BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

BENCHMARK OF THE COMPUTATIONAL TOOLS FOR THE PREDICTION OF THE EFFECT OF MUTATIONS ON PROTEIN STABILITY

BENCHMARKING VÝPOČETNÍCH NÁSTROJŮ PRO PREDIKCI VLIVU MUTACÍ NA STABILITU PROTEINŮ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. MATEJ BEREZNÝ**
AUTOR PRÁCE

SUPERVISOR — **Ing. MILOŠ MUSIL, Ph.D.**
VEDOUCÍ PRÁCE

CONSULTANT — **Ing. JAN VELECKÝ**
KONZULTANT

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

154611

Institut:	Department of Information Systems (DIFS)
Student:	**Berezný Matej, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Bioinformatics and Biocomputing
Title:	**Benchmark of the Computational Tools for the Prediction of the Effect of Mutations on Protein Stability**
Category:	Biocomputing
Academic year:	2023/24

Assignment:

1. Study the problematics of protein engineering and protein stability.
2. Study the process of the construction of the training datasets, metrics of evaluation, predictive tools, and the characteristics of the protein stability data.
3. Research the current state-of-the-art of predictors of the effect of mutations on protein stability.
4. Design a system for the fully automatized evaluation of the given set of mutations on the list of (mainly web-based) predictive tools.
5. Implement the designed system using the preferred programming language.
6. Evaluate the designed system on the experimentally verified dataset.
7. Discuss the results (consider different metrics, mutations, protein families, etc.).

Literature:

- Musil, M., Konegger, H., Hon, J., Bednar, D., Damborsky, J., 2019: Computational Design of Stable and Soluble Biocatalysts. ACS Catalysis 9: 1033−1054.
- Mazurenko, S., 2020: Predicting Protein Stability and Solubility Changes Upon Mutations: Data Perspective. ChemCatChem 12: 1-10.
- Stourac, J., Dubrava, J., Musil, M., Horackova, J., Damborsky, J., Mazurenko, S., Bednar, D., 2021: FireProtDB: database of manually curated protein stability data. Nucleic Acids Res. 49(D1):D319-D324. doi: 10.1093/nar/gkaa981.

Requirements for the semestral defence:
No requirements.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Musil Miloš, Ing., Ph.D.**
Consultant:	Jan Velecký, Ing.
Head of Department:	Kolář Dušan, doc. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	17.5.2024
Approval date:	30.10.2023

Abstract

Protein design necessitates understanding how mutations influence their stability. Numerous online predictors exist for this aim, but it is challenging to compare them or to use them collectively. For that purpose I developed BenchStab, a console application/Python package designed for the swift and straightforward operation of 18 predictors, gathering results from a series of mutants. Benchstab is freely available on GitHub and can be expanded to include more predictors. To avoid potential dataset bias towards some predictors, I have constructed a new unique dataset, sourced from FireProtDB. I utilized this dataset to assess 24 distinct prediction methods from the three different perspectives.

Abstrakt

Návrh proteínov vyžaduje informáciu o tom ako mutácie ovplyvňujú celkovú stabilitu proteinu. Pre tento prípad existuje mnoho verejne dostupných nástrojov avšak ich kolektívne používanie či porovnávanie je veľmi pracné. Presne pre tento prípad som vyvinul BenchStab; konzolovú aplikáciu/Python knižnicu navrhnutú pre rýchlu a priamočiaru manipuláciu s 18 prediktormi, umožňujúc hromadné získavanie mutačných výsledkov. Zároveň som vytvoril novú unikátnu dátovú sadu, získanú z FireProtDB. Tento dataset som použil na porovnanie 24 rôznych predikčných metód pomocou rôznych metrík.

Keywords

protein stability, prediction, benchmark, predictor, protein stability predictor, stability, protein stability benchmark, software tool, stability querying, structure, sequence, protein, dataset, DDG, FoldX4, Foldx5, DDMut, PoPMuSiC, Maestro, mutations, web-tool client

Klíčová slova

stabilita proteínov, predikcia, stabilita, benchmark, prediktor, softvérový nástroj, benchmark stability proteínov, získavanie stability, štruktúra, sekvencia, proteín, dataset, DDG, FoldX4, Foldx5, DDMut, PoPMuSiC, Maestro, mutácie, webový klient

Reference

BEREZNÝ, Matej. *Benchmark of the Computational Tools for the Prediction of the Effect of Mutations on Protein Stability.* Brno, 2024. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Miloš Musil, Ph.D. Consultant Ing. Jan Velecký

Rozšířený abstrakt

Proteíny sú biomolekuly, ktoré plnia kľúčové funkcie v každom živom organizme. Skladajú sa z aminokyselín, ktoré sú navzájom spojené peptidovými väzbami a tvoria dlhé reťazce. Tie sa následne skladajú do komplexnejších tvarov, nazývaných proteinové štruktúry. Proteíny sú nevyhnutné napríklad pre rast a opravu tkanív, produkciu hormónov, a zaisťovanie rôznych biochemických procesov. V snahe o zefektívnenie týchto procesov výskumníci pracujú na vylepšení ich vlastností, napríklad stabilizáciou ich proteínovej štruktúry pomocou mutácií. Pre tento pripad bolo vyvinutých mnoho nástrojov, ktoré pomocou rôznych *in silico* metód dokážu túto zmenu v stabilite proteínu predikovať. Avšak, zložitá obsluha, chýbajúce API, a samotný počet dostupných prediktorov spolu z rôznymi konfiguráciami často komplikuje ich aplikáciu na väčšie datasety.

Jedným z cieľov tejto diplomovej práce bolo preto vytvoriť nástroj, ktorý by tento proces zautomatizoval. Ako programovací jazyk som zvolil Python, a implementoval BenchStab, plne automatizovaný nástroj pokrývajúci celý process získavania predikcií z rôznych webových prediktorov. Pôvodným cieľom bolo zahrnúť všetkých 27 preskúmaných prediktorov, avšak vďaka ich nedostupnosťi či nefunkčnosti bolo nakoniec implementovaných celkovo 18 nájstrojov. Pre demonštráciu flexibility a rozšíriteľnosti nástroja som implementovaním prediktoru PonSol2 umožnil BenchStab-u získavať okrem stability aj predikcie rozpustnosti proteínu. BenchStab je verejne dostupný na GitHube [6] v dvoch podobách: ako konzolová aplikácia a Python knižnica. Samotný nástroj možno rovnako rozložiť na dve časti: predzpracovávanie vstupných dát a sieťovú komunikáciu s prediktormi.

BenchStab obsahuje robustné predspracovanie dát, dovoľujúc uživateľovi použiť všetky populárne formáty proteínov. Zároveň podporuje kaskádové spracovanie dát, tzn. zo zložitejších proteínových formátov (štruktúra) automaticky extrahuje tie jednoduchšie (sekvencia). To, spolu s efektínvym algoritmom na mapovanie mutačných pozicií medzi sekvenciami a štruktúrami umožnuje užívateľovi pomocou proteinových štruktúr získavať výsledky aj zo sekvenčných prediktorov. Za účelom minimalizácie hardvérových požiadavkov som nepoužil žiadne knižnice určené priamo na škrabanie webu, a namiesto toho implementoval celú komunikáciu s prediktormi iba za pomoci HTTP požiadavkov. Na to aby sa predišlo zbytočným prestojom v komunikácií z prediktormi, plynúcimi z povahy samotného predikčného procesu, bola celá sieťová komunikácia navrhnutá asynchrónne. Tento prístup garantuje semi-paralelný priebeh, zlepšuje rýchlosť nástroja a zároveň znižuje celkovú záťaž interných operácií.

Ďalším z cieľov tejto diplomovej práce bolo použiť tento nástroj na porovnanie výkonnosti jednotlivých prediktorov. Po analýze verejne dostupných dátových sád proteínovej stability som sa pre zaručenie nezaujatosti benchmarku voči niektorým prediktorom rozhodol vytvoriť vlastný dataset. Ako zdroj dát som použil FireProtDB, a po aplikovaní sekvencie netriviálnych krokov vznikol BenchStab dataset [91]. Celkovo sa v tomto datasete nachádza 289 mutačných dát pochádzajúcich z 36 proteínov. Zároveň tento datset vyznačuje unikátnou vlastnosťou; každý proteín z tohoto datasetu má menej než polovičnú sekvenčnú identitu z hociktorým iným proteínom pochádzajúcim z trénovacích sád prediktorov.

Samotný benchmarking nájstrojov na predikciu stability proteínov bol vyhotovený troma spôsobmi: na základe doby trvania predikčného procesu, pomocou grafickej analýzy zahrňujúc korelačné grafy a ROC/PR krivky a nakoniec za pomoci sady prediktívnych metrík ako vyvážená presnosť, stredná absolútna odchylka, Pearsonov korelačný koeficient atď. Analýza priemernej doby predikcie ukázala, že väčšina prediktorov, s výnimkou SAAFEC, PremPS a PoPMuSiC-u doručí svoje výsledky v rozumnom časovom okne (2 až 180 sekúnd). Naopak korelačné grafy ukázali, že výsledky väčšiny prediktorov dosahujú iba slabú ($\rho \in$

$[0, 0.3)$), poprípade miernu úroveň korelácie ($\rho \in [0.3, 0.5)$) s výsleskami získanými pomocou experimentálnych metód. Na základe ROC/PR kriviek je možné dedukovať, že väčšina prediktorov, s výnimkou prediktorov CUPSAT a Automute, prekonáva výsledky namerané náhodným prediktorom. Posledná z grafických analýz, analýza vlastností aminokyselín/proteínov za pomoci tepelných máp neukázala žiadnu výraznú odchylku od priemernej presnosti prediktorov.

Kvantitatívna analýza prediktívnych metrík dokázala, že väčšina nástrojov je použiteľná pre prioritizáciu mutácií (vyvážená presnosť v rozsahu 51% a 64%). Na druhej strane, takmer všetky nástroje vykazovali obzvlášť slabý výkon v regresii, t. j. v predpovedaní presnej zmeny stability proteínu (najhoršie a najlepšie R^2 sa rovnali 0,01 a 0,15) s častými falošne pozitívnymi aj chybnými negatívnymi identifikáciami. Navyše, veľká väčšina testovaných prediktorov vykazovala sklon k predikcií mutácií ako destabilizujúcich, čo bolo tiež preukázané priemernou znamienkovou odchylkou v rozsahu od -0,79 do -0,11.

V tejto práci som implementoval BenchStab, rýchly a automatický nástroj na získavanie predikcí z webových stránok prediktorov. K tomu som vytvoril unikátnu dátovú sadu s menej než polovičnou sekvenčnou identitou voči trénovacím dátam prediktorov. Pomocou tejto sady a BenchStabu som analyzoval a vyhodnotil predikívnu schopnosť 18 rôznych prediktorov. Výsledky potvrdili aplikovateľnosť prediktorov v praxi a zároveň poukázali na mierne nedostatky ktoré by mohli byť cieľom zlepšenia v budúcnosti.

Benchmark of the Computational Tools for the Prediction of the Effect of Mutations on Protein Stability

Declaration

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Miloše Musila, Ph.D. Další informace mi poskytl Ing. Jan Velecký. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....

Matej Berezný

May 17, 2024

Acknowledgements

First and foremost, I would like to thank both my supervisor Ing. Miloš Musil Ph.D, and my consultant, Ing. Velecký for their continuous support during the writing of this thesis. Next, I would like to thank Ing. Simeon Borko for an in-depth code review of BenchStab, catching all the details I would have otherwise missed. Finally, I would like to thank my family and friends for putting up with me in these dire times.

Contents

1 Introduction **3**

2 Proteins **5**
2.1 Amino acids 5
2.2 Amino acid interactions 8
2.3 Structure 10
2.4 Synthesis 12

3 Protein stability **14**
3.1 Quantification 14
3.2 Acquisition methods 17
3.3 Predition methods 18

4 Data **22**
4.1 Available sources 22
4.2 BenchStab dataset 23

5 Predictors **27**
5.1 Energy calculations 27
5.2 Machine learning 28
5.3 Meta predictors 29
5.4 Performance metrics 30

6 Application design **34**
6.1 Design 35
6.2 Data processing 37
6.3 HTTP communication 39

7 Implementation **43**
7.1 Preprocessing 44
7.2 Predictor clients 46

8 Results and discussion **49**
8.1 Prediction time 50
8.2 Visual analysis 51
8.3 Quantitative analysis 57

9 Conclusion **60**

Bibliography **62**

A Excluded datasets **72**

B ROC curves **73**

C PR curves **74**

D Kernel density estimate **75**

E Balanced accuracy heat maps **76**

F Mean absolute error heat maps **79**

G CLI usage hint **83**

H Example of predictor configuration file **85**

Chapter 1

Introduction

Protein molecules are known to be involved in every aspect that characterizes a living organism. For example, enzymes are proteins that enable and accelerate metabolic reactions, which are critical for the sustenance of life. The structure and morphology of various body parts are also largely dependent on the way proteins are assembled. They are also involved in the transport of different molecules, ensuring their delivery to desired locations. To perform these and many other functions, proteins adopt specific structures [1].

Throughout the years, the three-dimensional structure of a protein has been reshaped and altered countless times before reaching its current form. Mutations are a source of alterations in the genetic code that drive evolution at the molecular level. This allowed some protein molecules to develop into shapes that are more suited for the environment in which they reside and the tasks they perform.

Advances in DNA-reading techniques allowed access to many previously unknown properties of different naturally available proteins and their utilization in various medical, industrial, and biotechnical applications. This, coupled with recent success in the field of protein engineering, allowed improvements in different functional characteristics, such as stability and activity, increased efficiency, and allowed their repurposing to function in different non-native environments [48]. For example, the synthesis of an anabolic hormone called *insulin* with an artificially increased blood absorption rate positively affected the treatment of patients diagnosed with diabetes.

