigm. **(b)** Learning curves for models trained trained by segregating protein positions.

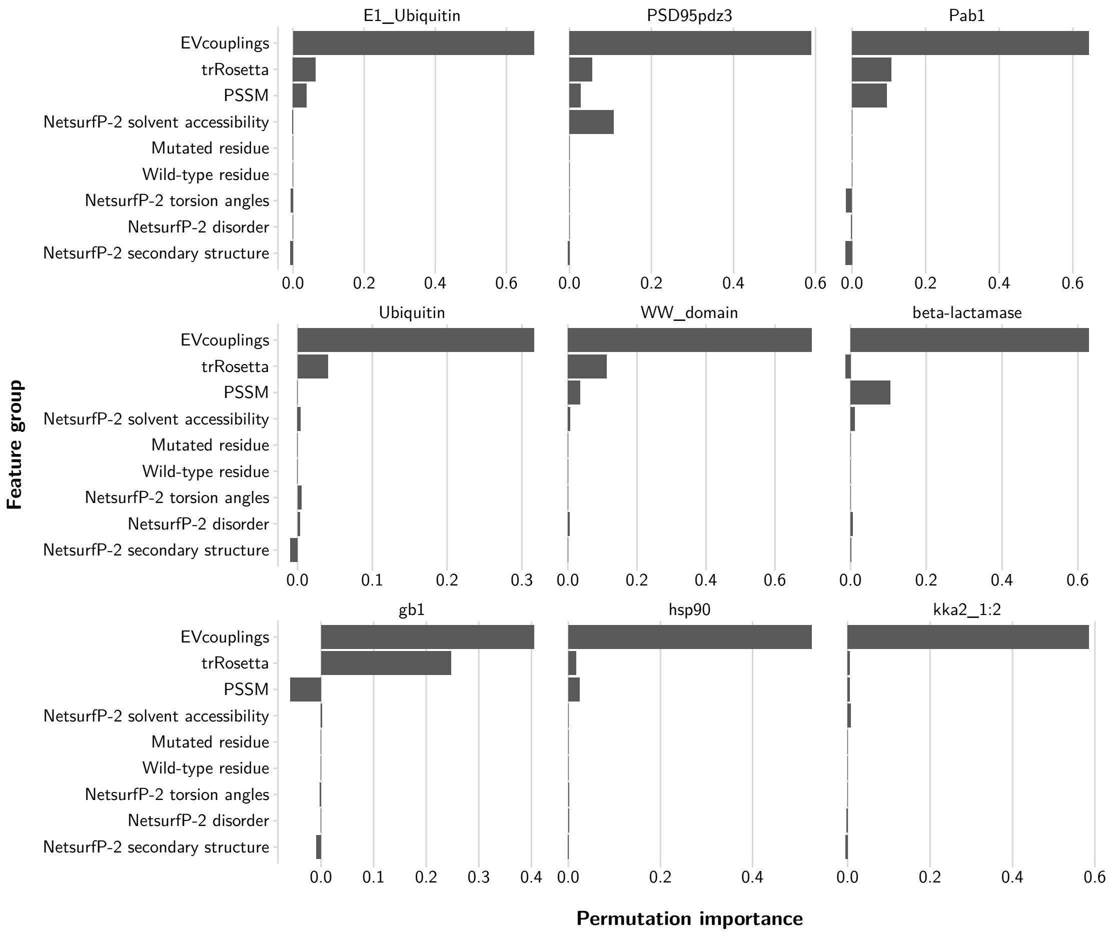


Figure 3.9: Feature importance for the general gradient boosted trees models. Series of bar plots showing the grouped permutation importance of the features used in the general models trained in a LOPO fashion, subdivided by dataset. These plots refer to the importances in the gradient boosted tree models.

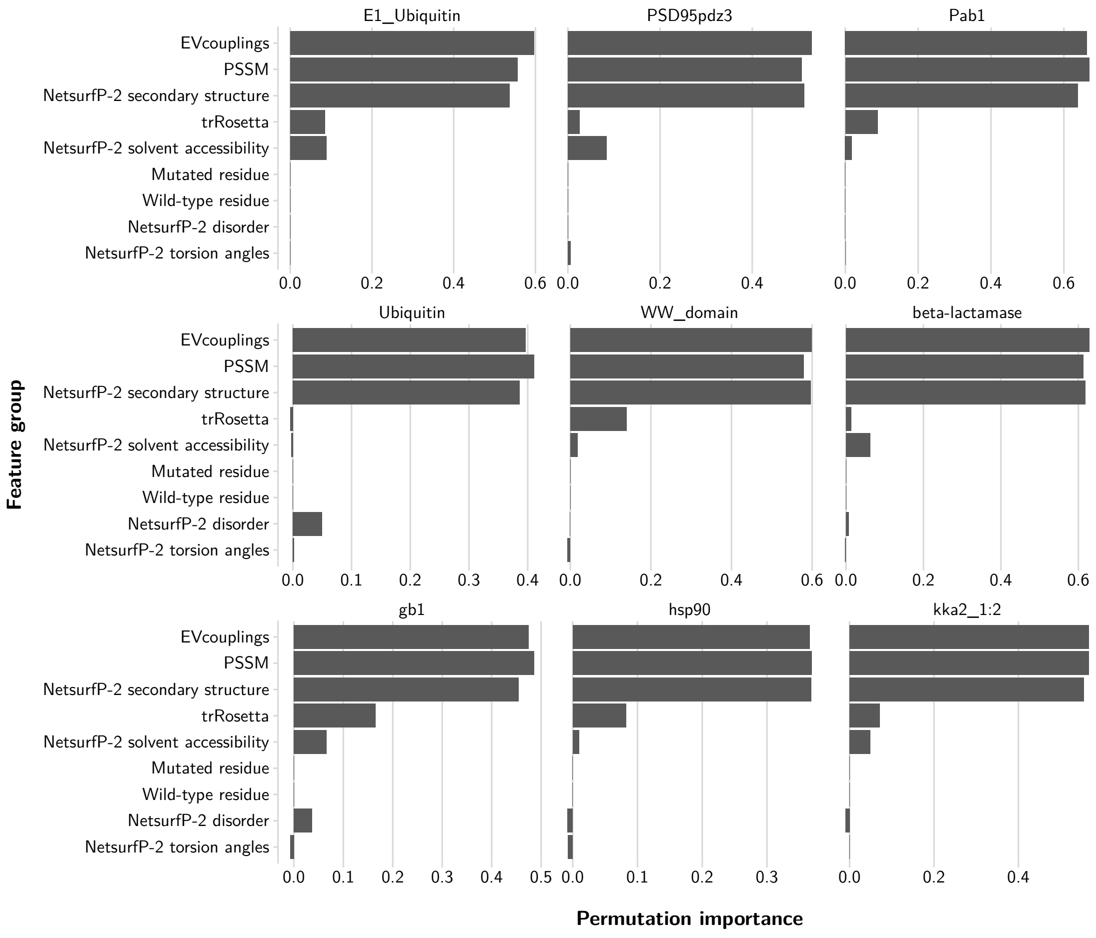


Figure 3.10: Learning curves for the general linear regression models. Series of bar plots showing the grouped permutation importance of the features used in the general models trained in a LOPO fashion, subdivided by dataset. These plots refer to the importances in the linear regression models.

Chapter 4

Discussion

As evidenced by the comparison of my models to Envision (Gray et al., 2018), it is possible to obtain relatively good performances on quantitative variant effect prediction also by replacing experimental structural features with sequence-based predictions. The use of structure-derived features, particularly solvent accessibility (Savojardo et al., 2021) but also, as shown in Section 3.7, residue contacts, is of paramount importance in the prediction of the effect of mutations. The observation that it is possible to obtain good results using predicted structural features is interesting since for many protein families an experimental structure is not yet available. My method builds upon the substantial improvements in the protein structure prediction field that happened in the last few years. Notable protein structure predictors that obtained impressive results in the last two edition of the Critical Assessment of Structure Prediction (Abriata et al., 2019) are trRosetta (Yang et al., 2020, CASP13), AlphaFold (Senior et al., 2020, CASP13), and AlphaFold2 (Jumper et al., 2020, CASP14).

A surprising observation from my results is the relatively small importance of the identity of the mutant residue in predicting the effect of a mutation. This indicates that the position along the protein of a given mutation and its local context, more than which residue is introduced, are strong determinants of the effect of mutations. Nonetheless, the limited importance of the mutated residue could be explained by collinearity with other features. As described in Section 2.11 collinearity can lead to underestimation of feature importance. The same reasoning applies also to the identity of the wild-type residue, which appeared to be non-critical for predictions. In general, structural properties and information on mutational patterns obtained from multiple sequence alignments seem to be superior features for predicting the effect of mutations than the identity of the mutation itself.

The results of my work suggest also that simple models, such as linear regression, can reach comparable performances to those obtained by more complex architectures such as gradient boosted trees. What is more, unsupervised models perform comparatively very well, rivalling and in some instances even exceeding the results of supervised models. However, it is important to notice that some of those unsupervised models such as DeepSequence (Riesselman et al., 2018) do so at the cost of training protein-specific deep neural network models. This is a notable computational requirement that can make the predictor inaccessible for casual users who lack computational expertise and available resources.

Another main conclusion from this work is that how a mutation effect predictor is tested is crucial when comparing the performance of different models. A naive approach to validation and testing, which involves simply splitting a deep mutational scanning dataset randomly, can lead to serious overestimation of the performances. Nonetheless, such an approach has been used in recent work, for example in Gelman et al. (2020).

It is noteworthy how just subdividing the training and test sets according to the mutated position in a protein-specific model leads to similar performances to training a general model on completely different proteins. This suggests that knowledge of the mutational pattern at different positions of the same protein is not very informative for the susceptibility to mutations of other protein positions. This fact could be considered in the design of new predictors.

A central point in the use of deep mutational scanning data for variant effect prediction is how to make the experimental scores from different datasets comparable. Different experiments target proteins with very different susceptibility to mutation, in different biological organisms, and using vastly different selection techniques. It is not trivial to devise a way to disentangle the experimental range of a particular experiment from the mutational sensitivity of the protein. For example, a wide score distribution in a particular experiment could be due to the methodology adopted, but also to intrinsic sensitivity to mutations of some residues in the protein. The use of an intra-dataset ranking loss instead of a regression loss, or the quantile normalization of the fitness scores could be possible solutions. These approaches, however, inevitably lead to a loss of information and predictive power. It would be interesting to assess systematically which normalization strategy works best for deep mutational scanning data, and how information can be extracted from the vast variability intrinsic in this methodology. Said this, the finding of this work that single protein models trained by segregating protein positions perform at a similar level to a general model trained in a LOPO fashion suggest that this issue of normalization is less influential than I expected.

The fact that unsupervised models perform well compared to supervised ones, and their independence from deep mutational scanning data altogether, makes them an interesting research direction in the field. In addition, the use of unsupervised models would completely bypass the issue of how to normalize deep mutational scanning data.

Coming back to supervised models, the relative scarcity of deep mutational scanning datasets is probably a big factor in the limited performances that can be obtained with quantitative effect predictors. When more deep mutational scanning studies will be available, it will be possible to use increasingly complex models and extract more subtle signals of mutation effect. This will lead to better predictions on one hand, and a better understanding of the biology underpinning the effect of mutations on the other.

On a more practical note, the set of features used in this work, albeit easy to obtain and not dependent on structural data, is probably sub-optimal. Many of the features used showed to have very limited importance. It would be interesting to find new features that better capture the factors determining the effect of mutations. I believe that the use of predicted structural features instead of experimental ones can expand the applicability of mutation effect predictors, and could be explored further.

As said above simple models such as linear regression performed at the same level as more complex ones, but this is not to say that further exploration of model architectures is not advisable. The poor performances obtained with complex models could be also due to a lack of data. I suggest that the exploration of new models and architectures in the field is still of great interest. In particular, the use of a deep learning model could be attempted. The availability of contacts graph predictions from trRosetta would make it possible to implement a graph-convolutional network, similar to what was done, albeit for a different purpose, in Baldassarre et al. (2020). Also, the implementation of a sliding window on the sequence context of a mutation could be done, for example using a recurrent neural network architecture. Once more deep mutational scanning data will be available, it could be also attempted an approach that dispenses from crafted features altogether and just relies on the raw multiple sequence alignments, similarly to what was done by Mirabello et al. (2019). Finally, the fact that all of the feature used in this work are obtainable from multiple sequence alignments suggests that it could be possible, instead of using other models to predict structural features, to create a single, end-to-end differentiable model from the multiple sequence alignments to the fitness scores. To achieve this, a self-supervised approach could be suitable.