This thesis focuses mainly on the aspect of protein stability as a main characteristic determining the adoption of proteins in the harsh environment of medical and industrial applications. More stable proteins have a higher tolerance to the presence of organic solvents and are capable of operating at more extreme pH levels and temperatures [60]. These intriguing properties led researchers to focus on attempts to improve protein stability, leading to the development of many different methods capable of accomplishing the above task.

In the ideal scenario, every possible mutation in each position would be tested using laboratory techniques, such as saturation mutagenesis [78]. However, in the case of more complex proteins, the dimensions of the search space would be colossal; thus, laboratory evaluation would be nearly impossible. This limitation, along with the steady growth of computational resources, influenced the elevation of *in silico* methods. Unfortunately, the sheer amount of available prediction methods, coupled with long waiting times, different input formats between applications, and occasional service downtime, result in an extremely tedious and time-consuming process of evaluating large datasets across multiple predictors.

The main **objective of this thesis** was to develop an automated software tool capable of an efficient and almost effortless gathering of a multitude of predictions for proteins

and mutations from a wide range of different stability prediction tools in relatively short amounts of time. This would allow the user to comfortably collect and evaluate predictions of changes in the stability of a protein upon a mutation without the need to perform any additional manual work. Furthermore, with the lack of publicly available comparisons between different predictors, this tool could serve as a basis for future benchmarks of such prediction tools. Highlighting both their positives and negatives might allow less hesitation in selecting the best tools for a given problem.

This thesis is organized as follows. For a reader unfamiliar with the topic at hand, **Chapter** 2 is devoted to the introduction of proteins as biological entities. The chapter expands on all important factors related to protein stability, such as their structure, attributes, or interactions. This is followed by **Chapter** 3, which serves as a deeper dive into the problem of protein stability. It defines the term itself, describes various thermodynamic quantification variables, and provides insight into different protein stability measurement methods. **Chapter** 4 delves into the origins of accessible protein stability data, examining both their limitations and distinctive characteristics. Beyond summarizing the data, this chapter also details the extraction and rigorous filtering of the Benchstab dataset. This dataset was developed to serve as an impartial standard for all available protein stability prediction tools. The last theoretical chapter, **Chapter** 5 revolves around the use of these measurement methods as the core principles of protein stability prediction tools.

Chapter 6 examines the operational processes of online protein stability predictors and discusses the challenges associated with these predictors that were addressed in the design of the application. Furthermore, it presents the application's workflow, encompassing both data handling and network communication. This is followed by **Chapter** 7 delving deeper into the ideas presented in the preceding chapter, offering precise information on the applications' software tools and the logic behind their implementation. Finally, the **Chapter** 8 focuses on the evaluation of the predictors. It details the benchmarking methodology and examines the outcomes from the predictors from three different perspectives.

Chapter 2

Proteins

This chapter is devoted to a brief explanation of one of the largest, heaviest, and easiest-to-observe elements in a living cell, proteins. They are complex molecules (or macromolecules) that perform a variety of tasks, often crucial to our survival. A single living cell can contain thousands of different proteins, each with its unique function. For example, *collagen* and *elastin* is the main building components of fibers in tendons and ligaments, *hemoglobin* and *transferrin* are carriers that transport oxygen and iron throughout our circulatory system, and enzymes such as *pepsin* are catalysts for both the formation and destruction of bonds between atoms.

Individual subsections contain general information about the protein: what is a protein usually composed of (from amino acids to atoms), what are the interactions between its atoms, how does the uniqueness of individual amino acids influence the protein structure, and which are the most common observable structures present in proteins.

2.1 Amino acids

A protein molecule generally comprises one or more long chains of amino acid residues. As it can be noticed in Figure 2.1, individual amino acids in the protein are interconnected by covalent peptide bonds 2.2. In total, there are around twenty naturally occurring amino acids present in proteins. Each of these amino acids has, in addition to its unique name, assigned both three-letter and one-letter codes for its clearer representation in protein sequences. For example, the amino acid *cysteine* has a three-letter code **Cys** and a one-letter code **C**.

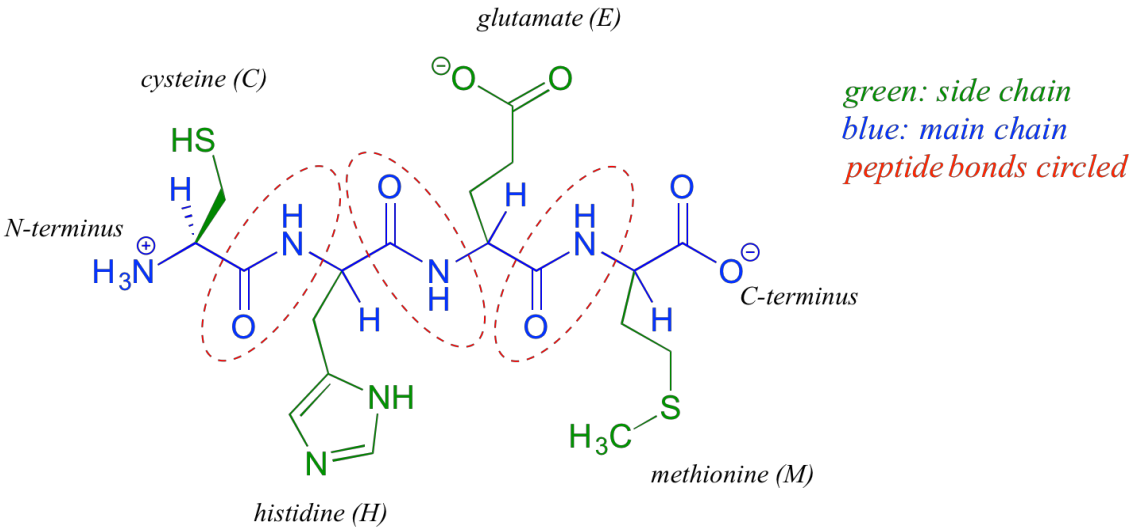


Figure 2.1: Image depicts different amino acids linked to each other forming the amino acid chain. The colors represent the two basic components of each amino -acid – common *main chain* (**blue**) in the middle, unique *side chain* e.g. cysteine (**green**) and its connection to the adjacent amino acids via *peptide bonds* (**red** dashed line). Image is taken from [82].

The structure of each individual amino acid can be separated into two essential components [82]: polypeptide backbone and the side chain. The polypeptide backbone, also called the *main chain* of the amino acid, generally comprises an ammonium group, a carboxyl group, and an alpha carbon (**C**α), which acts as a connection point for the side chain, also referred to as an R group in a field of organic chemistry. The side chains of amino acids usually do not participate in the formation of peptide bonds, but provide amino acids with their unique chemical properties. It is the uniqueness of the R group that defines the identity of the whole amino acid [82]. Despite the fact that the structure of every side chain can be considered unique, they do have some similarities with each other, allowing their classification into four different groups [54]:

Nonpolar amino acids

An amino acid is considered nonpolar if it does not have any constituent bearing a charge. This occurs when the total sum of its charge equals `0`, usually because its atoms share the same positive charge, causing obstruction of any potential electron sharing between each other. The most common component of the nonpolar amino acid R group is a hydrocarbon paired with an uncharged functional group and thus incapable of participating in any potential hydrogen bond formation.

These specific structural elements cause the amino acid molecule to be **hydrophobic** or "water-fearing", making it more likely to be located in the interior of the protein molecule after folding in aqueous environments.

Polar amino acids

In opposition to the neutral charge, polar amino acids have a net positive or net negative charge. The rate of polarity varies from chain to chain depending on the chemical properties of the molecules. For example, *serine*s shorter side chain and lack of ring structure makes

it less nonpolar than some other amino acids in this group (e.g. *threonine* or *thyrosine*). Although the side chains in this category of amino acids contain a multitude of different functional groups, most of them have at least one atom (*nitrogen*, *oxygen* and *sulfur*) capable of forming hydrogen bonds with water and other molecules.

Acidic amino acids

Since the backbone of the 20 amino acids present in proteins contains a **acidic** carboxyl group and a **basic** amino group, they can technically be classified as having acid-base properties. However, from an acid-base viewpoint, these groups neutralize each other, causing the side chain to play a crucial role in determining the molecule's acidity level. The two amino acids with such R groups are *aspartic acid* and *glutamic acid*. Both have carboxylic acid on their side chains, providing them with acidic (proton-donating) capabilities. In aqueous environments at physiological pH levels (ranging between 7.35 and 7.45 [35]), these amino acids are capable of becoming ionized by donating a hydrogen atom to the surrounding environment. This results in a negative charge, transforming the amino acids into their respective ionic forms, *aspartate* and *glutamate*. In this state, they are capable of forming ionic bonds, also called "*salt bridges*" or function as hydrogen bond acceptors [67].

Basic amino acids

Each of the amino acid side chains in this category contains a supplementary amino group, in addition to the amino group already existing in the core structure, which makes them considered basic molecules, and thus capable of accepting a hydrogen atom from an outside environment. This chemical property improves the water solubility of the amino acid, making these amino acids **hydrophilic**. Hydrophilic or "water loving" amino acids are more likely to be located in **exterior** of the molecule in the protein folding process.

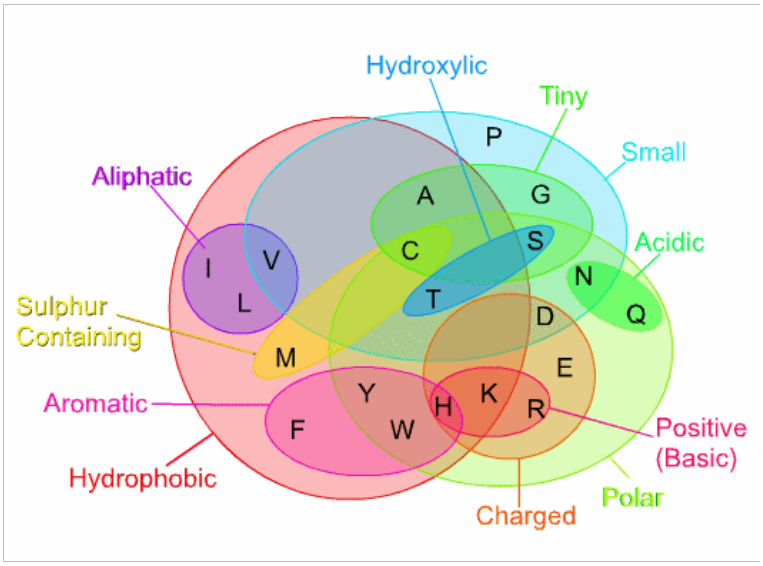


Figure 2.2: A Venn diagram depicting the categorization of twenty amino acids present in proteins by their chemical properties. Image is taken from [24].

As shown in figure 2.2, side chains can also be distinguished by their **size** (e.g., the smallest amino acid - *glycine*), hydrophobicity, or having some kind of special functional

group. For example, the R group of the *cysteine* amino acid contains **sulfur**, allowing it to form disulfide bonds.

2.2 Amino acid interactions

As hinted several times in section 2.1, amino-acid *bonds* can be explained as specific interactions occurring between various atoms in amino acids. For a specific bond to develop, the participating atoms must be at an adequate distance from each other. This distance, also known as a length of a bond is usually measured in *Angstrom* units (Å), where $1\,\text{Å} = 10^{-10}\,\text{m}$.

The two main bond categories that exist in amino acids are *covalent* and *non-covalent* bonds. The main difference between them is that formation of the **non-covalent** bond does not involve electron sharing as in its **covalent** counterpart, but rather a variety of different interactions between molecules (electrostatic, hydrophobic, Van der Waals forces, etc.).

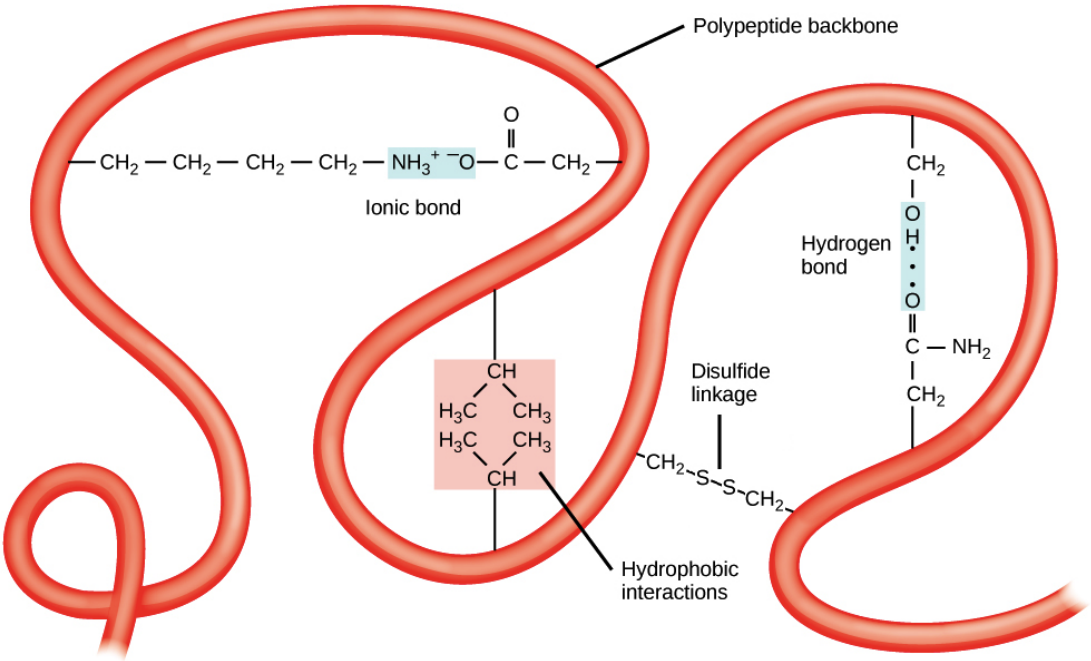


Figure 2.3: Picture shows the different types of bonds and interactions that influence the final shape of a tertiary protein structure. The bonding elements of these bonds are usually located in the *side chain* of the amino acid. Image is taken from [18]

Covalent bonds

Covalent bonds are the strongest chemical bonds found in protein molecules. They can be further separated into *peptide bonds* and *disulfide bridges*, depending on the location and atoms that participate in the formation of bonds.