A final issue that must be addressed, even though outside of the scope of this project, is how the deep mutational scanning fitness scores relate to pathogenicity. This is very dependent on the experimental selection adopted and on how critical the assessed protein is. When interpreting the biological meaning of mutation effect predictions obtained with a model trained on something other than pathogenicity, I would caution against the direct connection between fitness scores, which are often a proxy for thermodynamic stability, and the pathogenicity of a genotypic variation.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). Tensorflow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/

Abriata, L. A., Tamò, G. E. & Peraro, M. D. (2019). A further leap of improvement in tertiary structure prediction in CASP13 prompts new routes for future assessments. *Proteins: Structure, Function, and Bioinformatics*, *87*(12), 1100–1112. https://doi.org/10.1002/prot.25787

Adzhubei, I. A., Schmidt, S., Peshkin, L., Ramensky, V. E., Gerasimova, A., Bork, P., Kondrashov, A. S. & Sunyaev, S. R. (2010). A method and server for predicting damaging missense mutations. *Nature Methods*, *7*(4), 248–249. https://doi.org/10.1038/nmeth0410-248

Agostinelli, C. & Lund, U. (2017). *R package circular: Circular statistics*. CA: Department of Environmental Sciences, Informatics, Statistics, Ca' Foscari University, Venice, Italy. UL: Department of Statistics, California Polytechnic State University, San Luis Obispo, California, USA. https://r-forge.r-project.org/projects/circular/

Ahmad, S., Gromiha, M. M. & Sarai, A. (2003). Real value prediction of solvent accessibility from amino acid sequence. *Proteins: Structure, Function, and Bioinformatics*, *50*(4), 629–635. https://doi.org/10.1002/prot.10328

Baldassarre, F., Hurtado, D. M., Elofsson, A. & Azizpour, H. (2020). GraphQA: Protein model quality assessment using graph convolutional networks (Y. Ponty, Ed.). *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btaa714

Bateman, A., Martin, M.-J., Orchard, S., Magrane, M., Agivetova, R., Ahmad, S., Alpi, E., Bowler-Barnett, E. H., Britto, R., Bursteinas, B., A-Jee, H. B., Coetzee, R., Cukura, A., Silva, A. D., Denny, P., Dogan, T., Ebenezer, T., Fan, J., Castro, L. G., … Teodoro, D. (2020). UniProt: The universal protein knowledgebase in 2021. *Nucleic Acids Research*, *49*(D1), D480–D489. https://doi.org/10.1093/nar/gkaa1100

Bennett, S. (2004). Solexa ltd. *Pharmacogenomics*, *5*(4), 433–438. https://doi.org/10.1517/14622416.5.4.433

Bergstra, J. & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(10), 281–305. http://jmlr.org/papers/v13/bergstra12a.html

Bhagavatula, G., Rich, M. S., Young, D. L., Marin, M. & Fields, S. (2017). A massively parallel fluorescence assay to characterize the effects of synonymous mutations on TP53 expression. *Molecular Cancer Research*, *15*(10), 1301–1307. https://doi.org/10.1158/1541-7786.mcr-17-0245

Blum, M., Chang, H.-Y., Chuguransky, S., Grego, T., Kandasaamy, S., Mitchell, A., Nuka, G., Paysan-Lafosse, T., Qureshi, M., Raj, S., Richardson, L., Salazar, G. A., Williams, L., Bork, P., Bridge, A., Gough, J., Haft, D. H., Letunic, I., Marchler-Bauer, A., … Finn, R. D. (2020). The InterPro protein families and domains database: 20 years on. *Nucleic Acids Research*, *49*(D1), D344–D354. https://doi.org/10.1093/nar/gkaa977

Boldi, P. & Vigna, S. (2014). Axioms for centrality. *Internet Mathematics*, *10*(3-4), 222–262. https://doi.org/10.1080/15427951.2013.865686

Brandes, U. (2001). A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, *25*(2), 163–177. https://doi.org/10.1080/0022250x.2001.9990249

Burges, C. J. C. (2010). From RankNet to LambdaRank to LambdaMART: An overview. *Microsoft Research*. https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/?from=http%3A%2F%2Fresearch.microsoft.com%2Fpubs%2F132652%2Fmsr-tr-2010-82.pdf

Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., Costanzo, L. D., Christie, C., Duarte, J. M., Dutta, S., Feng, Z., Ghosh, S., Goodsell, D. S., Green, R. K., Guranovic, V., Guzenko, D., Hudson, B. P., Liang, Y., Lowe, R., Peisach, E., … Ioannidis, Y. E. (2018). Protein Data Bank: The single global archive for 3D macromolecular structure data. *Nucleic Acids Research*, *47*(D1), D520–D528. https://doi.org/10.1093/nar/gky949

Canty, A. & Ripley, B. D. (2021). *boot: Bootstrap R (S-Plus) functions*.

Chen, T. & Guestrin, C. (2016). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939785

Chicco, D. (2017). Ten quick tips for machine learning in computational biology. *BioData Mining*, *10*(1). https://doi.org/10.1186/s13040-017-0155-3

Chiti, F. (2000). Mutational analysis of the propensity for amyloid formation by a globular protein. *The EMBO Journal*, *19*(7), 1441–1449. https://doi.org/10.1093/emboj/19.7.1441

Chollet, F. et al. (2015). Keras.

Claesen, M. & Moor, B. D. (2015). Hyperparameter search in machine learning.

Collins, F. S., Brooks, L. D. & Chakravarti, A. (1998). A DNA polymorphism discovery resource for research on human genetic variation. *Genome Research*, *8*(12), 1229–1231. https://doi.org/10.1101/gr.8.12.1229

Dalcin, L. D., Paz, R. R., Kler, P. A. & Cosimo, A. (2011). Parallel distributed computing using Python. *Advances in Water Resources*, *34*(9), 1124–1139. https://doi.org/10.1016/j.advwatres.2011.04.013

Dana, J. M., Gutmanas, A., Tyagi, N., Qi, G., O'Donovan, C., Martin, M. & Velankar, S. (2018). SIFTS: Updated structure integration with function, taxonomy and sequences resource allows 40-fold increase in coverage of structure-based annotations for proteins. *Nucleic Acids Research*, *47*(D1), D482–D489. https://doi.org/10.1093/nar/gky1114

Daopin, S., Alber, T., Baase, W. A., Wozniak, J. A. & Matthews, B. W. (1991). Structural and thermodynamic analysis of the packing of two α-helices in bacteriophage T4 lysozyme. *Journal of Molecular Biology*, *221*(2), 647–667. https://doi.org/10.1016/0022-2836(91)80079-a

Davies, J. (1988). Genetic engineering: Processes and products. *Trends in Biotechnology*, *6*(4), S7–S11. https://doi.org/10.1016/0167-7799(88)90005-4

Davison, A. C. (1999). *Bootstrap methods and their application*. Cambridge University Press.

DePristo, M. A., Weinreich, D. M. & Hartl, D. L. (2005). Missense meanderings in sequence space: A biophysical view of protein evolution. *Nature Reviews Genetics*, *6*(9), 678–687. https://doi.org/10.1038/nrg1672

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, *56*(293), 52–64. https://doi.org/10.1080/01621459.1961.10482090

Eddy, S. R. (2011). Accelerated profile HMM searches (W. R. Pearson, Ed.). *PLoS Computational Biology*, *7*(10), e1002195. https://doi.org/10.1371/journal.pcbi.1002195

Eid, J., Fehr, A., Gray, J., Luong, K., Lyle, J., Otto, G., Peluso, P., Rank, D., Baybayan, P., Bettman, B., Bibillo, A., Bjornson, K., Chaudhuri, B., Christians, F., Cicero, R., Clark, S., Dalal, R., deWinter, A., Dixon, J., … Turner, S. (2009). Real-time DNA sequencing from single polymerase molecules. *Science*, *323*(5910), 133–138. https://doi.org/10.1126/science.1162986

Firnberg, E., Labonte, J. W., Gray, J. J. & Ostermeier, M. (2014). A comprehensive, high-resolution map of a gene's fitness landscape. *Molecular Biology and Evolution*, *31*(6), 1581–1592. https://doi.org/10.1093/molbev/msu081

Fowler, D. M., Araya, C. L., Fleishman, S. J., Kellogg, E. H., Stephany, J. J., Baker, D. & Fields, S. (2010). High-resolution mapping of protein sequence-function relationships. *Nature Methods*, *7*(9), 741–746. https://doi.org/10.1038/nmeth.1492

Fowler, D. M. & Fields, S. (2014). Deep mutational scanning: A new style of protein science. *Nature Methods*, *11*(8), 801–807. https://doi.org/10.1038/nmeth.3027

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social Networks*, *1*(3), 215–239. https://doi.org/10.1016/0378-8733(78)90021-7

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Friedman, J., Hastie, T. & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, *28*(2). https://doi.org/10.1214/aos/1016218223

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5). https://doi.org/10.1214/aos/1013203451

Gelman, S., Romero, P. A. & Gitter, A. (2020). Neural networks to learn protein sequence-function relationships from deep mutational scanning data. https://doi.org/10.1101/2020.10.25.353946

Goetz, L. H. & Schork, N. J. (2018). Personalized medicine: Motivation, challenges, and progress. *Fertility and Sterility*, *109*(6), 952–963. https://doi.org/10.1016/j.fertnstert.2018.05.006

Gray, V. E., Hause, R. J., Luebeck, J., Shendure, J. & Fowler, D. M. (2018). Quantitative missense variant effect prediction using large-scale mutagenesis data. *Cell Systems*, *6*(1), 116–124.e3. https://doi.org/10.1016/j.cels.2017.11.003

Green, S. M. & Shortle, D. (1993). Patterns of nonadditivity between pairs of stability mutations in staphylococcal nuclease. *Biochemistry*, *32*(38), 10131–10139. https://doi.org/10.1021/bi00089a032

Hagberg, A. A., Schult, D. A. & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught & J. Millman (Eds.), *Proceedings of the 7th Python in science conference* (pp. 11–15).

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hecht, M., Bromberg, Y. & Rost, B. (2015). Better prediction of functional effects for sequence variants. *BMC Genomics*, *16*(S8). https://doi.org/10.1186/1471-2164-16-s8-s1

Henikoff, S. & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, *89*(22), 10915–10919. https://doi.org/10.1073/pnas.89.22.10915

Henry, L. & Wickham, H. (2021). *rlang: Functions for base types and core R and Tidyverse features*. https://CRAN.R-project.org/package=rlang

Hiatt, J. B., Patwardhan, R. P., Turner, E. H., Lee, C. & Shendure, J. (2010). Parallel, tag-directed assembly of locally derived short sequence reads. *Nature Methods*, *7*(2), 119–122. https://doi.org/10.1038/nmeth.1416

Hollander, M., Chicken, E. & Wolfe, D. A. (2013, November 15). *Nonparametric statistical methods*. John Wiley & Sons.

Holme, I. B., Gregersen, P. L. & Brinch-Pedersen, H. (2019). Induced genetic variation in crop plants by random or targeted mutagenesis: Convergence and differences. *Frontiers in Plant Science*, *10*. https://doi.org/10.3389/fpls.2019.01468

Hopf, T., Ingraham, J., Poelwijk, F., Schärfe, C., Springer, M., Sander, C. & Marks, D. (2017). Mutation effects predicted from sequence co-variation. *Nature Biotechnology*, *35*(2), 128–135. https://doi.org/10.1038/nbt.3769

Hopf, T. A., Green, A. G., Schubert, B., Mersmann, S., Schärfe, C. P. I., Ingraham, J. B., Toth-Petroczy, A., Brock, K., Riesselman, A. J., Palmedo, P., Kang, C., Sheridan, R., Draizen, E. J., Dallago, C., Sander, C. & Marks, D. S. (2018). The EVcouplings Python framework for coevolutionary sequence analysis (A. Valencia, Ed.). *Bioinformatics*, *35*(9), 1582–1584. https://doi.org/10.1093/bioinformatics/bty862

Hunt, R. C., Simhadri, V. L., Iandoli, M., Sauna, Z. E. & Kimchi-Sarfaty, C. (2014). Exposing synonymous mutations. *Trends in Genetics*, *30*(7), 308–321. https://doi.org/10.1016/j.tig.2014.04.006

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/mcse.2007.55

Hutchison, C. A., Phillips, S., Edgell, M. H., Gillam, S., Jahnke, P. & Smith, M. (1978). Mutagenesis at a specific position in a DNA sequence. *Journal of Biological Chemistry*, *253*(18), 6551–6560. https://doi.org/10.1016/s0021-9258(19)46967-6

Ito, N., Phillips, S. E. V., Stevens, C., Ogel, Z. B., McPherson, M. J., Keen, J. N., Yadav, K. D. S. & Knowles, P. F. (1991). Novel thioether bond revealed by a 1.7 Å crystal structure of galactose oxidase. *Nature*, *350*(6313), 87–90. https://doi.org/10.1038/350087a0

Jain, P. C. & Varadarajan, R. (2014). A rapid, efficient, and economical inverse polymerase chain reaction-based method for generating a site saturation mutant library. *Analytical Biochemistry*, *449*, 90–98. https://doi.org/10.1016/j.ab.2013.12.002

Jammalamadaka, S. (2001). *Topics in circular statistics*. World Scientific.

Jumper, J., Evans, R., Hassabis, A., Tim, P., Michael, G., Kathryn, F., Olaf, T., Russ, R., Augustin, B., Alex, Ž., Clemens, B., Simon, M., Kohl, A. A., Potapenko, A., Ballard, A. J., Bernardino, A. C., Romera-paredes, Nikolov, S., Jain, R., … Demis, P. K. (2020). High accuracy protein structure prediction using deep learning. *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*. https://predictioncenter.org/casp14/doc/CASP14_Abstracts.pdf

Kabsch, W. & Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, *22*(12), 2577–2637. https://doi.org/10.1002/bip.360221211

Kalds, P., Zhou, S., Cai, B., Liu, J., Wang, Y., Petersen, B., Sonstegard, T., Wang, X. & Chen, Y. (2019). Sheep and goat genome engineering: From random transgenesis to the CRISPR era. *Frontiers in Genetics*, *10*. https://doi.org/10.3389/fgene.2019.00750

Kalinowska, B., Banach, M., Wiśniowski, Z., Konieczny, L. & Roterman, I. (2017). Is the hydrophobic core a universal structural element in proteins? *Journal of Molecular Modeling*, *23*(7). https://doi.org/10.1007/s00894-017-3367-z

Katsonis, P. & Lichtarge, O. (2014). A formal perturbation equation between genotype and phenotype determines the evolutionary action of protein-coding variations on fitness. *Genome Research*, *24*(12), 2050–2058. https://doi.org/10.1101/gr.176214.114

Kircher, M., Witten, D. M., Jain, P., O'Roak, B. J., Cooper, G. M. & Shendure, J. (2014). A general framework for estimating the relative pathogenicity of human genetic variants. *Nature Genetics*, *46*(3), 310–315. https://doi.org/10.1038/ng.2892

Klausen, M. S., Jespersen, M. C., Nielsen, H., Jensen, K. K., Jurtz, V. I., Sønderby, C. K., Sommer, M. O. A., Winther, O., Nielsen, M., Petersen, B. & Marcatili, P. (2019). NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning. *Proteins: Structure, Function, and Bioinformatics*, *87*(6), 520–527. https://doi.org/10.1002/prot.25674

Kluyver, T., Ragan-Kelley, B., Prez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C. & Team, J. D. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. *Positioning and power in academic publishing: Players, agents and agendas* (pp. 87–90). IOS Press. https://doi.org/10.3233/978-1-61499-649-1-87

Knight, R. (2003). Analyzing partially randomized nucleic acid pools: Straight dope on doping. *Nucleic Acids Research*, *31*(6), 30e–30. https://doi.org/10.1093/nar/gng030

Kuhn, M. (2021). *Caret: Classification and regression training* [R package version 6.0-88].

Liu, L. & Kumar, S. (2013). Evolutionary balancing is critical for correctly forecasting disease-associated amino acid variants. *Molecular Biology and Evolution*, *30*(6), 1252–1257. https://doi.org/10.1093/molbev/mst037

Mardia, K. V. & Jupp, P. E. (1999, January 3). *Directional statistics*. John Wiley & Sons.

Matreyek, K. A., Stephany, J. J. & Fowler, D. M. (2017). A platform for functional assessment of large variant libraries in mammalian cells. *Nucleic Acids Research*, *45*(11), e102–e102. https://doi.org/10.1093/nar/gkx183

McLaughlin, R. N., Poelwijk, F. J., Raman, A., Gosal, W. S. & Ranganathan, R. (2012). The spatial architecture of protein function and adaptation. *Nature*, *491*(7422), 138–142. https://doi.org/10.1038/nature11500

Melamed, D., Young, D. L., Gamble, C. E., Miller, C. R. & Fields, S. (2013). Deep mutational scanning of an RRM domain of the saccharomyces cerevisiae poly(A)-binding protein. *RNA*, *19*(11), 1537–1551. https://doi.org/10.1261/rna.040709.113

Melnikov, A., Rogov, P., Wang, L., Gnirke, A. & Mikkelsen, T. S. (2014). Comprehensive mutational scanning of a kinase in vivo reveals substrate-dependent fitness landscapes. *Nucleic Acids Research*, *42*(14), e112–e112. https://doi.org/10.1093/nar/gku511

Mirabello, C. & Wallner, B. (2019). rawMSA: End-to-end deep learning using raw multiple sequence alignments (Y. Zhang, Ed.). *PLOS ONE*, *14*(8), e0220182. https://doi.org/10.1371/journal.pone.0220182

Mirdita, M., von den Driesch, L., Galiez, C., Martin, M. J., Söding, J. & Steinegger, M. (2016). Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Research*, *45*(D1), D170–D176. https://doi.org/10.1093/nar/gkw1081

Mishra, P., Flynn, J. M., Starr, T. N. & Bolon, D. N. A. (2016). Systematic mutant analyses elucidate general and client-specific aspects of Hsp90 function. *Cell Reports*, *15*(3), 588–598. https://doi.org/10.1016/j.celrep.2016.03.046

Mitraki, A., Fane, B., Haase-Pettingell, C., Sturtevant, J. & King, J. (1991). Global suppression of protein folding defects and inclusion body formation. *Science*, *253*(5015), 54–58. https://doi.org/10.1126/science.1648264

Morrison, K. L. & Weiss, G. A. (2001). Combinatorial alanine-scanning. *Current Opinion in Chemical Biology*, *5*(3), 302–307. https://doi.org/10.1016/s1367-5931(00)00206-4

Muller, H. J. (1927). Artificial transmutation of the gene. *Science*, *66*(1699), 84–87. https://doi.org/10.1126/science.66.1699.84

Natekin, A. & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, *7*. https://doi.org/10.3389/fnbot.2013.00021

Newman, M. E. J. (2001). Scientific collaboration networks. II. shortest paths, weighted networks, and centrality. *Physical Review E*, *64*(1), 016132. https://doi.org/10.1103/physreve.64.016132

Ng, P. C. & Henikoff, S. (2001). Predicting deleterious amino acid substitutions. *Genome Research*, *11*(5), 863–874. https://doi.org/10.1101/gr.176601

Nisthal, A., Wang, C. Y., Ary, M. L. & Mayo, S. L. (2019). Protein stability engineering insights revealed by domain-wide comprehensive mutagenesis. *Proceedings of the National Academy of Sciences*, *116*(33), 16367–16377. https://doi.org/10.1073/pnas.1903888116

Olson, C. A., Wu, N. C. & Sun, R. (2014). A comprehensive biophysical description of pairwise epistasis throughout an entire protein domain. *Current Biology*, *24*(22), 2643–2651. https://doi.org/10.1016/j.cub.2014.09.072

Onnela, J.-P., Saramäki, J., Kertész, J. & Kaski, K. (2005). Intensity and coherence of motifs in weighted complex networks. *Physical Review E*, *71*(6), 065103. https://doi.org/10.1103/physreve.71.065103

Pagès, H., Aboyoun, P., Gentleman, R. & DebRoy, S. (2020). *Biostrings: Efficient manipulation of biological strings.*

Pawar, A. P., DuBay, K. F., Zurdo, J., Chiti, F., Vendruscolo, M. & Dobson, C. M. (2005). Prediction of "aggregation-prone" and "aggregation-susceptible" regions in proteins associated with neurodegenerative diseases. *Journal of Molecular Biology*, *350*(2), 379–392. https://doi.org/10.1016/j.jmb.2005.04.016

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peracchi, A. (2001). Enzyme catalysis: Removing chemically 'essential' residues by site-directed mutagenesis. *Trends in Biochemical Sciences*, *26*(8), 497–503. https://doi.org/10.1016/s0968-0004(01)01911-9

Perez, F. & Granger, B. E. (2007). IPython: A system for interactive scientific computing. *Computing in Science & Engineering*, *9*(3), 21–29. https://doi.org/10.1109/mcse.2007.53

R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Ramachandran, G. N., Ramakrishnan, C. & Sasisekharan, V. (1963). Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, *7*(1), 95–99. https://doi.org/10.1016/s0022-2836(63)80023-6

Ramensky, V. (2002). Human non-synonymous SNPs: Server and survey. *Nucleic Acids Research*, *30*(17), 3894–3900. https://doi.org/10.1093/nar/gkf493

Ramirez-Alvarado, M., Merkel, J. S. & Regan, L. (2000). A systematic exploration of the influence of the protein stability on amyloid fibril formation in vitro. *Proceedings of the National Academy of Sciences*, *97*(16), 8979–8984. https://doi.org/10.1073/pnas.150091797

Reeb, J., Wirth, T. & Rost, B. (2020). Variant effect predictions capture some aspects of deep mutational scanning experiments. *BMC Bioinformatics*, *21*(1). https://doi.org/10.1186/s12859-020-3439-4

Rentzsch, P., Schubach, M., Shendure, J. & Kircher, M. (2021). CADD-Splice—-improving genome-wide variant effect prediction using deep learning-derived splice scores. *Genome Medicine*, *13*(1). https://doi.org/10.1186/s13073-021-00835-9

Rentzsch, P., Witten, D., Cooper, G. M., Shendure, J. & Kircher, M. (2018). CADD: Predicting the deleteriousness of variants throughout the human genome. *Nucleic Acids Research*, *47*(D1), D886–D894. https://doi.org/10.1093/nar/gky1016

Riesselman, A. J., Ingraham, J. B. & Marks, D. S. (2018). Deep generative models of genetic variation capture the effects of mutations. *Nature Methods*, *15*(10), 816–822. https://doi.org/10.1038/s41592-018-0138-4

Roscoe, B. P. & Bolon, D. N. A. (2014). Systematic exploration of ubiquitin sequence, E1 activation efficiency, and experimental fitness in yeast. *Journal of Molecular Biology*, *426*(15), 2854–2870. https://doi.org/10.1016/j.jmb.2014.05.019

Roscoe, B. P., Thayer, K. M., Zeldovich, K. B., Fushman, D. & Bolon, D. N. A. (2013). Analyses of the effects of all ubiquitin point mutants on yeast growth rate. *Journal of Molecular Biology*, *425*(8), 1363–1377. https://doi.org/10.1016/j.jmb.2013.01.032

RStudio Team. (2021). *Rstudio: Integrated development environment for r*. RStudio, PBC. Boston, MA. http://www.rstudio.com/

Rubin, A. F., Gelman, H., Lucas, N., Bajjalieh, S. M., Papenfuss, A. T., Speed, T. P. & Fowler, D. M. (2017). A statistical framework for analyzing deep mutational scanning data. *Genome Biology*, *18*(1). https://doi.org/10.1186/s13059-017-1272-5

Sarkisyan, K. S., Bolotin, D. A., Meer, M. V., Usmanova, D. R., Mishin, A. S., Sharonov, G. V., Ivankov, D. N., Bozhanova, N. G., Baranov, M. S., Soylemez, O., Bogatyreva, N. S., Vlasov, P. K., Egorov, E. S., Logacheva, M. D., Kondrashov, A. S., Chudakov, D. M., Putintseva, E. V., Mamedov, I. Z., Tawfik, D. S., … Kondrashov, F. A. (2016). Local fitness landscape of the green fluorescent protein. *Nature*, *533*(7603), 397–401. https://doi.org/10.1038/nature17995

Savojardo, C., Manfredi, M., Martelli, P. L. & Casadio, R. (2021). Solvent accessibility of residues undergoing pathogenic variations in humans: From protein structures to protein sequences. *Frontiers in Molecular Biosciences*, *7*. https://doi.org/10.3389/fmolb.2020.626363