Peptide bonds (`-CO-NH-`) are the most common type of covalent bonds, which connect repeating sequences of atoms present in amino acids, allowing the formation of the amino acid backbone. They develop when atoms from the carboxylate group (`COOH`) of one amino

acid share electrons with the amine group of another amino acid nitrogen ($\mathtt{NH_2}$), resulting in the release of two excess hydrogens and one oxygen, producing a water molecule

$$COOH + NH_2 = \text{CO–NH} + H_2O.$$

In addition to the peptide bonds that appear on the main chain of the amino acid, the *cysteine* molecules can also form covalent bonds with each other through the side chain, producing **disulfide bridges** ($\mathtt{-CH_2-S-S-CH_2-}$). The distance between individual atoms aiding in the formation of covalent bonds is usually shorter: around 2 Å for the *disulfide bridges* and 1.33 Å for the *peptide* bonds.

Electrostatic interactions

Hydrogen bonds – The formation of a hydrogen bond requires two atoms bearing partial negative charges to share a partially positively charged hydrogen or two atoms bearing partial positive charges to share a partially negatively charged hydrogen. Both of these cases lead to the formation of hydrogen bonds. Most hydrogen bonds in the protein are between the main chain `-N-H` and `-C=O` groups in alpha helices or beta sheets. The length of the hydrogen bond ranges between 1 Å and 3 Å.

Ionic bonds (salt bridges) develop after the complete transfer of one or more electrons between adjacent atoms [83]. The result of a such transfer is the electron-giving donor becoming a positively charged ion (cation), while the electron-receiving acceptor changes into an anion – negatively charged ion. Since these ions then bear opposite charges, electrostatic attraction occurs, resulting in the formation of an ionic bond. Sometimes, in hydrophobic interiors of the protein molecule, the strength of the ionic bond can approach the level of the peptide bond. The length of the ionic bonds is usually smaller than 5 Å.

Hydrophobic interactions

They are the consequence of the interaction between hydrophobic amino acids found in proteins and a polar solvent, water. Because these amino acids have a nonpolar hydrocarbon side chain, they are forced into close proximity to further distance themselves from the surrounding aqueous environment. These bonds are the main force driving the protein folding process, forming an interior hydrophobic protein core that protects amino acids from interactions with outside fluids. The length of hydrophobic bonds ranges from 2.6 Å to 3.3 Å.

Van der Waals forces

Similarly to hydrogen bonds, Van der Waals forces are weak attractions or interactions between molecules. They exist because every atom has an electron that can fluctuate, resulting in a temporary dipole. If the atoms are at a sufficient distance from each other, the temporary dipole in one atom could lead to the development of the complementary dipole in the other atom, creating a weak electrostatic attraction, called *Van der Waals force*. Although these forces are weak, they can be considered to be an important part of the protein structure because of their sheer number. The distance between atoms connected by van der Waals forces ranges between 4 Å and 6 Å. A shorter distance causes the atoms to repel each other due to the negatively charged nature of their electrons, while a long distance prevents the formation of electrostatic attractions.

2.3 Structure

Each protein has its own three-dimensional structure determined by the order of amino acids in its amino-acid chain. The final folded structure of the protein is called *folded conformation* [3], and is usually characterized by energy aspects: every protein tries to fold itself with the least amount of free energy. In addition to computational predictions, three-dimensional structures can be obtained through laboratory procedures such as X-ray crystallography [81], NMR spectroscopy [79] or cryo-electron microscopy [22]. The results are then generally stored as PDB or CIF files in a large, freely accessible database of three-dimensional structures *Protein Data Bank* (PDB) [95]

From the simple amino acid sequence to the union of multiple proteins folded into different complex shapes, the four most commonly known protein structures can be recognized. Each of the following structures highlights specific components that directly participate in or indirectly influence the final product of the folding process. The following structures, illustrated in the Figure 2.3, are:

Primary structure

The initial state of the protein at the beginning of the assembly process is called *primary structure*. The primary structure represents a group of one or more sequences of amino acids in a polypeptide chain, linked to each other through various amino acid interactions. Each chain has its own set of amino acids, arranged in a particular order. The primary structure, along with knowledge of the local biological environment in which the protein resides, allocates most of the information required for the prediction of its final three-dimensional shape. Every primary structure is known to start from an *amino-terminal* (`N`) end to the *carboxyl-terminal* (`C`).

Secondary structure

The secondary structure can be considered as a local spatial transformation of the linear amino acid sequence with the exclusion of its side chains. Secondary structures are the result of the degree of freedom provided by peptide bond connections throughout the entire sequence, causing individual parts of the chain to rotate around axes defined by bonds between `N` and C-α (ϕ torsion angle) or between C-α and `C` (ψ torsion angle) as well as various other amino acid interactions. Regular secondary structures are as follows:

- **α-helix** – the formation of a α helix can be described as a right-handed helical conformation of the primary structure, with close resemblance to a 3D spiral. Usually, there are roughly 3.6 residues per turn, with hydrogen bonds formed between *carbonyl* (`CO`) of the i^{th} residue and `NH` of the $(i+n)^{th}$ residue, where typically $n = 4$, but there are cases where $n = 3$, forming the less common 3_{10} α-helix. These residues act as supporting pillars of the α-helix.

- **β-sheet** – another common secondary structure, the β sheet, can be viewed as a set of multiple adjacent β strands - secondary structures where the shape of the protein chain almost perfectly resembles that of a straight line, connected via hydrogen bonds. There is no requirement for β-strands to exist close to each other in primary structure, so they are often separated by long sequences of amino acids that do not participate in a given β-sheet structure. β-sheets can be further separated into *parallel*, *anti-*

parallel, *mixed* β sheets, given by the orientation of the individual β strands in the sheet.

- **Turns and loops** – these secondary structures serve primarily as a connecting agent between α helices and β sheets in proteins. Usually, they take on irregular shapes, with the exception being **reverse turns** (β-turns or β-bends) - tight turns connecting adjacent antiparallel β-strands. Their distinctive feature is a hydrogen bond between `CO` of the first residue in the turn and `NH` of the fourth and last residue in the turn. The term *loop* usually refers to elements of secondary structures with less regular hydrogen bonding patterns than in other cases (α-helix, β-sheet, β-turn).

Although their length may vary from structure to structure, they are easily recognizable by their characteristic torsion angles of the main chain ψ and ϕ, and by the patterns produced by hydrogen bonds.

Tertiary structure

One of the simplest ways to understand what the tertiary structure represents is usually to envision a shape of a fully folded protein in three-dimensional space. The tertiary structure will often consist of a polypeptide chain backbone and single or multiple secondary structures, called *protein domains*. The tertiary structure of the protein is defined by its atomic coordinates. These coordinates may refer to either a single protein domain or the entire tertiary structure [9].

Noncovalent interactions (hydrophobic, electrostatic, hydrophilic, etc.) between different amino acid side chains are usually one of the main contributors to the final shape of the tertiary structure. However, as shown in Figure 2.3, in addition to noncovalent bonds, there is a special type of covalent side chain interaction (*disulfide bridge*) that also contributes to the final alignment of the tertiary structure.

Quaternary structure

There are many instances where proteins function as a singular unit; hence, they contain only one polypeptide chain, making them capable of visualization in only the first three structural levels (primary, secondary, and tertiary). But in some cases, the protein structure may consist of multiple polypeptide chains, called *subunits*. The arrangement of such subunits into subunit complexes is called *quaternary structure*.

This structure describes the exact number and arrangement of various molecule organizations, ranging from simple dimers to large complexes with a multitude of subunits. For example, *hemoglobin*, a protein present in blood cells responsible for oxygen transport in our cardiovascular system, is made up of two α chains and two β chains.

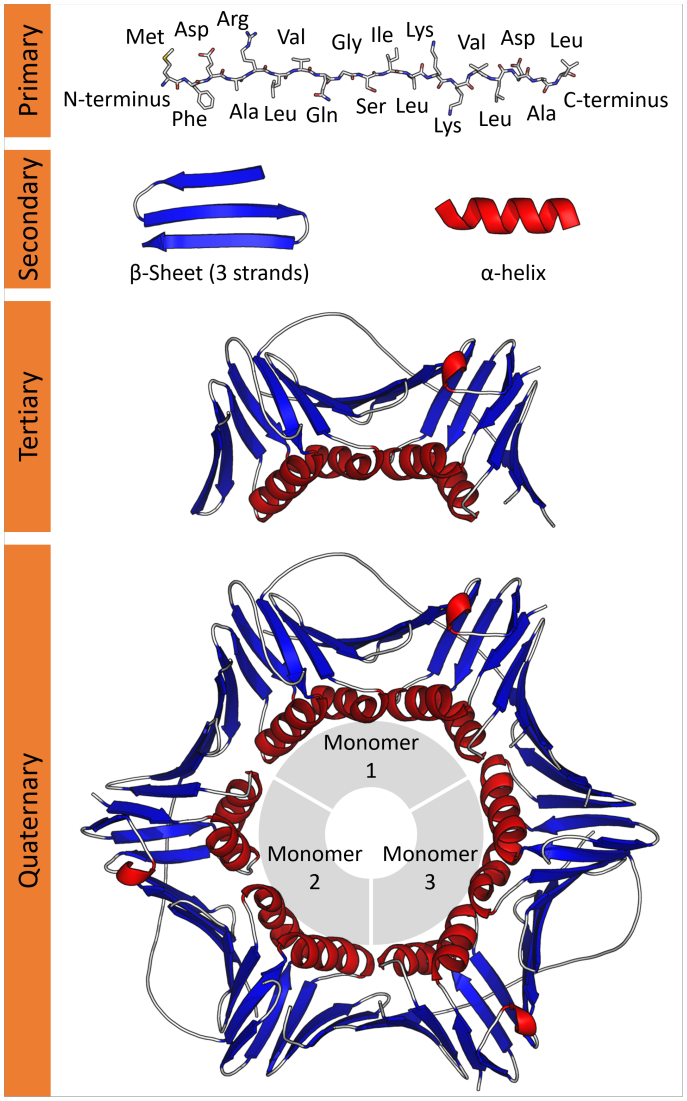


Figure 2.4: illustrating all commonly known protein structures present in protein complex PCNA (PDB: 1AXC).
Primary structure represents the linear sequence of three-letter codes representing specific amino acids present in the protein.
Below primary structure are depicted the most common secondary structures α-helix and β-sheet.
Tertiary structure shows the three-dimensional conformation of a protein sequence.
Quaternary structure describes the number and arrangement of multiple tertiary structures (folded protein subunits) in a subunit complex.
All illustrations of the protein structures present in this Figure (besides the primary structure) show only their *backbones* without *side chains*, but those are usually responsible for the final form of the tertiary/quaternary structure. The backbone is portrayed as a set of differently colored *ribbons*, so the whole visualization bears similar name *ribbon model*. Image is taken from [77]

2.4 Synthesis

The process of assembling proteins is called *synthesis*. There are two separate approaches to protein synthesis:

- *in vitro*, or **chemical** synthesis, is a production of proteins using biological machinery in a cell-free system [32].
- *in vivo*, or **biological** synthesis, represents a natural formation of protein molecules. This biological process also called *gene expression*, is divided into two different stages, *transcription* and *translation*.

Transcription

Transcription is responsible for copying and transferring genetic instructions from DNA to mRNA. In eukaryotic cells, the process itself takes place inside the nucleus of a cell. During

transcription, DNA is used as a template to transfer genetic information and produce the mRNA strand complementary to its DNA template. There are three distinct stages of transcription:

- **Initiation** is the starting point of transcription. Transcription is initiated when the RNA **polymerase** enzyme binds to a specific repeated sequence of nucleotides, called the **promoter** gene. This signals the DNA to "untangle", so the RNA polymerase will be able to read the individual bases of one of the DNA strands. RNA polymerase then proceeds to synthesize the new mRNA strand, beginning the **elongation** phase.
- **Elongation** is responsible for the extension of the mRNA strand, thus adding new bases to the mRNA nucleotide sequence.
- **Termination** ends the transcription process. The synthesized mRNA strand is detached from its DNA template.

In eukaryotic cells, this newly transcribed mRNA is not yet ready for translation. It is currently in a state of **pre-mRNA** and requires additional processing, such as *splicing*, *editing*, *polyadenylation*, producing mature mRNA, ready to continue the protein synthesis process.

Splicing is accountable for removal of **introns** from mRNA by small protein complexes called *ribonucleoproteins*. Introns are noncoding mRNA regions that do not participate in the translation process. Spliced mRNA is then composed of only coding regions, called **exons**.

Sometimes, nucleotides might be added, deleted, or changed in the mRNA sequence to modify the final shape of a future protein. For example, human protein *APOB*, which is useful for transporting lipids in the blood, has two different forms, one smaller than the other. This is a result of **editing**, where an early stop signal nucleotide sequence is inserted into the mRNA, forcing the translation process to terminate faster.

Finally, the *poly-A* tail, comprised of multiple adenine bases, is added to the 3' end of the already spliced and edited mRNA strand, signaling the end of the mRNA. Another function of the poly-A tail is to protect the mRNA from enzymes that break it down while it is transported outside the nucleus. This process is called *polyadnylation*.