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W. R., Bridgland, A., Penedones, H., Petersen, S., Simonyan, K., Crossan, S., Kohli, P., Jones, D. T., Silver, D., Kavukcuoglu, K. & Hassabis, D. (2020). Improved protein structure prediction using potentials from deep learning. *Nature*, *577*(7792), 706–710. https://doi.org/10.1038/s41586-019-1923-7

Sheather, S. J. & Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society. Series B (Methodological)*, *53*(3), 683–690. http://www.jstor.org/stable/2345597

Shortle, D., DiMaio, D. & Nathans, D. (1981). Directed mutagenesis. *Annual Review of Genetics*, *15*(1), 265–294. https://doi.org/10.1146/annurev.ge.15.120181.001405

Sim, N.-l., Kumar, P., Hu, J., Henikoff, S., Schneider, G. & Ng, P. C. (2012). SIFT web server: Predicting effects of amino acid substitutions on proteins. *Nucleic Acids Research*, *40*(W1), W452–W457. https://doi.org/10.1093/nar/gks539

Stadler, L. J. (1928). Mutations in barley induced by X-rays and radium. *Science*, *68*(1756), 186–187. https://doi.org/10.1126/science.68.1756.186

Starita, L. M., Young, D. L., Islam, M., Kitzman, J. O., Gullingsrud, J., Hause, R. J., Fowler, D. M., Parvin, J. D., Shendure, J. & Fields, S. (2015). Massively parallel functional analysis of BRCA1 RING domain variants. *Genetics*, *200*(2), 413–422. https://doi.org/10.1534/genetics.115.175802

Steinegger, M., Meier, M., Mirdita, M., Vöhringer, H., Haunsberger, S. J. & Söding, J. (2019). HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinformatics*, *20*(1). https://doi.org/10.1186/s12859-019-3019-7

Suzek, B. E., Wang, Y., Huang, H., McGarvey, P. B. & Wu, C. H. (2014). UniRef clusters: A comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, *31*(6), 926–932. https://doi.org/10.1093/bioinformatics/btu739

The joblib development team. (2020). *Joblib: Running Python functions as pipeline jobs.* `https://joblib.readthedocs.io/`

The pandas development team. (2021). Pandas-dev/pandas: Pandas 1.2.3. `https://doi.org/10.5281/ZENODO.4572994`

Tsagris, M., Athineou, G., Sajib, A., Amson, E. & Waldstein, M. J. (2021). *Directional: A collection of R functions for directional data analysis.* `https://CRAN.R-project.org/package=Directional`

van Rossum, G. (2010). *The Python language reference.* Python Software Foundation Books.

Varadi, M., Berrisford, J., Deshpande, M., Nair, S. S., Gutmanas, A., Armstrong, D., Pravda, L., Al-Lazikani, B., Anyango, S., Barton, G. J., Berka, K., Blundell, T., Borkakoti, N., Dana, J., Das, S., Dey, S., Micco, P. D., Fraternali, F., Gibson, T., … Velankar, S. (2019). PDBe-KB: A community-driven resource for structural and functional annotations. *Nucleic Acids Research*, *48*(D1), D344–D353. `https://doi.org/10.1093/nar/gkz853`

Vaser, R., Adusumalli, S., Leng, S. N., Sikic, M. & Ng, P. C. (2015). SIFT missense predictions for genomes. *Nature Protocols*, *11*(1), 1–9. `https://doi.org/10.1038/nprot.2015.123`

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … van Mulbregt, P. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. `https://doi.org/10.1038/s41592-019-0686-2`

Waskom, M. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. `https://doi.org/10.21105/joss.03021`

Wensien, M., von Pappenheim, F. R., Funk, L.-M., Kloskowski, P., Curth, U., Diederichsen, U., Uranga, J., Ye, J., Fang, P., Pan, K.-T., Urlaub, H., Mata, R. A., Sautner, V. & Tittmann, K. (2021). A lysine–cysteine redox switch with an NOS bridge regulates enzyme function. *Nature*, *593*(7859), 460–464. `https://doi.org/10.1038/s41586-021-03513-3`

Wickham, H. (2007). Reshaping data with the reshape package. *Journal of Statistical Software*, *21*(12), 1–20. `http://www.jstatsoft.org/v21/i12/`

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis.* Springer-Verlag New York. `https://ggplot2.tidyverse.org`

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. `https://doi.org/10.21105/joss.01686`

Wickham, H. & Seidel, D. (2020). *scales: Scale functions for visualization.* `https://CRAN.R-project.org/package=scales`

Wilke, C. O. (2020). *cowplot: Streamlined plot theme and plot annotations for 'ggplot2'*. https://CRAN.R-project.org/package=cowplot

Yan, X. & Su, X. G. (2009, June). *Linear regression analysis*. World Scientific. https://doi.org/10.1142/6986

Yang, J., Anishchenko, I., Park, H., Peng, Z., Ovchinnikov, S. & Baker, D. (2020). Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences*, *117*(3), 1496–1503. https://doi.org/10.1073/pnas.1914677117

Supplementary Material

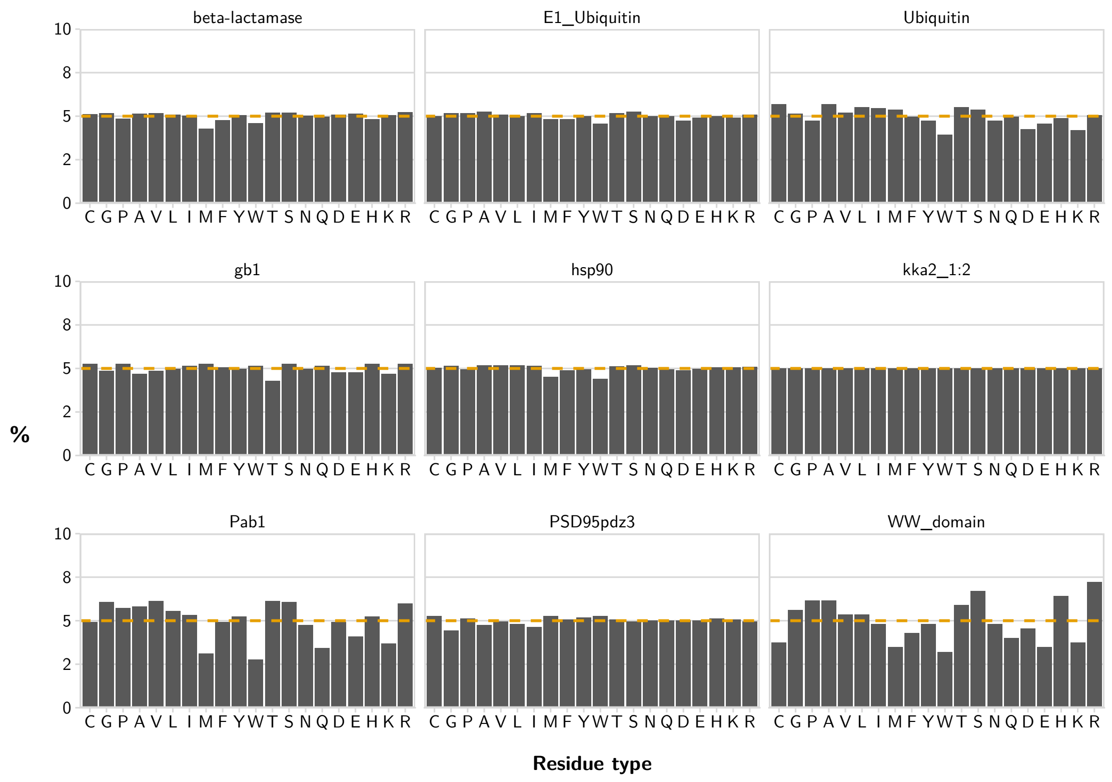


Figure S.1: Fraction of mutations towards a given residue, by dataset. Series of bar plots that represent the fraction of mutations towards each of the 20 standard residues, grouped by dataset. The orange dashed lines mark the expected fraction of mutations in the case of uniform frequency. For most datasets the distribution is uniform and no mutation is privileged. In the dataset `kka2_1:2` mutations towards isoleucine, tyrosine, serine, asparagine, and glutamate are overrepresented.

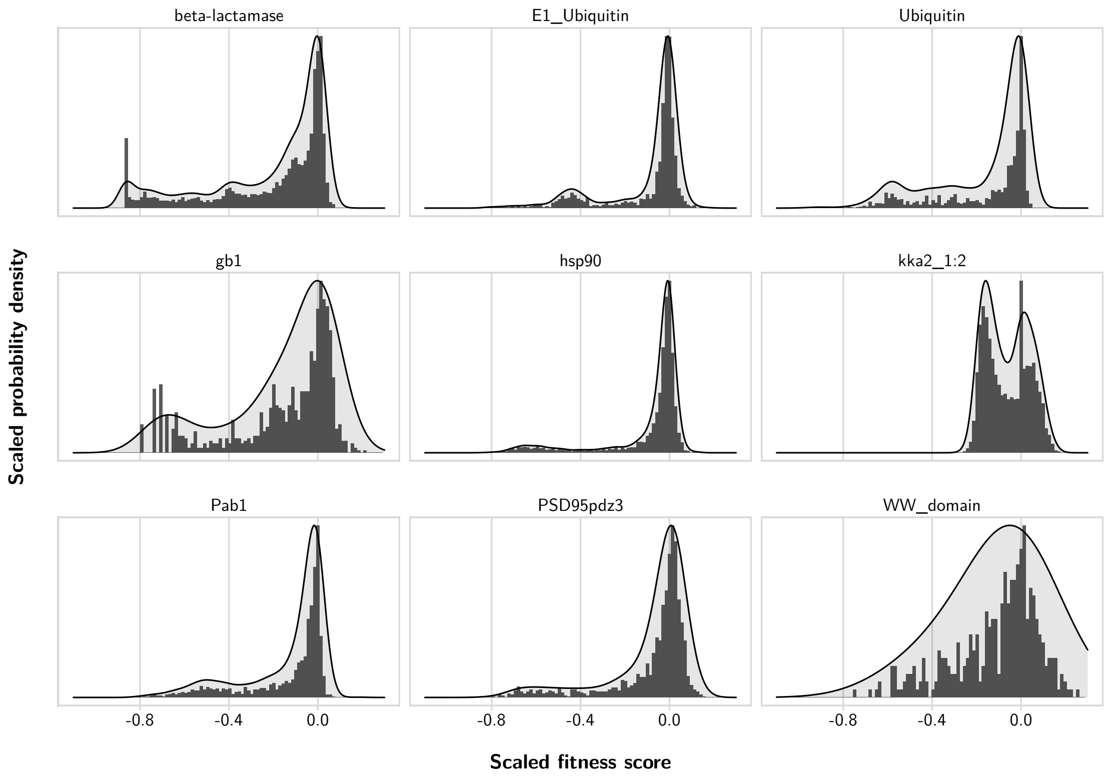


Figure S.2: Distribution of fitness scores, by dataset. Series of overlapping kernel density estimates (Gaussian kernel, bandwidth according to Sheather et al. (1991) adjusted with a multiplicative factor of 5, showing 512 estimation points) and histograms (100 bins) showing the distribution of the fitness scores for the datasets used in this work. To facilitate comparisons, the fitness scores were scaled independently for each dataset to the range 0 to 1 and centred such that a neutral mutation would have a scaled score of 0. The probability density was normalized to have a maximum value of 1 (and not a total area of 1). The vertical axis is expressed in terms of normalized probability density also for the histogram. All of the datasets seem to have a more or less pronounced bimodal distribution. A strong skew towards negative fitness scores is present. Neutral effects are by far the most frequent, followed by smaller density peaks in the detrimental effect region. The dataset `beta-lactamase` shows a possible sensitivity artifact at the extreme negative end.