Translation

Translation transforms the genetic code included in the mRNA into a sequence of amino acids. After the mRNA leaves the nucleus, it is transported to *ribosome*, a complex consisting of rRNA and proteins. The ribosome then proceeds to read the nucleotide triplets, called *codons*, from the mRNA strand, assembling the amino acid sequence with the help of **tRNA**.

tRNA is responsible for fetching the right amino acids for the ribosome to construct the right polypeptide chain. Each tRNA has an anticodon, complementary to the codon present in the mRNA sequence. This ensures that most of the time, the newly synthesized polypeptide chain is comprised of a sequence of correct amino acids in exactly the same order as in its mRNA template.

Chapter 3

Protein stability

This chapter is an introduction to protein stability, the motivation behind its research, and a description of the techniques used for its quantification. The main sources used in this chapter were *Protein Bioinformatics* [33], and *Computational Design of Stable Proteins* [51].

Because the extent of evolution of native proteins often does not exceed the mild conditions inside living cells, these proteins often struggle to survive harsher and more extreme industrial conditions. To combat this "softness", there is a growing interest in improving **protein stability** and developing more stable proteins that can then be used in a wide range of medical and biotechnical applications.

Protein stability can be considered as a net balance of forces that determines to what extent the protein will be in its native folded conformation (tertiary/quaternary structure) in opposition to a denatured state (unfolded). The net stability of proteins may be interpreted as a difference between two large opposing forces. The net stability of complex proteins, stabilized by various amino acid interactions, is typically miniature in comparison to those of their unfolded states, often dominated by entropic and non-entropic free energies.

Protein stability can be modified by applying artificial **mutations** to the protein sequence, DNA, causing changes in its final three-dimensional arrangement. A mutation is an alteration in the nucleic-acid sequence and can be categorized into four different types: substitution, insertion, deletion, and translocation. As the name might suggest, **substitutions** are also called point mutations since they affect only a single base pair of DNA, substituting it with another base pair. Substitutions are the most commonly appearing type of mutation. **Insertion** is a result of the addition of new bases to the nucleotide sequence, and **deletion** is the result of their removal from the nucleotide sequence. For example, the result of the insertion of three nucleotide bases is the addition of whole amino acids in the translation process. Changes smaller than that are termed frameshift mutations, because they shift the reading frame of the codons during translation, causing tRNA to fetch completely different amino acids, thus significantly altering the final amino acid sequence. A combination of insertion and deletion is called **translocation**, where a subsequence is moved from one region into another.

3.1 Quantification

The two most commonly used numerical measurements in the biotechnical area of research to evaluate protein stability are Gibbs free energy and the melting temperature.

Gibbs free energy

Gibbs free energy G is thermodynamic potential, which can be used to calculate the maximum amount of work performed by the thermodynamically closed system at a constant temperature and pressure. Gibbs free energy ($G[J]$)is defined accordingly:

$$G = H - TS, \tag{3.1}$$

where $H[J]$ is *enthalpy*, property of the thermodynamic system, $S[JK^{-1}]$ symbolizes *entropy* of the protein and $T[K]$ is the absolute temperature. **Enthalpy** represents the total sum of the internal energy in the protein (U) and the product of its pressure (p) and volume (V) [2]:

$$H = U + pV. \tag{3.2}$$

The value of enthalpy in a protein decreases during the folding process, mainly because of events such as optimization of the polar group orientation, and systematic grouping of the hydrophobic side chains together. These, and other types of interactions help proteins achieve near-perfect proximity to their ideal bond lengths and angles. **Entropy** is both a scientific concept and a measurable property that is generally related to order/disorder, randomness, or uncertainty [92]. In proteins, it symbolizes the order of the system, and, similarly to the enthalpy, it also frequently decreases during the protein folding stage as a result of the diminishing dynamics of the residues compared to the residue dynamics of the protein in the unfolded or denatured state. Although the official SI unit for the Gibbs free energy is *Joule*, biologists often substitute Joules for **calories** as a measurement unit.

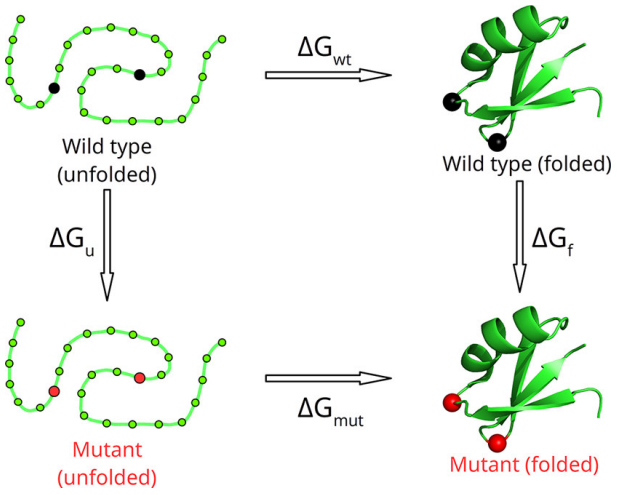


Figure 3.1: Image shows the visual representation of the thermodynamic cycle frequently used in calculations of $\Delta\Delta G$. The change ($\Delta\Delta G$) in Gibbs free energy between mutated protein and its wild-type counterpart is measured as the difference of changes in Gibbs free energies (ΔG) upon transition from unfolded to folded state (or vice versa) for each respective variation. Individual mutation zones are highlighted in **black** for wild-type protein and **red** for its mutant [51]. Image is taken from [50].

As hinted at previously, the more sophisticated and preferred method for displaying protein stability using Gibbs free energy in proteins is its difference (ΔG) between the protein in its folded and unfolded state.

$$\Delta G = G_{folded} - G_{unfolded} \tag{3.3}$$

For a multitude of reasons (protein improvement, disease research, etc.), some areas of research are trying to discover the effects of various amino acid mutations on wild-type proteins. As shown in Figure 3.1, for protein stability, the change ($\Delta\Delta G$) in Gibbs free energy is calculated between ΔG of a wild-type protein and ΔG of a mutant.

$$\Delta\Delta G = \Delta G_{wild-type} - \Delta G_{mutant} \tag{3.4}$$

In this case, the positive result would indicate that the mutation had a beneficial impact on the stability of the mutated protein compared to that of its wild-type counterpart. The calculation of $\Delta\Delta G$ is not standardized; therefore, whether the stabilizing mutation will yield a negative or positive $\Delta\Delta G$ value is entirely dependent on the positions of wild-type and mutant ΔG in the equation. The unit of measurement used most frequently for $\Delta\Delta G$ is $kcal.mol^{-1}$.

Melting temperature

Another common numerical measurement of the thermal stability of proteins is the melting temperature (T_m). It is defined as the temperature at which the free energy of the protein in its folded state equals the free energy of the unfolded state, while half of the protein is folded and the other half is unfolded.

$$\Delta G_{folding}(T_m) = 0 \tag{3.5}$$

Similarly to Gibbs free energy, measuring the effects of a specific mutation(s) on the stability of a protein would be to calculate the difference between its melting temperatures ΔT_m.

$$\Delta T_m = T_m^{mut} - T_m^{wild} \tag{3.6}$$

Although this might suggest that there is a strong correlation between experimental ΔT_m and $\Delta\Delta G$ at room temperature (T_r), the results show [66] that the correlation coefficient is about -0.7, which symbolizes imperfect correlation. In the case of highly destabilizing or highly stabilizing mutations, there are instances where ΔT_m and $\Delta\Delta G(T_r)$ do not correlate as well as in the case of neutral mutations or mutations with very limited effect, as shown in (shown in figure 3.2. This is because the transformation between these two descriptors is not linear; thus, there does not exist a simple equation for the estimation of $\Delta\Delta G$ based on ΔT_m and vice versa.

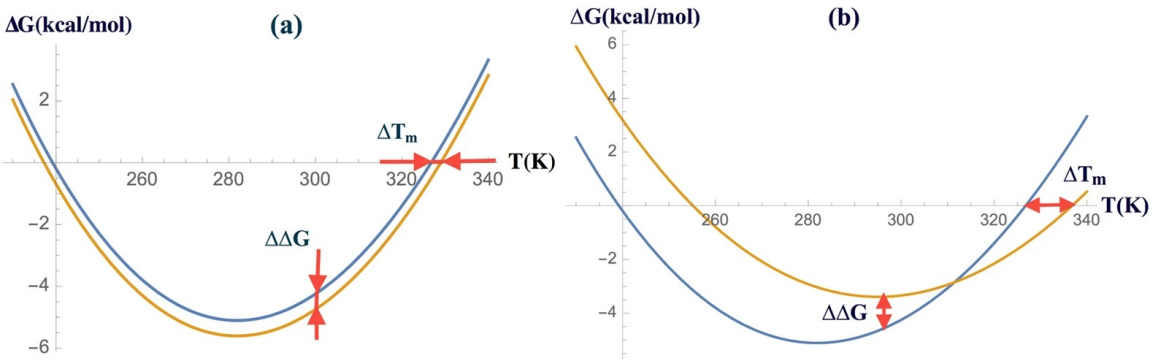


Figure 3.2: Picture shows the effects of mutations on the full stability curve of a hypothetical protein. The stability of the wild-type is represented by the blue line, while the orange color symbolizes the stability curve of a mutant. In the case of neutral mutations (**a**), or mutations with very limited effects, a negative correlation between ΔT_m and $\Delta\Delta G$ can be observed. While in the case of significant changes between mutant and wild-type (**b**), such relation is lost. Image is taken from [66].

3.2 Acquisition methods

There are two distinct approaches to obtaining the stability of the observed protein. The first option is to obtain measurements from machines commonly found in laboratory environments, called *in vitro*.

Laboratory measurements

When attempting to determine protein stability via laboratory means, scientists have numerous methods at their disposal. These measurement methods will differ not only in the setting of the experimental conditions (pH, temperature, etc.) but also in the type of results they provide. This implies that compared experimental results should ideally share the same experimental conditions and measurement method [50] for the comparison to hold any meaningful value.

Differential scanning calorimetry is one of the most widely distributed methods for studying the thermodynamic properties of protein unfolding, partially because it does not require any models of *a priori* assumptions. This thermoanalytical technique operates by measuring physical and chemical changes within a protein in response to temperature, determining precise thermodynamic properties such as melting temperature (T_m), the difference between the heat capacity of the protein in the unfolded state versus the protein in the folded state (ΔC_p), or change in calorimetric enthalpy (ΔH_{unf}^{cal}).

Circular Dichroism is an absorption spectroscopy method based on the differential absorption of left and right circularly polarized light by the sample protein. When circularly polarized light is used, the magnitude of linearly polarized light is kept, while the electric field vector rotates around its axis clockwise or counterclockwise. When the absorbance of the two different rotation directions is measured separately, both the absorption difference (ΔA) and the circular molar dichroism ($\Delta\epsilon$) can be calculated. These values then provide a lot of essential information about the structural nature of the observed protein, such as the secondary structure, the folding, and unfolding responses, or the determination of pH, solvent, and temperature-induced structural changes.

Absorption spectroscopy determines chemical elements of the sample using the absorption of electromagnet radiation as a function of frequency or wavelength with the help of free atoms in a gaseous state. It operates based on the Beer-Lambert law [16], which establishes the relation between the concentration of the sample analyzed currently and the standards of an already known analysis [7]. This technique is used in the field of analytical chemistry.

Recently, the discoveries of new experimental techniques and advances in the technology required for their realization have resulted in much cheaper and faster protein analysis through laboratory measurements. Although this progress successfully simplified some of the smaller problems in protein stability research, some other large-scale analyses, like determining the effects of every possible mutation in the protein of interest, are still way too inefficient if performed only with the use of laboratory equipment, hence the increasing demand for *in silico* approaches.

3.3 Predition methods

The second approach to obtaining protein stability estimates is to take advantage of statistics and already-established structural and evolutionary information about proteins. . Methods belonging to this category are called *in silico* and include force field calculations and machine learning, among others.

Force field calculations

The rapid evolution of computer hardware and recent breakthroughs in the algorithmic world, combined with the increased acceptance of *in silico* calculations in biotechnology-related fields, lifted the limitations posed by experimental methods and allowed scientists to dive into more complex research topics. One such technique is to calculate the difference in free energy (ΔG) between the native folded and unfolded states, which are devoid of any ordered conformational character, using existing force fields. Force fields are mathematical functions and parameter sets that are used to calculate the potential energy in atomic and molecular systems. The formula for the difference in free energies between two states ΔG is [62]:

$$\Delta G = G_{folded} - G_{unfolded}, \tag{3.7}$$

where G_{folded} represents the free energy of the folded protein, while $G_{unfolded}$ can be interpreted as a sum of forces present in its unfolded state.

The total free energy of the protein in a **unfolded state** is computed as:

$$G_{unfolded} = G_{en} + G_{ne}, \tag{3.8}$$

where G_{en} stands for the contribution of entropic free energy, while G_{ne} represents the contribution of nonentropic free energy. The free energy of the unfolded protein is directly related to the size of the protein, and it was estimated that the participation of each residue is approximately 1.2 $kcal.mol^{-1}$ [5].

To achieve a proper estimation of the force field of the protein in **folded state**, all atomic forces that influence its three-dimensional structure must be taken into account, resulting in formula [62]:

$$G_{folded} = G_{hy} + G_{el} + G_{hb} + G_{ss} + G_{vw}, \tag{3.9}$$

where G_{hy}, G_{el}, G_{hb}, G_{ss}, G_{vw}, represent, respectively, the free energy contributions of the hydrophobic, electrostatic, hydrogen bonding, disulfide, and Van der Waals forces.

The first element from the list of forces that affect the free energy of a folded protein, **hydrophobic energies** can be expressed from the accessibility of the solvent as

$$G_{hy} = \sum_i \Delta\sigma_i [A_i(folded) - A_i(unfolded)], \tag{3.10}$$

where i denotes a specific atom, $A_i(folded)$ represents solvent-accessible areas of each atom in a folded state, while $A_i(unfolded)$ represents solvent-accessible areas in an unfolded state of the protein. Lastly, $\Delta\sigma_i$ denotes the atomic solvation parameters.