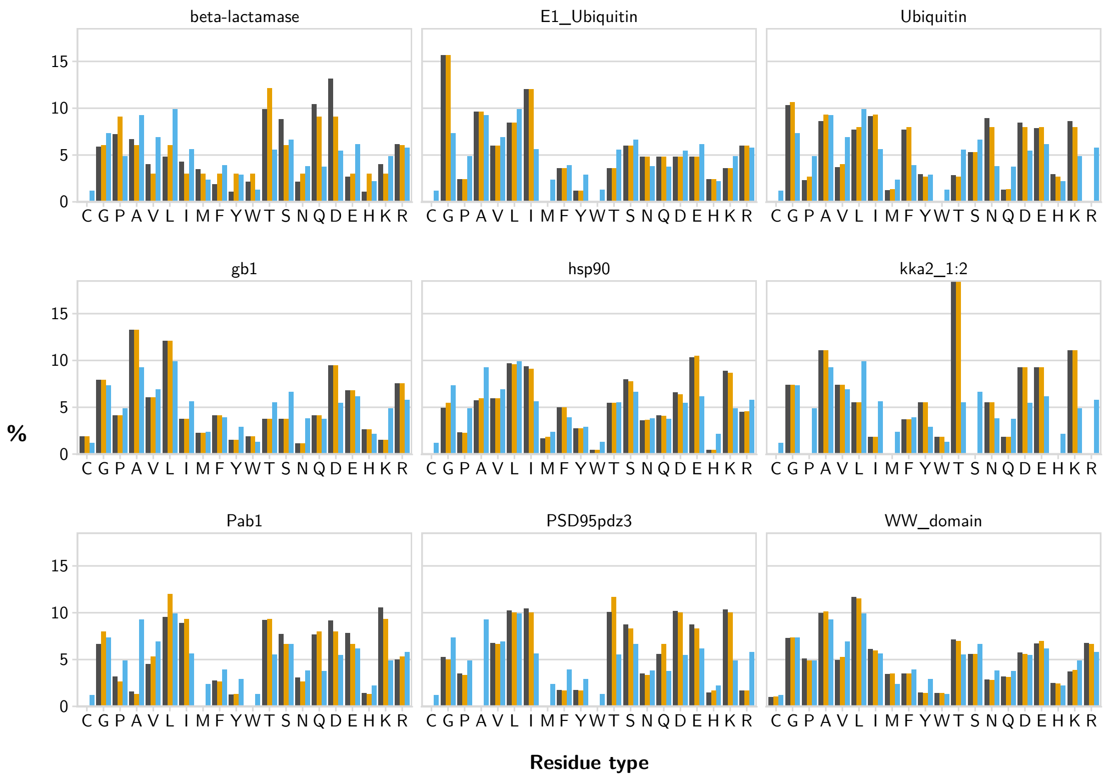


Figure S.3: Wild-type residue composition of each dataset compared to the wild-type protein sequence and the residue composition of UniProt. The composition of the respective datasets in terms of wild-type residues for each mutation is shown in grey, the residue composition of the wild-type sequence of the mutagenized protein region is shown in orange, and the composition of the whole UniProt is shown in blue. The UniProt composition refers to the release 2020_01, sourced from `https://www.uniprot.org/statistics/`.

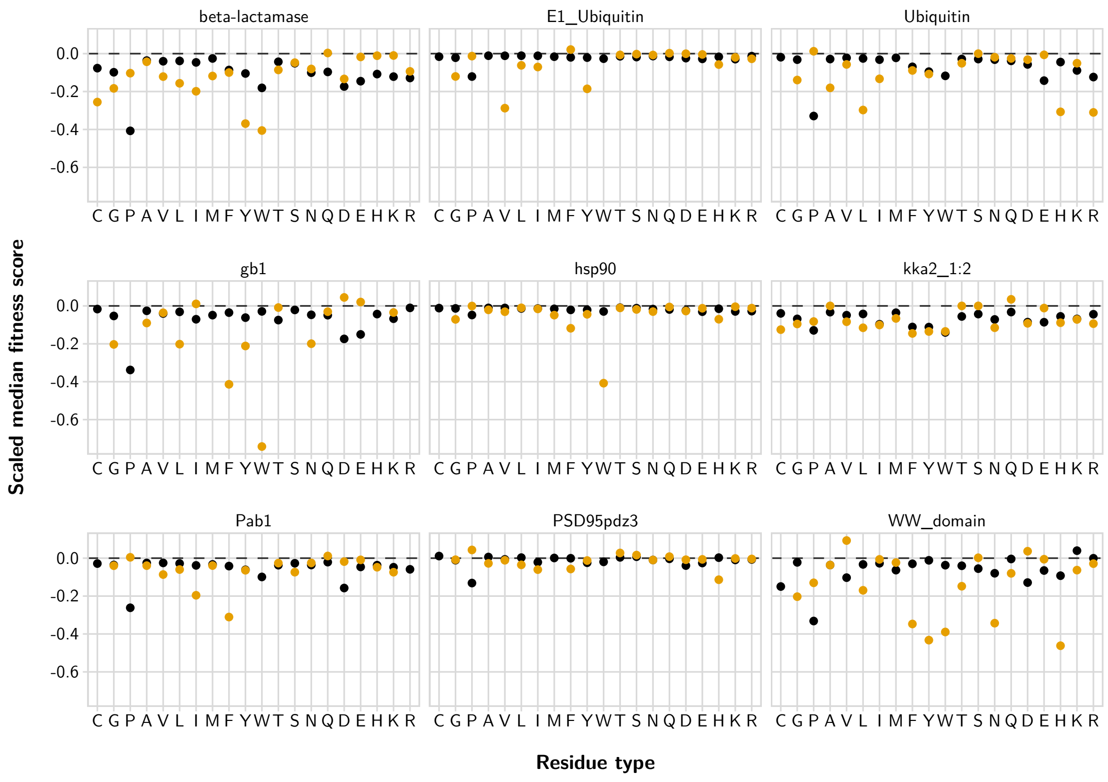


Figure S.4: Median score of mutations to and from each residue type, grouped by dataset. Orange dots represent the median score of mutations from the residue. Black dots represent the median score of mutations towards the residue. For this visualization, the fitness scores were scaled independently for each dataset to the range 0 to 1 and centered such that a neutral mutation would have a scaled score of 0. Scaling was applied before calculating the medians.

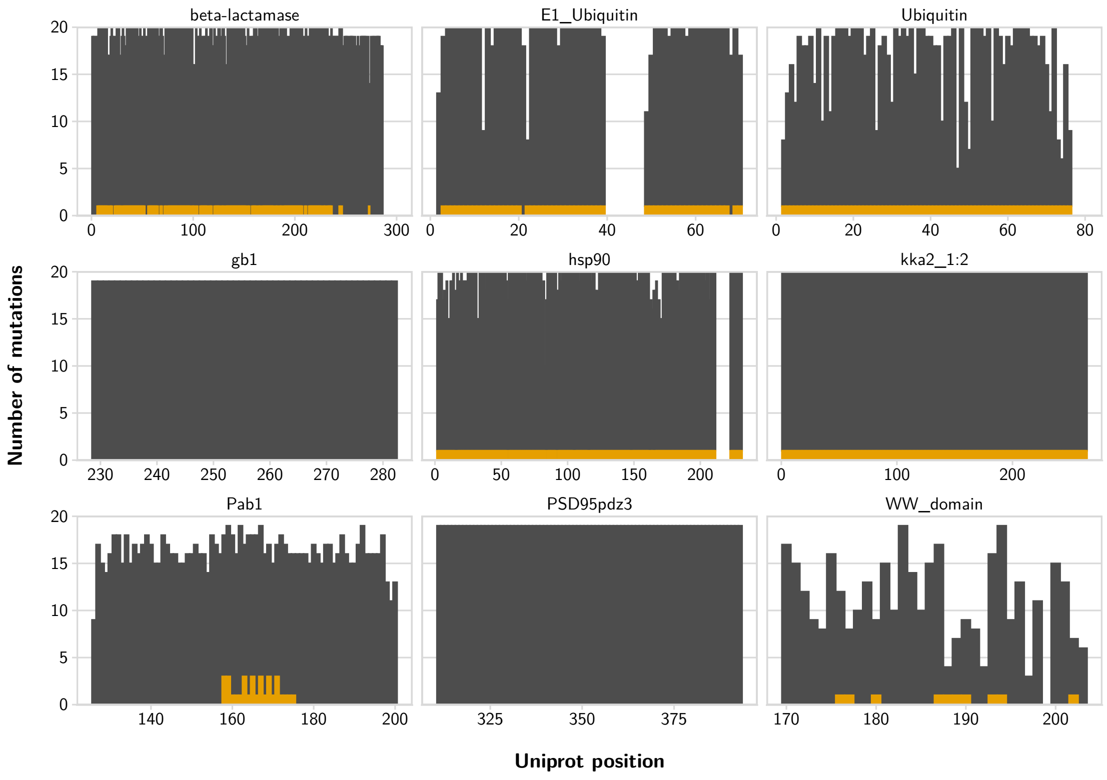


Figure S.5: Mutagenesis coverage along the primary protein sequence for each experiment. Series of stacked bar plots showing the number of mutations covering a given position along the primary sequence of each of the mutagenized proteins. The number of synonymous mutations is in orange and the number of missense mutations is in grey. Most experiments cover uniformly the respective mutagenized regions with 19 missense mutations, covering almost the full mutational landscape of the sequence. The dataset `E1_Ubiquitin` is missing all the mutations at position 40 to 48 (UniProt numbering). The dataset `hsp90` is missing all the mutations at position 212 to 222 (UniProt numbering). Synonymous mutations are available only for a limited region in `Pab1` and are absent from some of the datasets. `WW_domain` has a lower coverage compared to other datasets, not sampling the full mutational landscape of the protein. See Section 2.4.2 for a discussion about the missing positions.

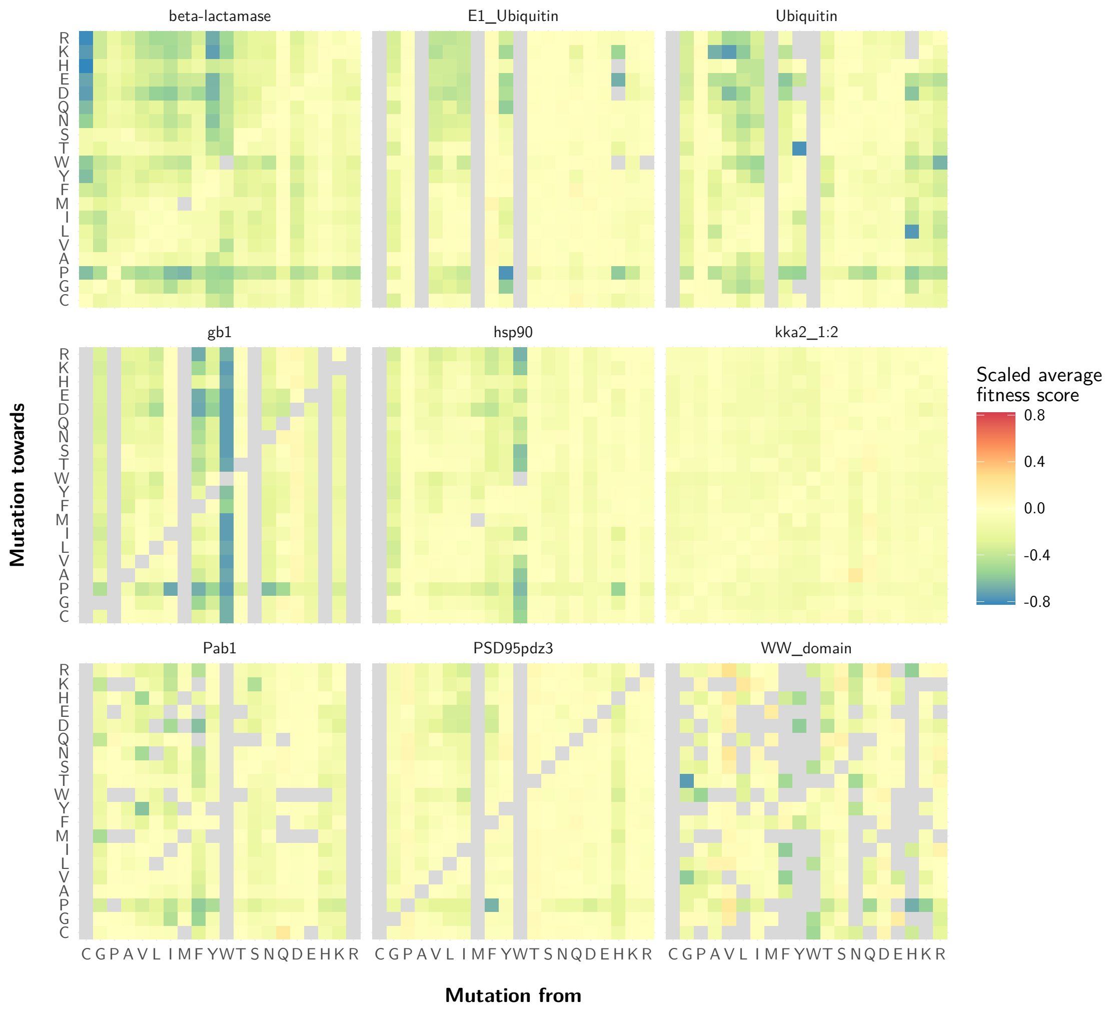


Figure S.6: Average fitness score for each possible mutation, by dataset. Series of heatmaps showing the average scaled fitness score for mutations towards and from any of the 20 standard residue types, for each dataset used in this study. Fitness scores were scaled to make them more comparable across datasets. Scaling was done on a per-dataset basis bringing the scores to the range 0 to 1 and subtracting the scaled value for a score of 0. Positive values indicate favoured mutations, negative values indicate damaging mutations, and a value of 0 indicates a mutation that has a neutral effect. Mutations that are not observed in the dataset are in grey.

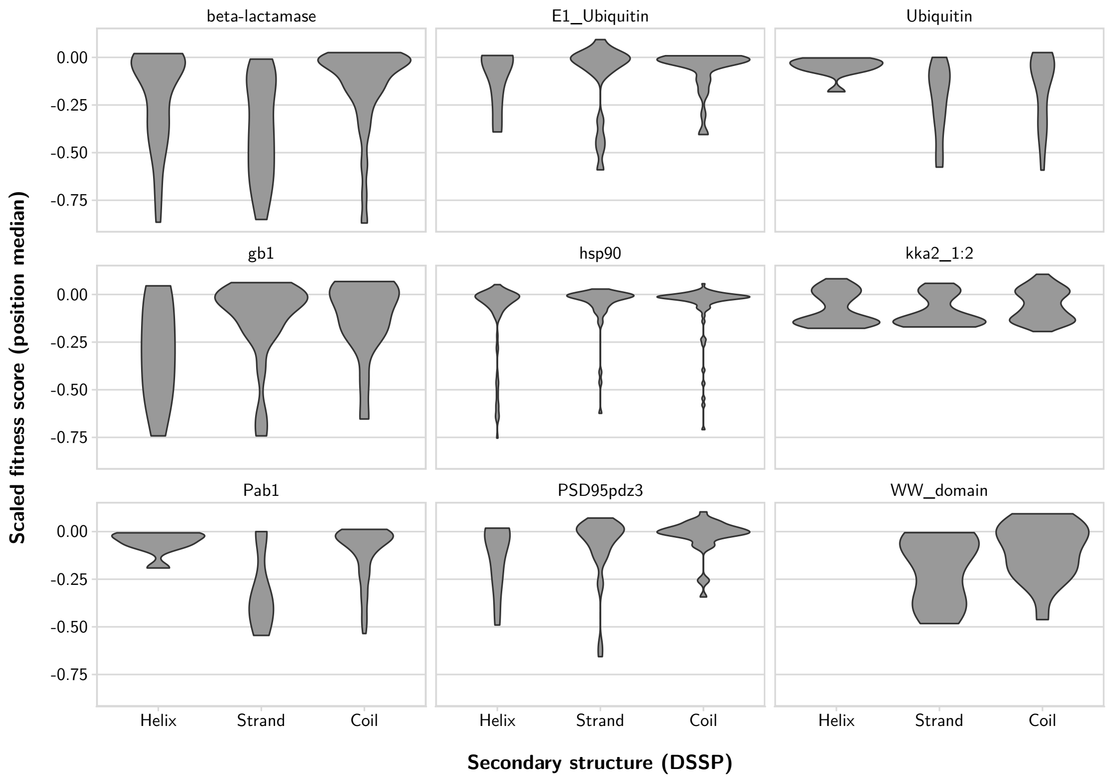


Figure S.7: Fitness score distribution by secondary structure, by dataset. Series of violin plots showing the distribution of median scaled fitness scores according to the secondary structure of the mutated position for each dataset used in this work. Fitness scores were scaled to make them more comparable across datasets. Scaling was done on a per-dataset basis bringing the scores to the range 0 to 1 and subtracting the scaled value for a raw score of 0. The median of the scaled fitness scores was taken by protein position and dataset. Only the subset of protein positions with an experimental secondary structure assignment is shown. The assignments were obtained as explained in Section 2.6.

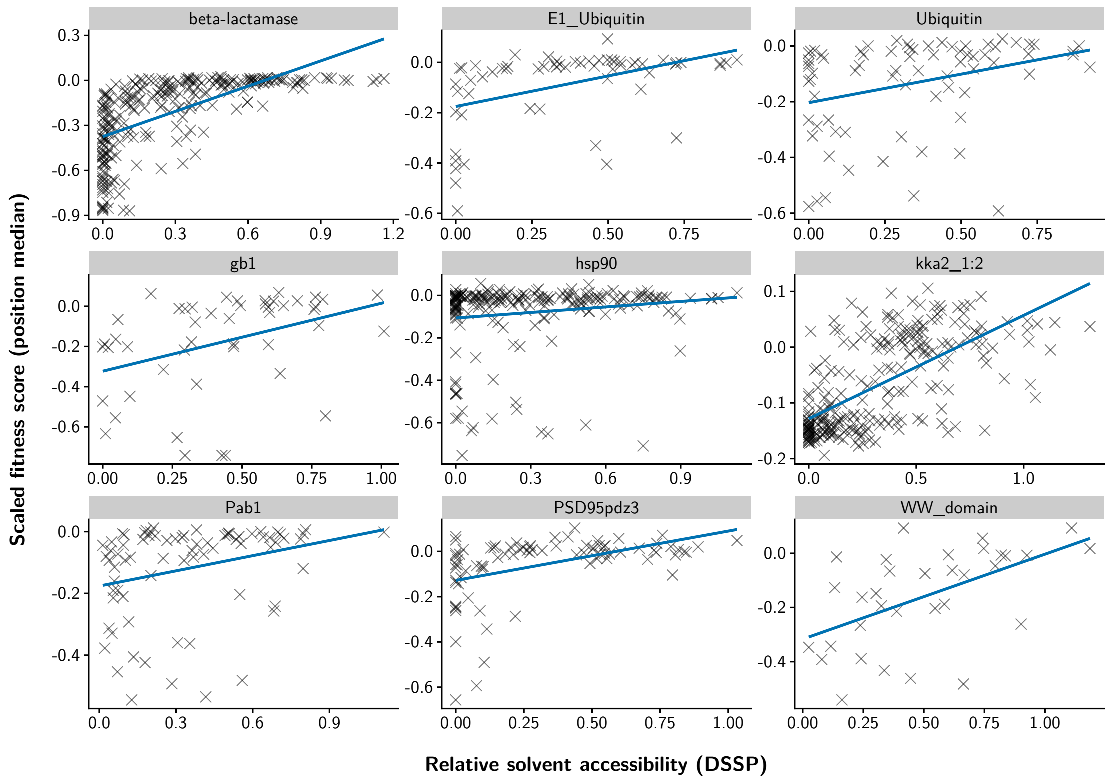
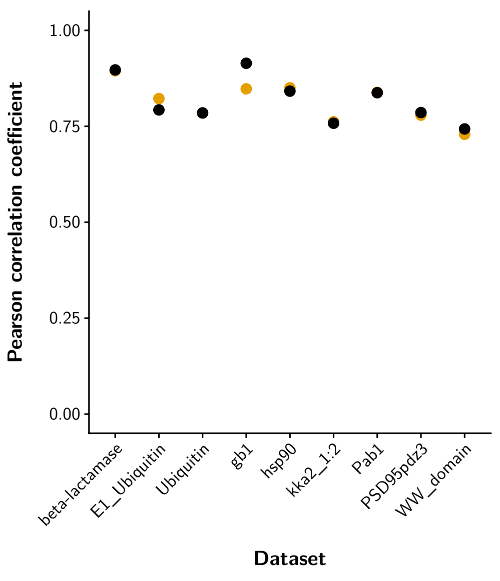
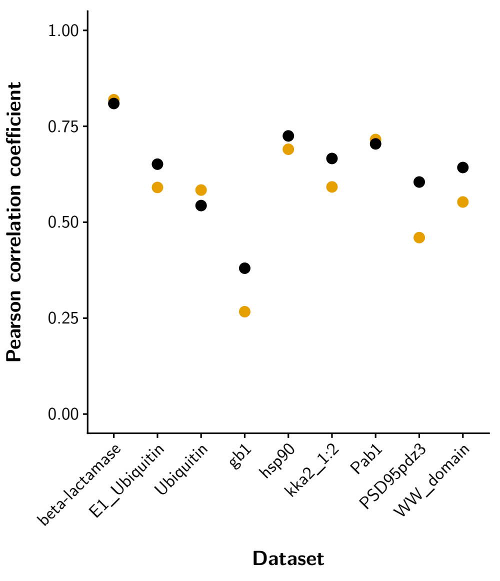


Figure S.8: Fitness score distribution by relative solvent accessibility, by dataset. Series of scatter plots showing the dependency between the experimental relative solvent accessibility of a mutated position and its median scaled fitness score. Fitness scores were scaled to make them more comparable across datasets. Scaling was done on a per-dataset basis bringing the scores to the range 0 to 1 and subtracting the scaled value for a raw score of 0. The median of the scaled fitness scores was taken by protein position and dataset.

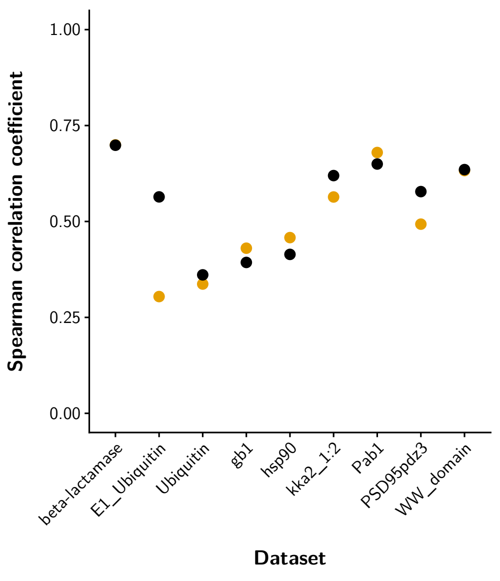
(a)



(b)



(c)



(d)

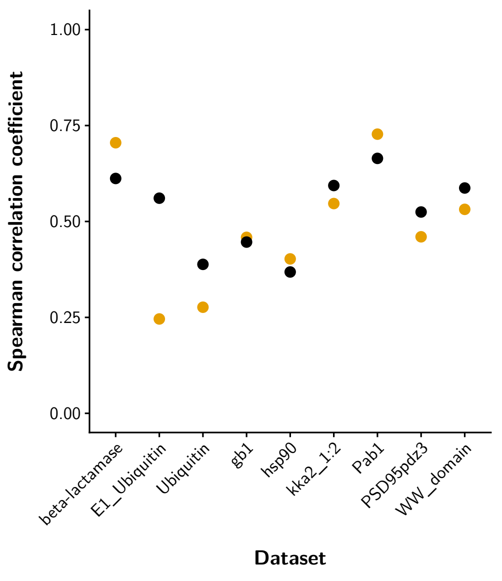


Figure S.9: Validation and testing performances. Series of plots showing the performances on the validation sets (orange points) and testing sets (black points) for the models developed in this work. **(a)** Naive single protein models. **(b)** Single protein models trained by segregating protein positions. **(c)** General gradient boosted tree models trained in a Leave-One-Protein-Out (LOPO) fashion. **(d)** General linear regression models trained in a Leave-One-Protein-Out (LOPO) fashion.