One of the important sources of **electrostatic interactions** in protein chains is the charged R groups of lysine, histidine, arginine, and finally glutamic and aspartic acid, plus the neutral nitrogen and oxygen atoms present in the backbone. These atomic groups are known to be frequent participants in the establishment of hydrogen bonds, charge-charge or *ionic* pairs, and *charge-helix* interactions. With the exclusion of hydrogen bonds, due to their importance that deserves a separate term G_{hb}, the free energy of the remaining interactions can be computed as follows:

$$G_{el} = E_1 + E_2, \tag{3.11}$$

where E_1 represents the electrostatic energy coming from the ion pairs and E_2 represents the electrostatic energy from the charge-helix dipoles. It has been observed that the contribution of surface-located ionizing pairs is around 1 $kcal.mol^{-1}$ [4], while interior-located ionizing pairs contribute roughly three times more [26]. In the case of helix-dipole interactions, their contribution lies somewhere in the 1.6 $kca.mol^{-1}$ area [71]. Thus, the respective electrostatic free energies can be calculated as

$$E_1 = 3N_{bi} * 1N_{si}, \tag{3.12}$$

$$E_2 = 1.6N_{ch}, \tag{3.13}$$

where N_{bi} represents the number of pairs on the surface of the protein and N_{si} the number of pairs buried in a given protein, while N_{ch} represents the number of charge-helix dipole interactions, often derived from the crystal structure of the protein.

The contribution of free energy by **hydrogen bonds** to the overall stability of a protein is significant [55] because they are one of the main actors in the development of secondary structures. Their exact free energy is usually extracted from spatial information, but it is possible to approximate their relative contribution using formula [63]:

$$N_{HB} = 0.73N, \tag{3.14}$$

where N indicates the total number of residues present in a protein. This approximation yields the total free energy of all hydrogen bonds, including the ionic. The contribution of ionic bonds is already accommodated in electrostatic interaction free energy calculation, so the final formula subtracts them to avoid double counts:

$$G_{hb} = N_{HB} - (N_{bi} - N_{si}). \tag{3.15}$$

The site-directed mutagenesis experiments had shown that the free energy contribution of one **disulfide bridge** is around 2.3 $kcal.mol^{-1}$ [87], so the total contribution can be calculated accordingly:

$$G_{ss} = 2.3N_{ss} \tag{3.16}$$

where N_{ss} denotes the total number of disulfide bridges in the protein.

One of the possible approaches to calculating the free energy from **Van der Waals forces** is the Lennard-Jones formula [20]:

$$G_{vw} = \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \tag{3.17}$$

where $A_{ij} = \epsilon_{ij}^{*}(R_{ij}^{*})^{12}$, $B_{ij} = 2\epsilon_{ij}^{*}(R_{ij}^{*})^{6}$, $R_{ij}^{*} = (R_{j}^{*} + R_{i}^{*})$, $\epsilon_{ij}^{*} = \epsilon_{j}^{*}\epsilon_{i}^{*}$. The indices i and j represent individual atoms, R^{*} represents the van der Waals radius, and ϵ^{*} the depth of the van der Waals well.

Machine learning

Recent advances in machine learning-based approaches have been a sensation in almost every possible area of research, including protein engineering [72, 86]. Their convenience makes them worth considering for the prediction task of whether substitution of the wild-type residue for the mutant will have a positive or negative effect on overall protein stability and, in the case of more recent ML models, even on predicting the exact difference caused by the substitution. Another reason for the widespread adaptation of ML stems from its ability to create self-sufficient predictive models without the need to program and tweak every possible detail manually. These models can be separated into two distinct groups based on their learning approach: *supervised* and *unsupervised* models.

During the training process, supervised models use experimental protein stability measurements as labels to validate the accuracy of their predictions. This is an attractive feature because models that learn directly from experimental data are capable of making solid predictions on an absolute scale [8]. However, the main limitation of the applicability of supervised learning models to protein stability prediction is their vulnerability to systematic biases. These biases range from typical ML problems, such as model overfitting to training data, to task-specific problems, such as biases towards destabilizing mutations due to their overrepresentation in training datasets [25] or amino-acid residue substitutions that have been probed experimentally. This leads to a degree of self-inconsistency in the model predictions, easily observable when comparing typical wild-type to mutant predictions with their hypothetical reverse counterparts [75].

Unlike the supervised learning process, unsupervised models do not validate their predictions with experimental stability measurements during their training process. Instead, they take advantage of different learning techniques, such as feature-based data clustering or prediction of the correct amino acid residues from self-generated masked sequences or structure information. This allows them to gain additional valuable information about the data. For example, unsupervised models can learn the amino-acid likelihood distributions at particular positions in proteins, which can be directly used to predict the effects of mutations on protein stability while avoiding the caveats of supervised learning. Although growing in popularity, unsupervised models still require far more computational resources during training than their supervised alternatives. Furthermore, they are often limited to making predictions only on a relative scale, further limiting their current practical application [8].

To address the weaknesses of both learning approaches and tackle the challenges of unbalanced datasets, various methods have been developed. It became common practice to evaluate the performance of newly developed models with *k-fold cross-validation* [68], severely reducing the chance that overfitting is left unnoticed. If it is impossible to directly

balance the ratio between stabilizing and destabilizing mutations, methods such as data augmentation, data oversampling, or using weighted loss functions can be used to artificially modify the data to prevent models from preferring one class over another.

Chapter 4

Data

This chapter discusses various common sources utilized for obtaining validation data for proteins and their mutants, highlighting challenges and approaches for ensuring data quality. It also introduces the BenchStab dataset, developed to address biases in existing data sources.

4.1 Available sources

The main source for obtaining validation data for proteins and their mutants is the ProTherm database [53]. ProTherm is the largest publicly available database with thermodynamic parameters related to protein stability, such as $\Delta\Delta G$, ΔT, and ΔC_p. As of today, it has more than 32 000 records of single mutants ($\sim 51\%$) and multipoint ($\sim 11\%$) mutants with their respective wild-type counterparts ($\sim 38\%$) and still serves as one of the primary sources of protein stability datasets used for the development of new software prediction tools. However, the legitimacy of ProTherm is a topic of discussion, as multiple critical issues have been reported [47, 64], with some of them being high redundancy, missing values, and severe inconsistencies between data from different sources. Specifically, the most problematic is the reoccurring lack of uniformity between signs of $\Delta\Delta G$ values, notable differences in pH levels in environments where thermodynamic measurements were made, and redundant records with contrasting thermodynamic values [84]. This hinders the effectiveness of the database since the data acquired from ProTherm might require additional manual filtering to avoid potential contamination before they can be used.

This is exactly the case for the VariBench database [52]. For new experimental data to be accepted by Varibench, they must fulfill multiple quality-ensuring conditions. These include non-redundancy of the data, the inclusion of both positive and negative entries to prevent large-scale imbalance between classes, acceptance of only verified experiments as valid data sources, and overall relevance of the dataset in capturing the characteristics of the given problem. Currently, the VariBench database contains 30 separate single-variant mutation protein stability datasets, many of which are used as training datasets for protein stability prediction tools. The most popular of them is the S2648, also known as the PopMuSiC data set, which was included in training or validation in at least half of the software prediction tools mentioned in Chapter 5. Still, even with the constraints posed by the VariBench database, this data set falls into the category of unbalanced data sets, since the destabilizing mutations outnumber the stabilizing in an almost 4:1 ratio. Furthermore, in 755 of its 2648 mutations, $\Delta\Delta G$ values range only from -0.5 to 0.5. This is problematic,

since the average experimental error in the $\Delta\Delta G$ measurements was determined to be 0.48 $kcal.mol^{-1}$ [40], so these 755 entries cannot be convincingly marked as either destabilizing or stabilizing.

Another popular dataset used to develop and validate protein stability prediction tools is SSYM [65]. SSYM is a manually curated data set extracted from the ProTherm database and contains $\Delta\Delta G$ values that had their mutant and wild-type 3D protein structures solved by X-Ray crystallography. The data set includes a total of 684 mutations, half of them being direct mutations inserted into 15 wild-type protein structures, and the other half coming from inverse mutations inserted into 342 different mutant protein structures.

The final source of biological data with thermodynamical measurements mentioned in this chapter is FireProtDB [84]. FireprotDB data come from a variety of sources, such as the VariBench data set, the ProTherm, and ProtaBank databases, or by manual extraction from the scientific literature. To avoid the potential pitfalls plaguing ProThermDB and other sources, the authors of FireProtDB filtered all new potential records by following these steps:

- The entry does have to contain a single-point mutation
- Mutation in the entry is not an insertion or deletion
- Protein assigned to the mutation has both SwissProt accession code and also entry in PDB
- The entry includes measured values of $\Delta\Delta G$ or ΔT_m

This was then followed by validating the accurate mutation site coordinates in protein structures and, if there were any discrepancies, making necessary corrections to the mapping. This rigorous validation procedure greatly enhanced the overall accuracy of the database.

4.2 BenchStab dataset

The previous section demonstrates the challenge of identifying an appropriate benchmarking dataset. The reliability of some existing data sources is in doubt, leading to the frequent utilization of limited public benchmarking datasets to validate or train new stability prediction tools. Consequently, during benchmarking, these datasets exhibit bias as the models trained on them tend to perform better. To mitigate this bias towards specific predictors, the *BenchStab* dataset was developed.

The initial stage in creating the BenchStab dataset involved selecting an appropriate data source. FireProtDB was selected due to its appealing attributes, including manual curation and thorough filtering procedures. To utilize FireProtDB data to evaluate stability prediction tools without the biases mentioned above, a set of additional procedures was implemented, as illustrated in Figure 4.1. The information from FireProtDB was obtained in two streams: the filtered FireProtDB dataset, referred to as *pre-Benchstab*, and the combined dataset of all evaluated predictors. This combined dataset included the validation datasets of machine learning predictors, as well as the validation and training datasets of energy-based predictors. A comprehensive summary of the chosen datasets for each predictor is available in Appendix A.

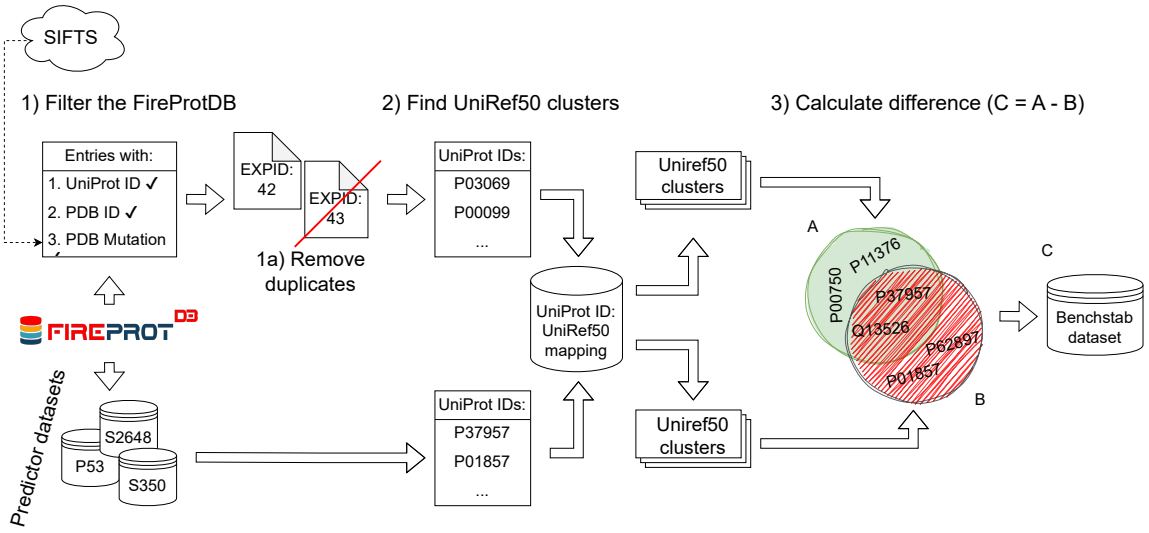


Figure 4.1: The pipeline for extracting data from the BenchStab dataset involves a series of steps that are applied to two distinct sets of data: the combined validation/training dataset from all available predictors and the filtered FireProtDB - pre-Benchstab. These steps consist of filtering the FireProtDB using a set of criteria, eliminating duplicate entries, associating UniRef50 [85] clusters with UniProt IDs, and excluding identical clusters found in both the pre-Benchstab and predictor datasets.

Although FireProtDB contains sequential and structural data, as well as $\Delta\Delta G$ and ΔT_m records, most existing online stability prediction tools only accept protein structures and predict $\Delta\Delta G$ values. To make the dataset more comprehensive for all predictors, additional criteria were introduced for FireProtDB:

- The entry must be assigned a Uniprot ID.
- The entry must be assigned a PDB ID.
- Protein sequences in the PDB are often subsets of those in Uniprot, leading to discrepancies in amino acid position mapping between the two databases. To address this issue, each selected entry must include the position of the wild-type amino acid to be mutated in both UniProt and PDB formats.The PDB position was sometimes also calculated based on the UniProt position, with offsets obtained from the SIFTS [21] API.
- Lastly, the entry must have experimentally measured $\Delta\Delta G$ value.

This filtering process decreased the initial **15989** entries to the **12034** records. Due to the presence of multiple experiments conducted for identical protein/chain/mutation combinations, duplications were eliminated, resulting in **5081** records. Occasionally, a single FireProtDB record was associated with multiple indistinguishable or nearly identical PDB structures. To eliminate redundant entries, the first PDB structure in the list was consistently opted for, aligning it with the reference structure displayed in the FireProtDB web interface. The filtering procedure was used exclusively on the pre-Benchstab dataset. When dealing with the merged predictor dataset, FireProtDB web interface filtering was used to choose the datasets of each predictor individually.