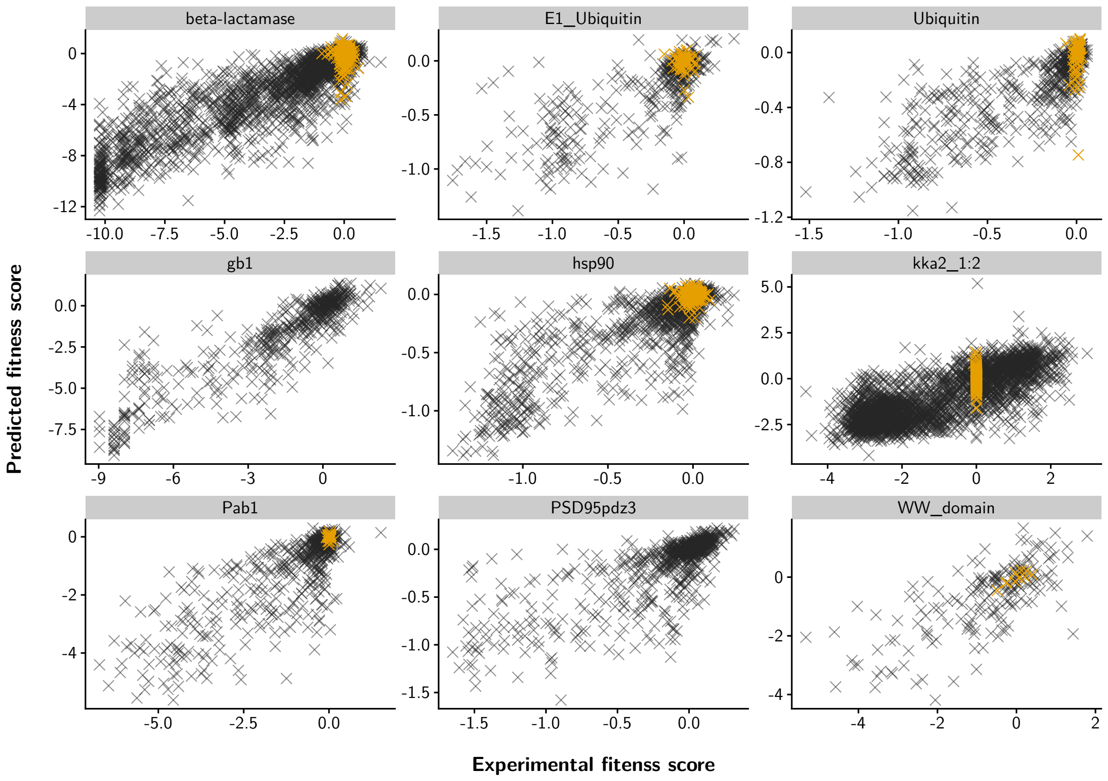


Figure S.10: Prediction correlations for naive single protein models. Series of scatter plots showing the correlations on the test set between the experimental fitness score and the predicted fitness score for the nine datasets used in this work. The predictions were obtained with independent gradient boosted tree models for each dataset. The validation and testing procedure followed the naive approach described in Section 2.7.1. Orange crosses are synonymous mutations. Black crosses are missense mutations.

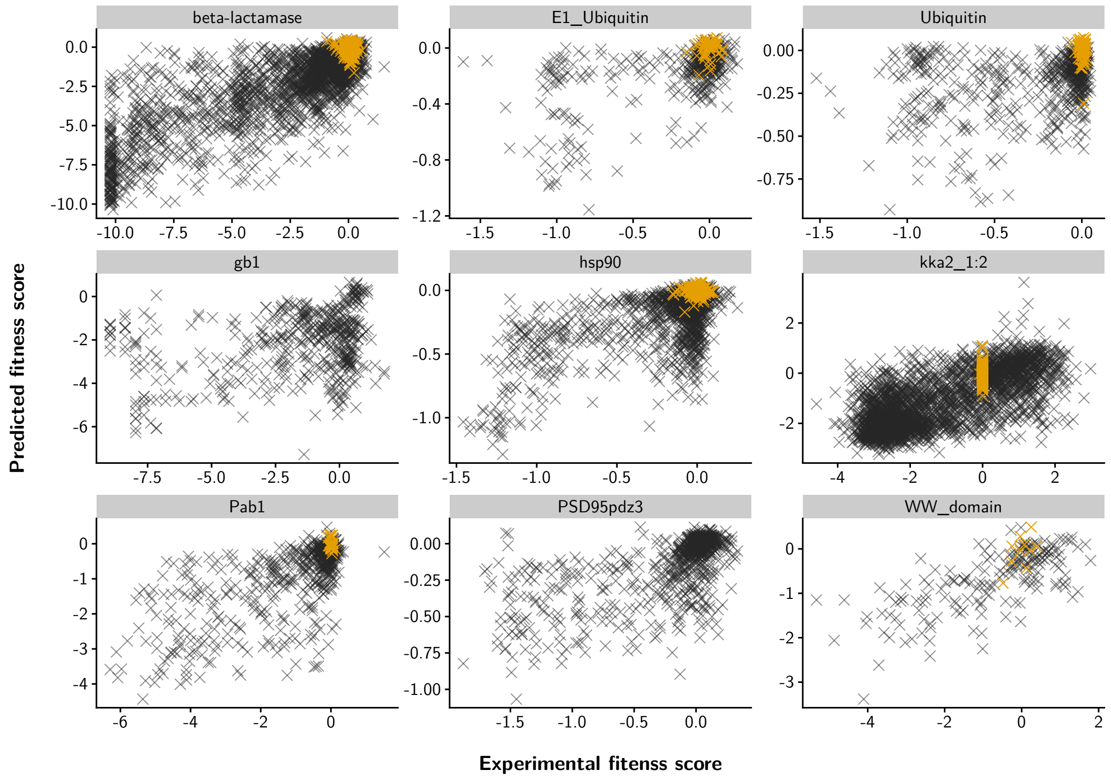


Figure S.11: Prediction correlations for single protein models trained by segregating protein positions. Series of scatter plots showing the correlations on the test set between the experimental fitness score and the predicted fitness score for the nine datasets used in this work. The predictions were obtained with independent gradient boosted tree models for each dataset. The validation and testing procedure followed the naive approach described in Section 2.7.1. Orange crosses are synonymous mutations. Black crosses are missense mutations.

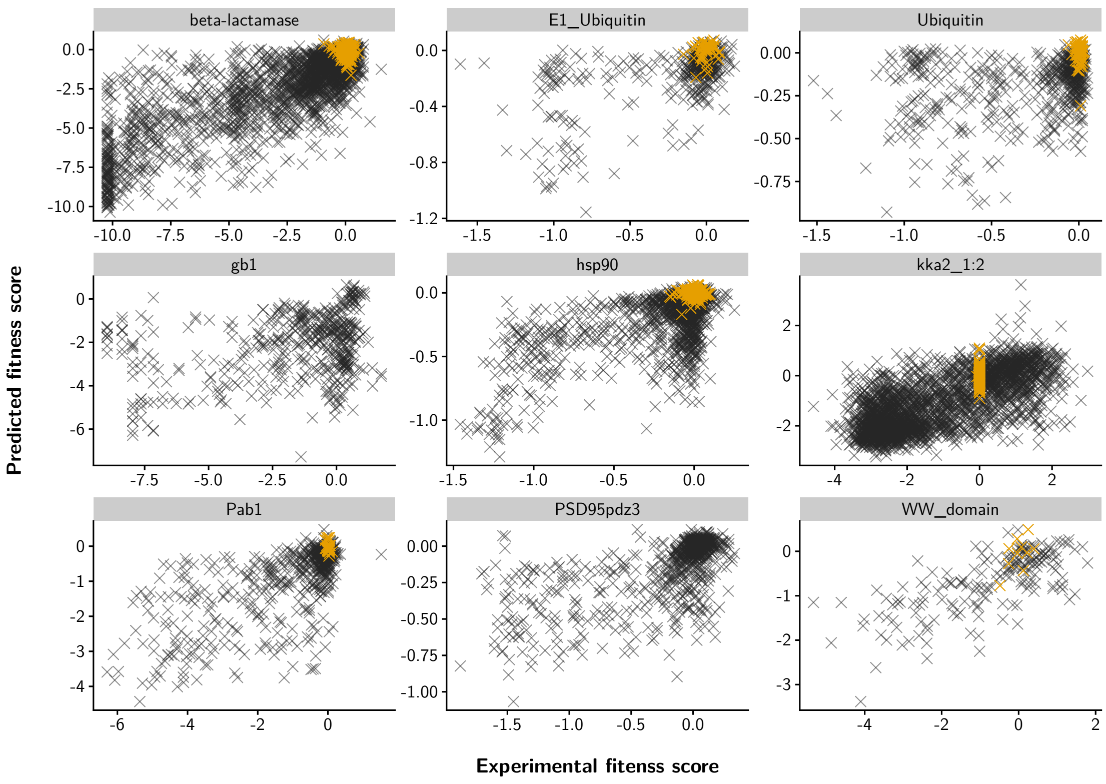


Figure S.12: Prediction correlations for the gradient boosted tree general models. Series of scatter plots showing the correlations on the test set between the experimental fitness score and the predicted fitness score for the nine datasets used in this work. The validation and testing procedure followed the naive approach described in Section 2.7.2. Orange crosses are synonymous mutations. Black crosses are missense mutations.

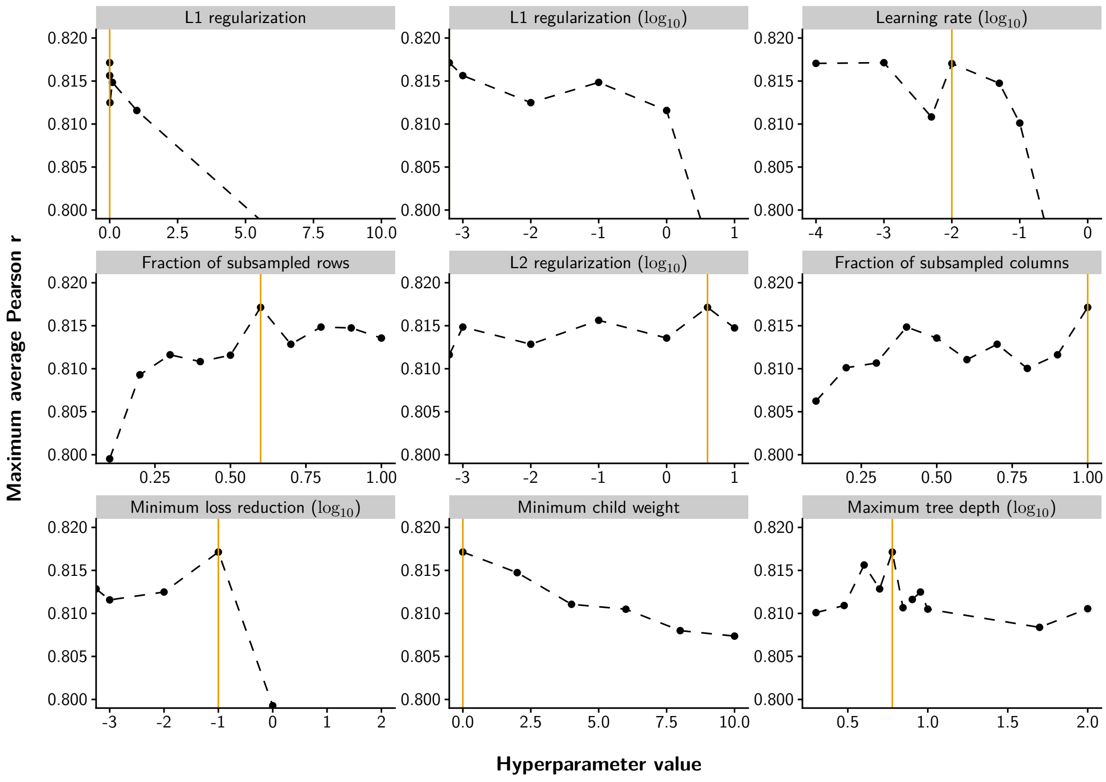


Figure S.13: Random hyperparameter search results for naive single protein models. Series of scatter plots showing the maximum average Pearson correlation coefficient for different values of the hyperparameters tested in cross-validation with a random search strategy. The Pearson correlation coefficient between the true fitness scores and the predicted fitness scores on the validation set was calculated independently for each dataset and each hyperparameter combination. The results were averaged across datasets, such that a single average Pearson r was obtained for each hyperparameter combination. The plotted quantities correspond to the maximum across all the random search iterations of those Pearson r averages, for a given hyperparameter value. The orange vertical line shows the chosen value for each hyperparameter, the value that was used in the final models. Some of the hyperparameter ranges are shown in logarithmic scale for ease of visualization. For the L1 regularization hyperparameter in logarithmic scale, the chosen value is 0, and it cannot be shown on a logarithmic scale.