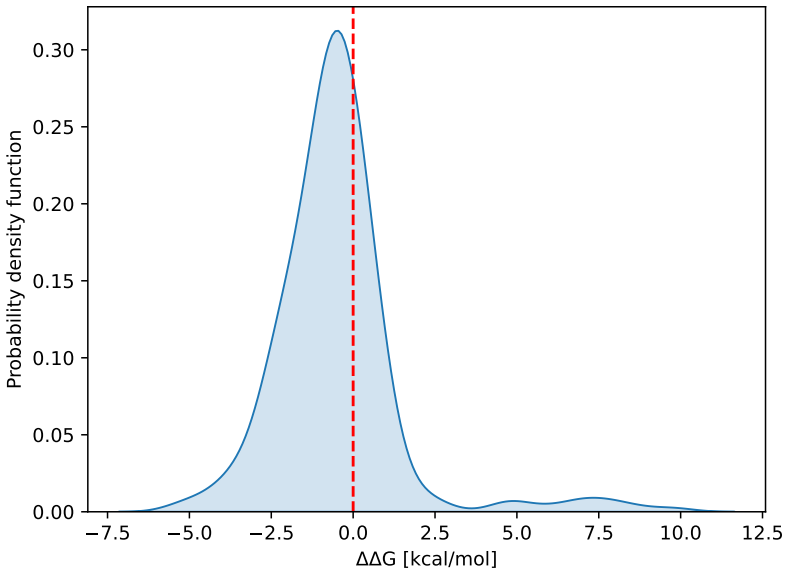


Figure 4.2: Kernel density estimate of experimentally determined $\Delta\Delta G$ of the BenchStab dataset (ground truth). Negative $\Delta\Delta G$ corresponds to destabilizing mutations.

The subsequent phase in the process aimed to reduce the risk of contamination in the Benstab dataset. In this context, contamination refers to the situation where some of the entries utilized for training or validation of the predictors inadvertently transfer to the benchmarking dataset, potentially introducing biases towards specific predictors. A straightforward strategy to address this issue would involve screening the pre-Benchstab data by eliminating entries with matching Uniprot IDs between the pre-Benchstab and the combined predictor dataset. While this method would address superficial contamination, it is worth noting that if a pre-Benchstab entry shares a high degree of sequence similarity with an entry in the merged predictor dataset, it may still be considered non-contaminating, as long as they do not have the same UniProt ID. But, even if there are changes in some amino acids within the entry, these alterations may not be sufficient to significantly impact its final structure or properties. Additionally, prediction tools gather a wealth of information about a sequence or structure beyond just its ID, including spatial details, sequence characteristics, and properties. So, despite differences in UniProt IDs, the predictors may already have encountered similar data points.

To prevent this issue, it was decided to retain only pre-Benchstab sequences that have less than 50% sequence identity[1] with sequences in the combined predictor dataset. One effective approach for accomplishing this was to cluster entries based on their sequential identity and compare their Cluster IDs instead of UniProt IDs. Fortunately, UniProt offers a well-maintained database of such clusters known as UniRef [85], eliminating the need for manual clustering. Consequently, each data point was mapped to its corresponding UniRef50 cluster using UniProt. Following the elimination of records with identical UniRef50 IDs, the resulting BenchStab dataset comprised **289** distinct entries. Figure 4.2 demonstrates that the distribution of $\Delta\Delta G$ in the BenchStab dataset leans towards negative values, indicating that when this technique is used on the FireProtDB, there is a slight bias towards mutations that destabilize the protein.

[1]The degree of similarity between two nucleotide sequences.

In order to validate the selected method and further investigate the distinctiveness of the dataset, the structural characteristics of the records were also compared. Due to the complexity and computational intensity of locally clustering the structures, the SCOP database [45] was utilized for this purpose. SCOP categorizes proteins into hierarchical groups based on their classes, folds, superfamilies, families, and so on. By associating these groups with the BenchStab dataset, it was revealed that the BenchStab dataset encompasses 8 distinct SCOP classes and 25 different SCOP folds, indicating a significant level of uniqueness. The comprehensive summary of the dataset is presented in the Table 4.1.

BenchStab dataset	
Mutations	289
Stabilizing mutations	80 (28%)
Proteins	36
UniRef50 clusters	35
SCOP classes	8
SCOP folds	25

Table 4.1: The statistics of the newly created BenchStab dataset.

Chapter 5

Predictors

This chapter is devoted to familiarizing the reader with the current and freely available stability prediction tools and their specific settings, capabilities, and drawbacks. Firstly, with regard to their accession method, the current pool of predictors could be separated between offline tools, and online tools, available for users through a public web interface. Since this thesis aims to develop a client tool for efficient stability predictions gathered from a wide range of predictors, the focus will be heavily shifted toward server-hosted predictors with public websites.

Another common categorization related to stability predictors is based on the computation method on which they rely:

- Predictors using free-energy calculations as their core.
- Predictors applying statistics and machine learning.
- Meta-predictors, which combine the previously mentioned approaches with the goal of achieving more precise and robust results.
- Predictors based on evolutionary information [69].

5.1 Energy calculations

Based on the level of detail in force-field modeling, energy-based software tools for stability predictions can be divided into three theoretical categories [36]. In the first approach, the results are derived from physical effective energy functions (PEEFs), which include simulations of physical interactions between individual atoms in tertiary structures. This method is extremely detailed and complex, so for every prediction, a substantial amount of computational time is required. A notable example of a software tool that belongs to this category is Eris [98], which in addition to the physical force field and atomic modeling includes fast side-chain packing and backbone relaxation algorithms. Another web-based software tool that belongs to this category is TSKA-MC [19], which, on the other hand, bases its predictions on charge-charge interactions. When the electrostatic free energy contribution of each ionizable residue in the protein is calculated using the Tanford-Kirkwood model [34], unfavorable residues in wild-type protein proteins can be discovered. These can then be used as possible targets for rational mutations.

The second theoretical group of energy-based predictors utilizes statistical energy functions (SEEF). Their advantage lies in the ability to describe complex interactions as entropic effects or many-body interactions, thus avoiding the difficulties of computationally

demanding processes of separating and quantifying. A typical example of SEEF-based online prediction tools is the widely known PoPMuSiC [23] and SDM [94]. The statistical energy function in PoPMuSiC is a result of a linear combination between thirteen biochemical and physical terms, the approximate values of which were obtained from databases of known protein structures [51]. A different approach was taken in SDM, where the force field equation uses environment-specific amino acid substitution frequencies within homologous protein families [56]. The full list of considered free energy predictors is in Table 5.1.

Method	**EEF**	**Input**	**Mutations**	**Output**
Sride [46]	Empirical	Structure	Single-Point	Binary decision
CUPSAT [57]	Empirical	Structure	Single-point	$\Delta\Delta G$
Eris [98]	Physical	Structure	Single/multi-point	$\Delta\Delta G$
SDM2 [94]	Statistical	Structure	Single-point	$\Delta\Delta G$
TSKA-MC [19]	Physical	Structure	All rational	$\Delta\Delta G$
DDGun [49]	Empirical	Sequence/structure	Single-point	$\Delta\Delta G$
PoPMuSiC [23]	Statistical	Structure	Single-point	$\Delta\Delta G$
pStab [31]	Empirical	Structure	All	Unfolding curves

Table 5.1: Online free-energy-based software tools for determining mutations' effects on a protein's overall stability. All tools are separated into three categories based on the nature of effective energy function (EEF) used in their calculation.

The last category of force-field-oriented prediction tools uses empirical effective energy functions (EEEF). The EEEF approach combines physical and statistical descriptions of possible interactions with experimentally determined empirical data [36] in its calculations, thus combining the SEEF and PEEF approaches. This provides a trade-off between the fast predictions achieved by the SEEF algorithms and the reliability of the PEEF-based tools, but it comes at the cost of a limited field of application due to the constraints placed by the environmental conditions under which the experimental data were acquired [51]. Some of the freely available predictors that employ this strategy are DDGun [49], Sride [46], CUPSAT [57], and pStab [31].

5.2 Machine learning

The increasing popularity of adapting ML algorithms to protein stability predictors can be attributed to their lower computational complexity compared to force field-based alternatives and, in some cases, the possibility of predicting stability changes solely from amino acid sequences [43, 12, 15]. Furthermore, machine learning predictors can rely exclusively on experimental training data without requiring any prior knowledge of the protein structure. However, the performance of these algorithms has been questioned multiple times [75, 25], where they failed to overcome the machine learning pitfalls mentioned in section 3.3. To alleviate the imprecision caused by unbalanced datasets, predictors, listed in Table 5.2, started using algorithms less susceptible to overtraining, precisely: Support Vector Machines, Random Forests, and Gradient-boosted Decision Trees.

Method	Model	Input	Mutations	Output
I-Mutant 2.0 [12]	SVM	Sequence/structure	Single-point	$\Delta\Delta G$
I-Mutant 3.0 [12]	SVM	Sequence/structure	Single-point	$\Delta\Delta G$
MUpro [15]	SVM	Sequence/structure	Single-point	$\Delta\Delta G$
mCSM [59]	Graph-based	Structure	Single-point	$\Delta\Delta G$
Auto-Mute [89]	SVM/RF	Structure	Single-point	$\Delta\Delta G$
INPS [74]	SVM	Sequence/structure	Single-point	$\Delta\Delta G$
PremPS [14]	Random Forest	Structure	Single-point	$\Delta\Delta G$
DeepDDG [11]	Neural Net	Structure	Single-point	$\Delta\Delta G$
NeEMO [30]	Graph-based	Structure	Single-point	$\Delta\Delta G$
SAAFEC-seq [43]	Decision tree	Sequence	Single-point	$\Delta\Delta G$
EASE-MM [27]	SVM	Sequence/structure	Single-point	$\Delta\Delta G$
PON-Tstab [97]	Random Forest	Sequence	Single-point	$\Delta\Delta G$
DDMut [99]	Neural Net	Structure	Single-point	$\Delta\Delta G$

Table 5.2: Online software tools utilizing machine learning algorithms to determine the effects of mutations on the overall stability of a protein.

The applicability of deep learning to obtain accurate stability predictions from protein structures and sequences has also been successfully tested [11, 8, 42], but overfitting still continues to pose a major problem when trying to develop generalizing deep neural networks [51]. This is caused by the increased complexity of neural nets in comparison to simpler ML algorithms, which allows them to learn very rare dependencies specific only to the training data. To solve the generalization problems, many techniques such as unit prunning [37], regularization, data-sampling, or unsupervised techniques (see section 3.3).

Another novel method used to obtain an authentic estimate of protein stability takes advantage of graph theory by representing protein structures as connected graphs with individual molecules as a *graph-based signatures* (e.g., NeEMO [30], mCSM [59]).

5.3 Meta predictors

The main idea behind meta-predictors is to combine multiple different prediction methods This allows for different views on an observed protein (e.g., statistical and structural), producing more precise and error-prone predictions. By combining the different complementary approaches to the prediction of protein stability, it is also possible to eliminate their drawbacks that appear when applied in isolation [8, 10]. This is the case for DUET [58], MAESTRO [41], iStable [13], and ELASPIC [93], which combine various machine learning algorithms with statistical energy-based functions to achieve the best possible predictions. Meta-predictors can be further categorized into two subgroups: those that independently compute results and ones that utilize other predictors.

The primary predictors in the initial category include ELASPIC, MAESTRO, and ENCoM. MAESTRO integrates the statistical energy function with various machine learning techniques such as neural networks, multiple linear regressions, and SVMs, using a multi-agent approach with a confidence score for each technique. In contrast, ELASPIC utilizes SVMs, Hidden Markov Models, and energy-based computations to forecast $\Delta\Delta G$ values. The predictor mentioned last, ENCom, makes use of normal mode analysis [28].

Predictors falling into the second category, which rely on other predictors for generating outcomes, are commonly referred to as **consensus predictors**. Examples of consensus

stability predictors, as shown in Table 5.3, include DUET and iStable. DUET integrates data from graph-based mCSM [59] and energy-based SDM [94] to form a final prediction using an optimized version of SVM. On the other hand, iStable aggregates individual predictions from five distinct prediction web tools: SVM-based iMutant2 [12] and MUpro [15], Random Forest and SVM variants of Auto-Mute [89], energy-centric PoPMuSiC [23], and CUPSAT [57] that utilizes environment-specific atomic potentials and torsion angles for prediction. Unlike a majority vote approach, iStable employs the Support Vector Machine algorithm to combine the individual predictions, resulting in more precise outcomes.

Method	**Input**	**Mutations**	**Output**
DynaMut2 [70]	Structure	Single/multi-point	$\Delta\Delta G$
DUET [58]	Structure	Single-point	$\Delta\Delta G$
MAESTRO [41]	Sequence/structure	Single/multi-point	$\Delta\Delta G$
iStable [13]	Sequence/structure	Single-point	$\Delta\Delta G$/Binary decision
ELASPIC [93]	Structure	Single-point	$\Delta\Delta G$
ENCoM [29]	Structure	Single-point	$\Delta\Delta G$

Table 5.3: Web-based software tools combining multiple protein stability prediction approaches used to determine the effects of mutations on the overall stability of a protein.

While MAESTRO, DUET, and iStable focus on combining the advantages of machine learning and free energy calculations, DynaMut2 [70] emphasizes the extraction of structural features. For this task, it employs graph-based signature extraction [59], normal mode analysis, wild-type residue environment, substitution propensities, contact potential scores, and interatomic interactions [38]. These features are then used to train the Random Forest ML algorithm, which produces the final prediction scores.

5.4 Performance metrics

Performance metrics are crucial in protein stability predictors, as they provide quantitative measures to assess the accuracy and reliability of predictions. These metrics provide insight into the effectiveness of predictive models in capturing complex relationships between protein sequence, structure, and stability. By evaluating performance metrics such as accuracy, precision, recall, mean absolute error, Pearson's correlation coefficient, and the area under the receiver operating characteristic curve (AUC-ROC), researchers can gauge the predictive power and robustness of algorithms.

Depending on the specific goals and requirements of the protein stability prediction task, metrics can be divided into two types: discrete and continuous. Continuous or regressive metrics focus on the accuracy of $\Delta\Delta G$ predictions as continuous values, while classification metrics evaluate the performance of stability prediction models in correctly categorizing mutations into stability classes.

Discrete metrics

Discrete metrics, also known as categorical or classification metrics, provide valuable insights into different aspects of the model's performance and help in assessing its effectiveness in classifying instances into the correct categories. In the realm of protein stability prediction, these categories are linked to the consequences of introducing mutations into proteins,

which could be stabilizing, destabilizing, or neutral mutations. To make things easier to understand, assume that the prediction tool only determines if the mutation results in stabilizing or destabilizing effects. Table 5.4 shows four potential outcomes of the prediction:

Name	**Explanation**
True Positive (TP)	Mutation was correctly predicted as stabilizing.
True Negative (TN)	Mutation was correctly predicted as destabilizing.
False Positive (FP)	Mutation was incorrectly predicted as stabilizing.
False Negative (FN)	Mutation was incorrectly predicted as destabilizing.

Table 5.4: Explanation of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) in the context of protein stability prediction.

The most commonly used metrics for the evaluation of the performance of predictive models are: accuracy (**Acc**), precision (**P**), recall (**R**), specificity (**S**), and false positive rate (**FPR**).

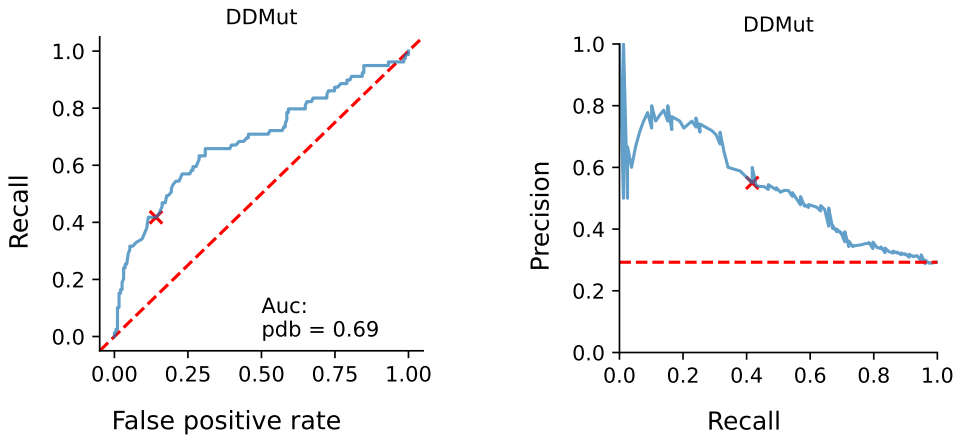
$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \qquad P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \qquad S = \frac{TN}{TN + FN} \tag{5.2}$$

where **Acc** denotes the success rate of the predictions. A bit more specific metric, **P** determines the proportion of correctly predicted stabilizing mutants out of all possible predictions. High precision indicates that when the predictor predicts a stabilizing mutation, it is likely to be correct. Although related, **R** assesses the model's capability to correctly identify stabilizing mutations. So, the primary distinction between precision and recall lies in the emphasis of precision on the accuracy of positive predictions, whereas recall evaluates the thoroughness of these predictions. Thus, a high recall means that the predictor identifies a significant fraction of positive cases. In contrast to recall, **S** defines the proportion of correctly predicted destabilizing mutations of all predicted destabilizing mutations. This means that a predictor with high specificity can avoid misclassifying destabilizing mutants as stabilizing. Lastly, the **FPR** is the proportion of true destabilizing mutations incorrectly identified as stabilizing by the predictor.

When working with imbalanced datasets, the use of accuracy might be misleading because it does not properly account for the class distribution. This poses a huge problem in protein stability prediction, because, as discussed in Chapter 4, most commonly used datasets are dominated by destabilizing mutations. Using accuracy with imbalanced protein stability datasets means that the predictor might achieve a high score simply by predicting the majority (destabilizing) class for all predictions and completely ignoring the minority (stabilizing) class. Another problem is that the goal of researchers is often to find stabilizing mutations, as these are often more helpful in strengthening protein properties. Again, in that case, high accuracy might give a false impression of predictor effectiveness, since the stabilizing mutations are underrepresented in the dataset. Possible metric without the problem are balanced accuracy (**BA**) and Matthews Correlation Coefficient (**MCC**):

$$BA = \frac{Recall + Specificity}{2} \qquad MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5.3}$$



(a) **ROC** (b) **PR**

Figure 5.1: **A.** Receiver operating characteristic curve depicts the relationship between false positive rate and true positive rate (sensitivity/recall) at different thresholds to classify mutations as stabilizing or destabilizing. **B**. Precision-recall curve depicting the relationship between precision and recall at different thresholds to classify mutations as stabilizing or destabilizing. The red dashed line represents a random predictor, while the red cross is the default threshold for binary classification (0 $kcal/mol$.

BA is defined as the arithmetic mean between recall and specificity, highlighting the equal significance of both minority and majority classes in assessments. Additionally, **MCC** is another valuable metric for assessing imbalanced datasets. MCC evaluates the classification effectiveness by taking into account both the model's specificity and recall.

Although most of the tools mentioned in this chapter predict continuous $\Delta\Delta G$ values, it is still possible to examine them using classification metrics. By empirically selecting the $\Delta\Delta G$ thresholds, continuous values can be binarized. However, this could lead to a potential classification bias.

This thresholding problem can be addressed using the receiver operating characteristic curve (**ROC**) or the precision-recall curve (**PR**), depending on the particular scenario. The ROC curve (Figrue 5.1a) is a curve that demonstrates the predictor's ability to classify by plotting the recall against the false-positive rate at various thresholds. Specifically, to predict protein stability, it indicates the effectiveness of the predictor in differentiating between stable and unstable mutations at different threshold levels. The performance of the predictor is better the closer the curve approaches the upper-left corner of the graph (high recall and low false-positive rate).

Like the ROC, the PR curve (Figure 5.1b) illustrates the interaction between two metrics at various thresholds. However, the PR curve specifically targets the classification of stabilizing mutations, assessing the balance between precision and recall. The closer the PR curve approaches the upper right corner, indicating higher precision and recall, the better the effectiveness of the predictor to minimize false positives and accurately identify stable mutations.

Continuous metrics

Continuous metrics for predicting protein stability present a quantitative method for evaluating protein stability. In contrast to binary classification metrics, which categorize mutations as stable or unstable, continuous metrics assign a numerical value that reflects the degree of change in stability ($\Delta\Delta G$). The most commonly used methods are mean absolute error (**MAE**) and root mean squared error (**RMSE**). They are calculated as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i| \qquad \text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{5.4}$$

where n denotes the number of predictions, y_i the experimentally measured $\Delta\Delta G$ and x_i the predicted $\Delta\Delta G$. The primary distinction between these metrics lies in their methods of calculation: **MAE** calculates the absolute mean deviation, while **RMSE** calculates root mean squared deviation between experimental and predicted $\Delta\Delta G$. The advantage of MAE is its straightforward interpretability, while RMSE is advantageous, as it imposes greater penalties on larger errors, thus heavily penalizing outliers compared to MAE. Some of the other insightful continuous metrics are:

$$PCC = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \qquad R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - x_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{5.5}$$

$$MSD = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i) \tag{5.6}$$

where for PCC, $\bar{x}$ and $\bar{y}$ represent the mean values for their respective categories. The mean signed deviation (**MSD**) provides insight into the bias of the predictors: a positive value indicates that the model tends to overestimate, while a negative value indicates an underestimate of the predicted $\Delta\Delta G$. On the other hand, the Pearson correlation coefficient (**PCC**) quantifies the linear relationship between predicted $\Delta\Delta G$ and experimental $\Delta\Delta G$. The PCC values range from $[-1, 1]$, where 1 indicates a strong positive correlation, and -1 a strong negative correlation. For protein stability predictions, a moderate to strong negative correlation often indicates a sign discrepancy between the predictors' output and the experimental $\Delta\Delta G$. An example of this scenario would be: the experimental $\Delta\Delta G < 0$ is considered destabilizing, while the predictor would consider $\Delta\Delta G < 0$ as stabilizing.

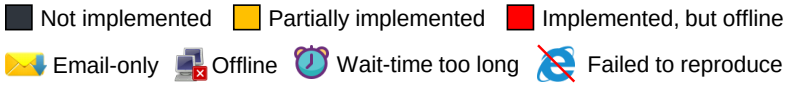
Given that the categories are inverted, the Pearson Correlation Coefficient (PCC) would demonstrate a negative correlation between the two methodologies in this scenario. Lastly, the coefficient of determination ($\mathbf{R^2}$) quantifies the variance proportion between the predicted $\Delta\Delta G$ and the experimentally derived $\Delta\Delta G$. The range of values of R^2 is $(-\infty, 1]$, where $R^2 = 1$ means an ideal prediction, while negative values suggest that the prediction model is less accurate than a simple horizontal line.

Chapter 6

Application design

This chapter aims to clarify the design decisions taken during the development of the predictor multi-client tool, called BenchStab. The purpose of BenchStab is to obtain $\Delta\Delta G$ predictions from various prediction tools accessible on the Internet in an efficient and time-saving manner.

Table 6.1 shows all predictors that were considered for implementation. Due to several reasons, including predictor unavailability, email-only results, and infinite wait times, only 18/29 predictors were included in the BenchStab.



Free energy	Machine learning		Meta approach
SRide	I-Mutant 2.0	STRUM	iStable
CUPSAT	I-Mutant 3.0	PON-tstab	DUET
Eris	MUPro	DeepDDG	ELASPIC
SDM2	mCSM	DynaMut2	Maestro
TKSA-MC	NeEMO	PremPS	DynaMut
pSTAB	Auto-Mute 2.0	SAAFEC-seq	ENCoM
DDGun	INPS-MD	DDMut	
PoPMuSiCsym	EASE-MM		

Figure 6.1: Full list of considered predictors. The **black** color represents manually tested predictors, **yellow** color predictors which I attempted but failed to implement, and **red** predictors implemented, but currently offline.

A prominent example of an excluded predictor is TKSA-MC. which had inaccessible predictor host servers, resulting in all HTTP requests sent automatically failing. Additionally, NeEMO serves as another example in which, although BenchStab could successfully request a prediction from the web server, the web server failed to embed these predictions into its results table, and ended up sending an empty response. Lastly, DeepDDG was

initially set up but became outdated when the web server unexpectedly switched to an email-only system.

Initially, the Maestro meta-predictor was considered inappropriate for programmatic querying because rapid, multiple requests occasionally led to web server failures, typically followed by extended periods of inactivity. To quantify, Maestro disconnected from BenchStab and subsequently from a user via the web interface three times, with an average of 25 requests sent within 1-2 hours. Ultimately, imposing a stringent limit of only one concurrent request enabled its integration into BenchStab.

Unfortunately, none of the included predictors, except DynaMut2 and DDMut, offered an Application Programming Interface (**API**) for easy programmatic access. So to include them in the client application, the semi-unique algorithm had to be developed for each predictor by following these steps:

1. Manually request multiple predictions through the web interface.
2. Wait for the results and capture all communication over the network performed by the browser.
3. Analyze the captured communication between the browser and the web server. Look for the `POST` requests that transfer payload containing data, such as sequences, protein structure files, PDB IDs or mutation coordinates, and reoccurring `GET` requests that periodically check the prediction status.
4. Programmatically replicate and parse the bare minimum of requests needed to retrieve and extract prediction results.

6.1 Design

In most cases, acquiring prediction results from a prediction web server meant programmatically simulating HTTP communication that is usually hidden behind the user interface (**UI**) in the browser. Since there are no strict design guidelines for the UI of protein stability predictors, each predictor possessed a certain amount of uniqueness when it came to its HTML layouts and the exact HTTP communication flow. Their HTML layouts usually varied in arrangement and types of HTML elements, making it impossible to construct a singular parsing algorithm. In terms of HTTP communication flows, each predictor uses different types and amounts of HTTP requests with custom payloads, adding another level of complexity to the problem at hand.

Differences aside, since every single one of the included web tools shares the same functionality, which is to predict the change of stability upon mutation, they will naturally share at least some degree of similarity amongst each other. The developed multi-client application takes advantage of such similarities and groups them. These feature groups are then used to construct a hierarchical model, which can be seen in Figure 6.2. As it includes all characteristics, from those shared among all predictors to those unique, it was used as a template to build a class inheritance diagram shown in Figure 7.2.