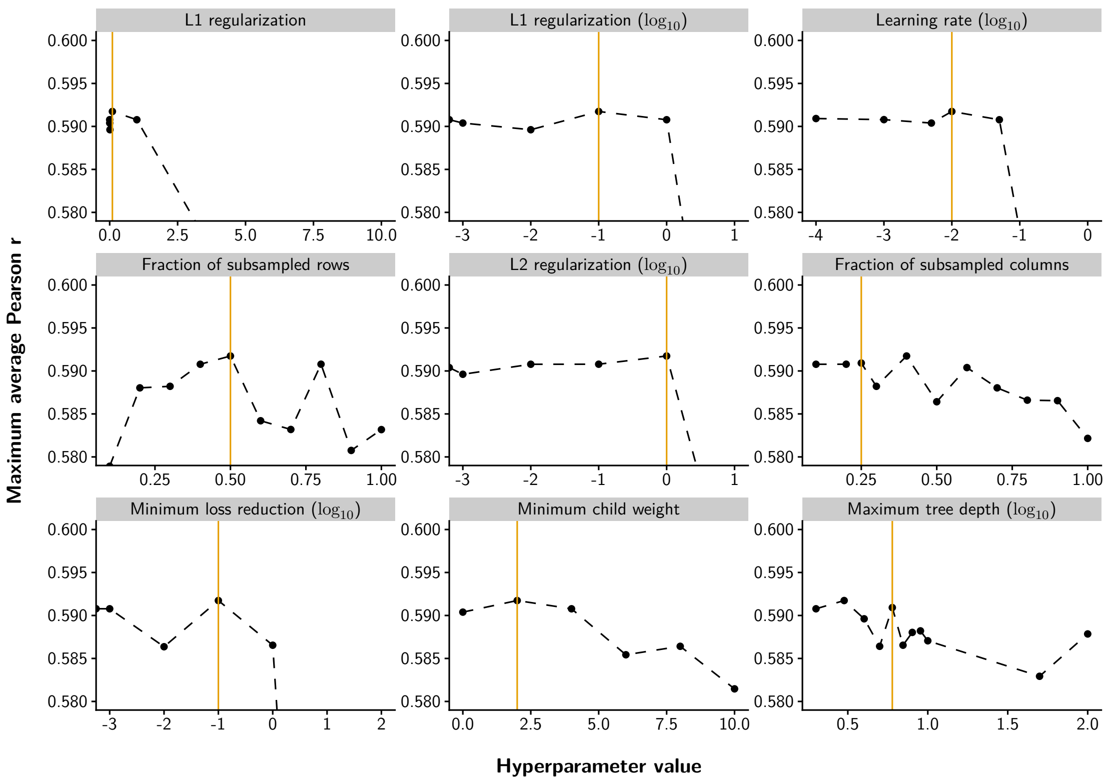


Figure S.14: Random hyperparameter search results for the single protein models trained by segregating protein positions. Series of scatter plots showing the maximum average Pearson correlation coefficient for different values of the hyperparameters tested in cross-validation with a random search strategy. The Pearson correlation coefficient between the true fitness scores and the predicted fitness scores on the validation set was calculated independently for each dataset and each hyperparameter combination. The results were averaged across datasets, such that a single average Pearson r was obtained for each hyperparameter combination. The plotted quantities correspond to the maximum across all the random search iterations of those Pearson r averages, for a given hyperparameter value. The orange vertical line shows the chosen value for each hyperparameter, the value that was used in the final models. Some of the hyperparameter ranges are shown in logarithmic scale for ease of visualization.

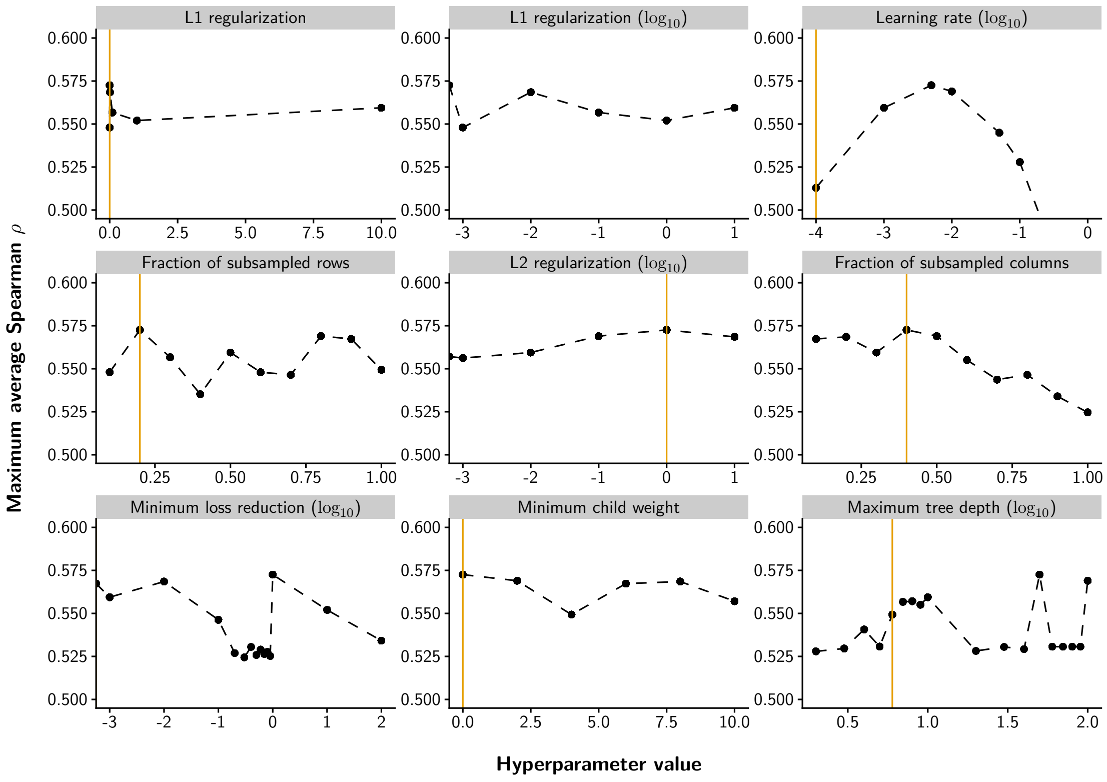


Figure S.15: Random hyperparameter search results for the gradient boosted trees general models. Series of scatter plots showing the maximum average Spearman correlation coefficient for different values of the hyperparameters tested in cross-validation with a random search strategy. The Spearman correlation coefficient between the true fitness scores and the predicted fitness scores on the validation set was calculated independently for each dataset and each hyperparameter combination. The results were averaged across datasets, such that a single average Pearson r was obtained for each hyperparameter combination. The plotted quantities correspond to the maximum across all the random search iterations of those averages, for a given hyperparameter value. The orange vertical line shows the chosen value for each hyperparameter, the value that was used in the final models. Some of the hyperparameter ranges are shown in logarithmic scale for ease of visualization. For the L1 regularization hyperparameter in logarithmic scale, the chosen value is 0, and it cannot be shown on a logarithmic scale.

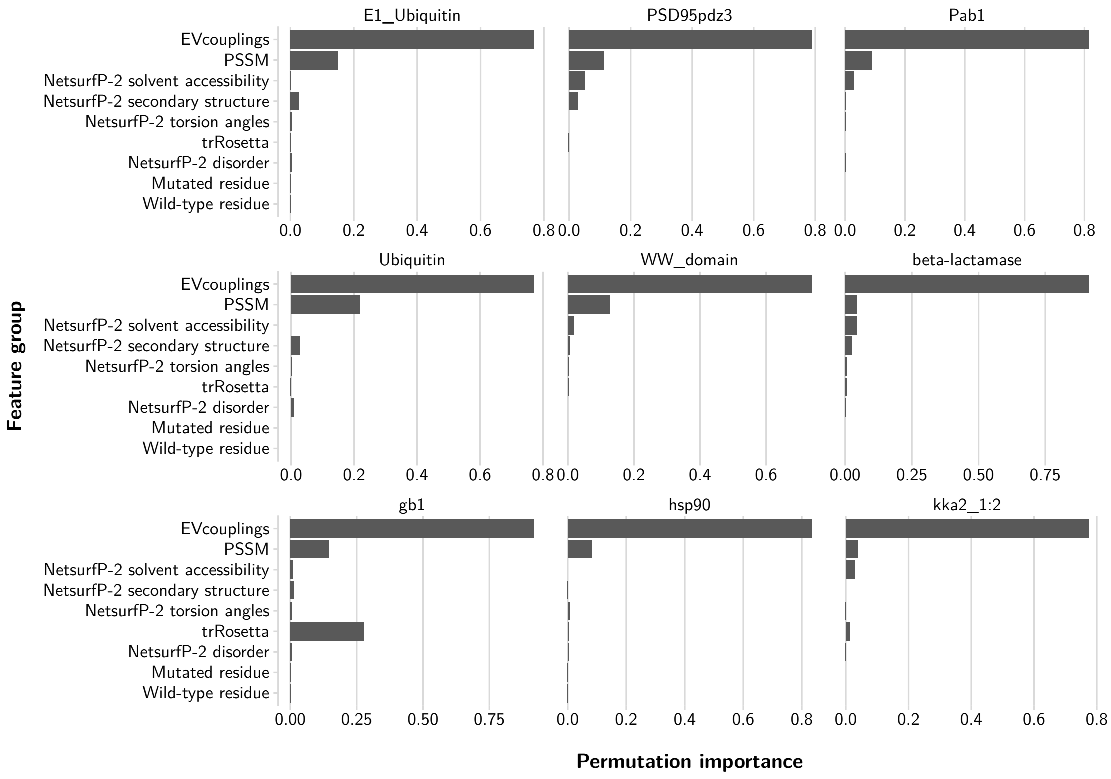


Figure S.16: Feature importance for the naive single protein models. Series of bar plots showing the grouped permutation importance of the features used in the single protein models trained with the naive approach, subdivided by dataset.

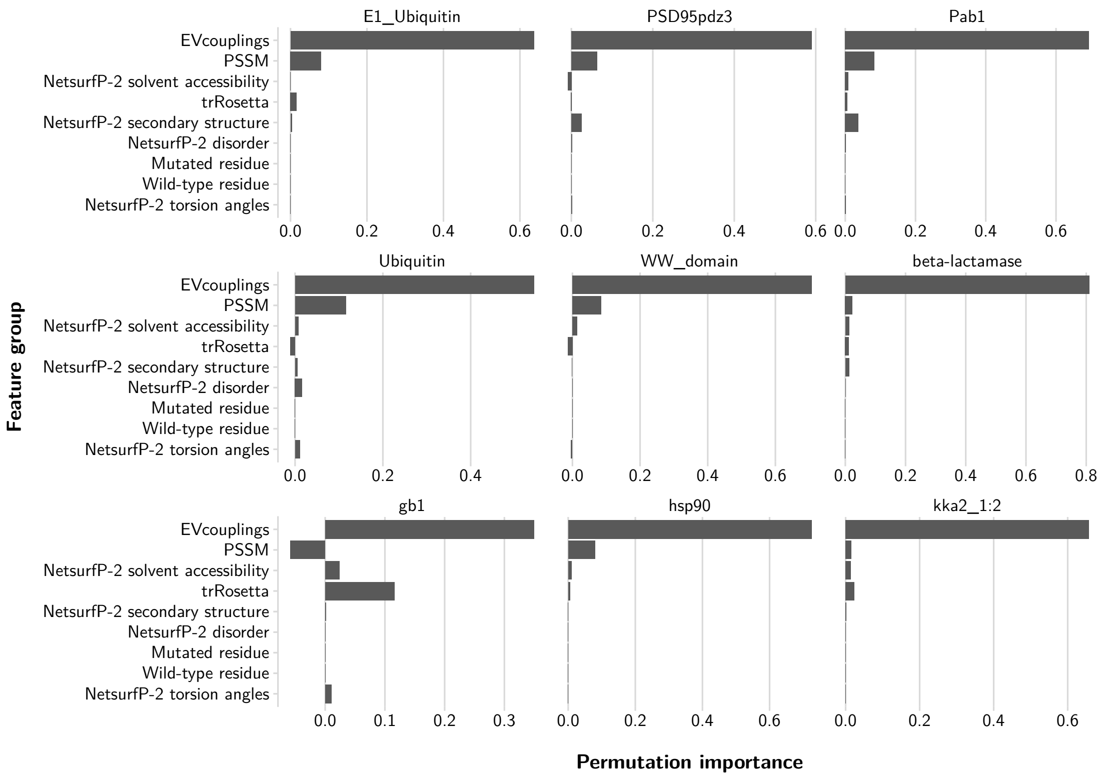


Figure S.17: Feature importance for the single protein models trained by segregating protein positions Series of bar plots showing the grouped permutation importance of the features used in the single protein models trained by segregating protein positions, subdivided by dataset.