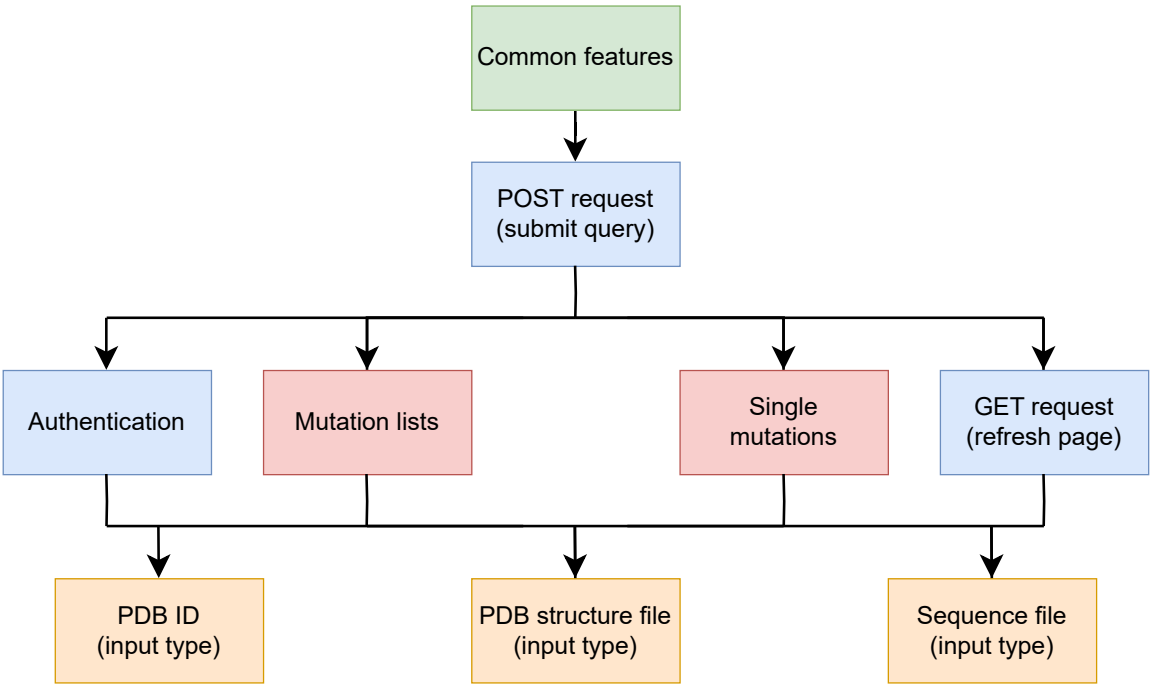


Figure 6.2: Vertical diagram displaying the hierarchy of shared characteristics found in the implemented predictors. The individual levels are based on the number of predictors sharing a common feature. Blue-colored features represent HTTP communication, orange-colored features the different variations of wild-type protein representations sent as a mutation source, while red-colored components divide predictors into two categories depending on the number of mutations that can fit in a single `POST` payload.

Based on the types of HTTP requests present in the communication captured between prediction servers and their web interfaces, the hierarchical model had shown that all included stability prediction tools could be separated into two categories: `POST`-only predictors and `POST`+`GET` predictors. Interestingly enough, there exists a strong correlation between these categories and the prediction speeds of individual software tools. In general, `POST`-only predictors employ faster prediction methods (e.g., machine learning), so they can deliver the results right in response to the query. On the other hand, `POST`+`GET` predictors, or just `GET` predictors, require more time to compute their results, thus queuing the prediction as a job via `POST` request and then periodically checking its status with `GET` request. A more detailed view of how exactly the entire request dynamic is handled inside the client application is provided in a subcategory of this section, called *HTTP communication*.

All of the modalities mentioned in this chapter led to a decision to create a custom predictor client for every web server/protein format combination that can be considered valid. In addition, their final output and accepted input formats had to be standardized to accommodate all needs and allow comparison and concurrent use of such predictor clients.

After careful selection of the required and optional parameters, the final uniform input and output formats were determined as:

.

Inputs

WT Mutated protein (= wild-type, structure or sequence)

MUT Mutation defined by wild-type amino acid, position, and mutant

CH Protein chain that will be mutated (structure only)

OPT Optional parameters (temperature, pH)

Outputs

INP Input parameters (= WT, MUT, CH, OPT)

DDG $\Delta\Delta G$ prediction

PN Name of the software tool that produced the prediction.

PT Type of the software tool that produced the prediction. (= sequence, structure, RCSB entry)

All of the predictor clients are part of one multi-client, which provides data processing capabilities, mapping of inputs to the specific predictors based on the input type and user preferences, and provides overall management for these tools. The multi-client can be accessed either as a standalone Python library or via a command-line interface.

6.2 Data processing

One of the marginal differences between web tools for predicting mutation stability is the biological data source based on which they infer their predictions, with some of them accepting only amino acid sequences in FASTA [44] format (e.g., SAAFEC), some only record IDs from the PDB [95] (e.g., PoPMuSiC), while some offer the option of uploading custom PDB files (e.g., SDM). This implies that the client application should not be limited to only one data format. For example, if the user wants to gather predictions of protein stability changes upon mutation from sequence-only software tools, He should be allowed to provide only amino-acid sequences in FASTA format, without needing to restrict himself to records that are available in PDB or have their structure already discovered.

Type	Format	Accessible data	Predictor type
Protein structure	PDB ID	Sequence file	Sequence
		Raw sequence	Sequence
		UniProt entry	Sequence
		Structure file	PdbFile
		RCSB entry	PdbID
	PDB File	Structure file	PdbFile
		Sequence file	Sequence
		Raw sequence	Sequence
Protein sequence	UniProt ID	UniProt entry	Sequence
		Sequence file	Sequence
		Raw sequence	Sequence
	Sequence file	Sequence file	Sequence
		Raw sequence	Sequence
	Raw sequence	Raw sequence	Sequence

Table 6.1: All of the data-type combinations that the BenchStab is capable of inferring from a given input format. Each predictor client has a unique data type assigned, and so can only accept data of such type.

Furthermore, prediction servers that support uploading files almost exclusively transfer prediction requests in FormData[1] format instead of the more commonly used `JSON` payloads. This adds another level of complexity that the application must address. In terms of data transformation, FormData supports two types of file formats defined by their content-type attribute:

- `text/plain`, which is a simple text representation, usually preferred in smaller files.
- `octet-stream`, usually the choice for larger files, encoding, and sending the files in binary form.

Due to these factors and the indirect benefits of improved readability and faster search capabilities, the client application employs the classes `Fasta` and `PDB`, which are subclasses of the `File` superclass. These classes primarily serve as wrappers for data formats of the same name, offering functions for data loading, parsing, and transformation to align with the specific requirements of the FormData format. Moreover, preprocessing facilitates the cascading acquisition of data, enabling the extraction of lower-level protein formats (sequence) from higher-level formats (structure). A complete enumeration of these extractions can be found in Table 6.1.

Some predictors also allow their $\Delta\Delta G$ predictions to be computed at different pH levels and temperatures; therefore, the application should also provide users with the option to define these environment-specific parameters for every mutation record. If these environmental parameters are not defined but required by the predictor, the neutral pH level and the room temperature are supplied as default values.

[1] https://developer.mozilla.org/en-US/docs/Web/API/FormData

Another data-related point of divergence among predictors is the number of single-point mutations that they are capable of processing in a single request. To separate them into categories, there are:

- Tools limited to **one single-point mutation** in their payload.
- Tools accepting **mutation lists** with `N` mutants of the same protein **chain**.
- Tools accepting **mutation lists** with `N` mutants of the same **protein**.
- Tools accepting **mutation lists** with `N` mutants of **different proteins**.

BenchStab takes advantage of this by arranging mutants into groups according to the payloads received by the predictors, decreasing the total number of separate requests to be sent over to the web server.

6.3 HTTP communication

I have observed and analyzed all instances of the HTTP communication carried out by all implemented predictors. Except for DynaMut2 and DDMut, it can be summarized into three steps:

1. The user fills in the form with the mutation site, mutant amino acid, wild-type protein, and some optional parameters, such as protein chain, temperature, or pH levels.
2. The fields in the completed form are then converted into a payload of a `POST` request and sent to the stability prediction server.
3. In case of `POST`-only predictors, the prediction server then either fetches the pre-computed stability change from the storage (database, cache, etc.) or calculates it immediately and returns the predicted results in a response to the `POST` request. As stated above, `GET` predictors require more time for their computations, so each requested query is queued as a job. While the job is in the queue or is being executed, the user is presented with a waiting screen that is refreshed periodically. Once the server calculates the final result, periodic refresh is stopped, and the user is presented with the prediction results.

What makes DynaMut2 and DDMut special is the presence of API, removing any unnecessary overhead caused by extensive HTML parsing, authentication, cookies, and header management. When querying these predictors, BenchStab submits a simple `POST` request with queried mutations and protein to the API endpoint, which then responds with the assigned job identifier. After submitting successfully, BenchStab periodically accesses the API endpoint via a `GET` request, monitoring the status of the submitted job. Whenever the job is completed, the API endpoint will respond with a list of predictions in an easily parseable `JSON` formatted response.

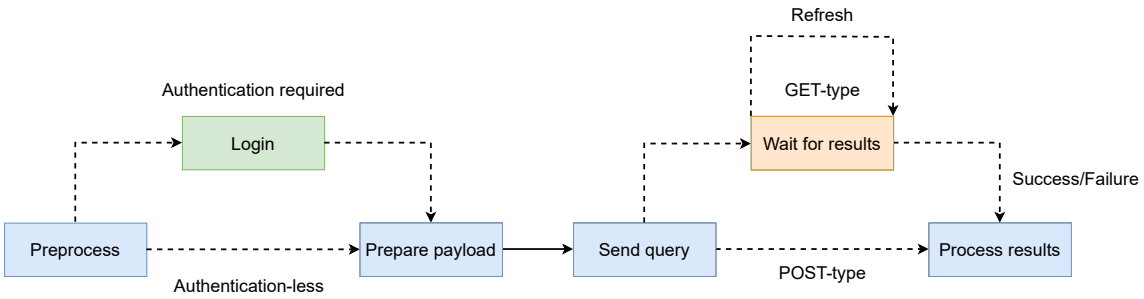


Figure 6.3: Data flow diagram showing the life cycle of single asynchronous prediction query. At first, the prediction query is preprocessed and, if the web server supports multiple mutations, they are merged into one list. The second step consists of the initiation of an HTTP session, optional authentication, and transformation of a query into a payload accepted by the stability prediction server. Lastly, in the case of `POST+GET` type of predictor, the client waits for the results in an asynchronous loop. Once the prediction is finished the obtained results are then extracted from the HTML response and forwarded along with additional information to the multi-client.

Initially, the implementation of the general predictor client followed the logic of a single HTTP session with one `POST` request, an optional `GET` waiting loop, and authentication (Figure 6.3). Each had an assigned callback function that processed HTML responses. This led to many failures, mainly because many predictors utilized session-specific identifiers, such as cookies and custom headers, causing them to constantly overwrite each other. To prevent this write-after-read hazard, instead of keeping a single session for all requests, a new instance of an HTTP session was created for every wild-type protein and mutation/s combination. Although this proved to be an ideal option in most cases, it still did not fit the needs of some more complex predictors, such as INPS, PremPS, or PoPMuSiC. As each of these web-based prediction tools required multiple requests to authenticate, submit prediction, or retrieve results, they did not conform to the original single or double request cycle depicted in Figure 6.3. To include them, the callback parameters were expanded by adding `session`, which together with the added definition of custom callbacks for HTTP requests allowed the chaining of requests. This meant that the user would be allowed to call new `POST` or `GET` requests from a callback, while also providing a new custom callback for those requests at the same time, thus creating a request chain. Figure 6.4 shows an example application of request chaining used in INPS, PremPS, and PoPMusic predictors.

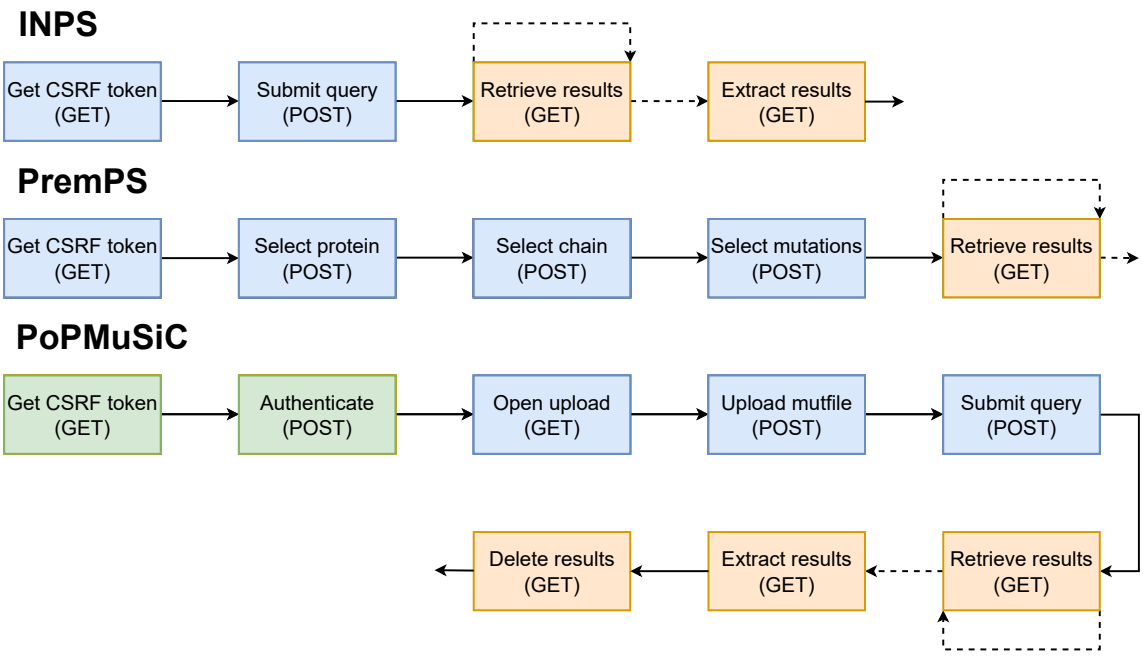


Figure 6.4: Data-flow diagram showing the minimum amount of requests needed for submitting a query to INPS, PremPS, or PoPMuSiC web server, and subsequently extracting the prediction results. Green cells represent the authentication process, blue cells procedure of submitting a single query, and orange blocks the retrieval of the prediction results.

Another of BenchStabs' distinctive features is its asynchronous approach to HTTP requests and predictor clients themselves. This is done by taking advantage of Python tasks and coroutines [90]. First, it is applied on the level of predictor clients to ensure their pseudoparallel execution, and then on the level of HTTP sessions themselves, avoiding any potential blocking from the side of HTTP requests. If BenchStab was fully sequential, the amount of time wasted while waiting for requests would equal most of its run time, and it would also mean that each predictor would have to wait for the previous one to finish, causing a further slowdown. Figure 6.5 provides a high-level view of individual components of a multi-client, with only the sequential part being the data processing block.

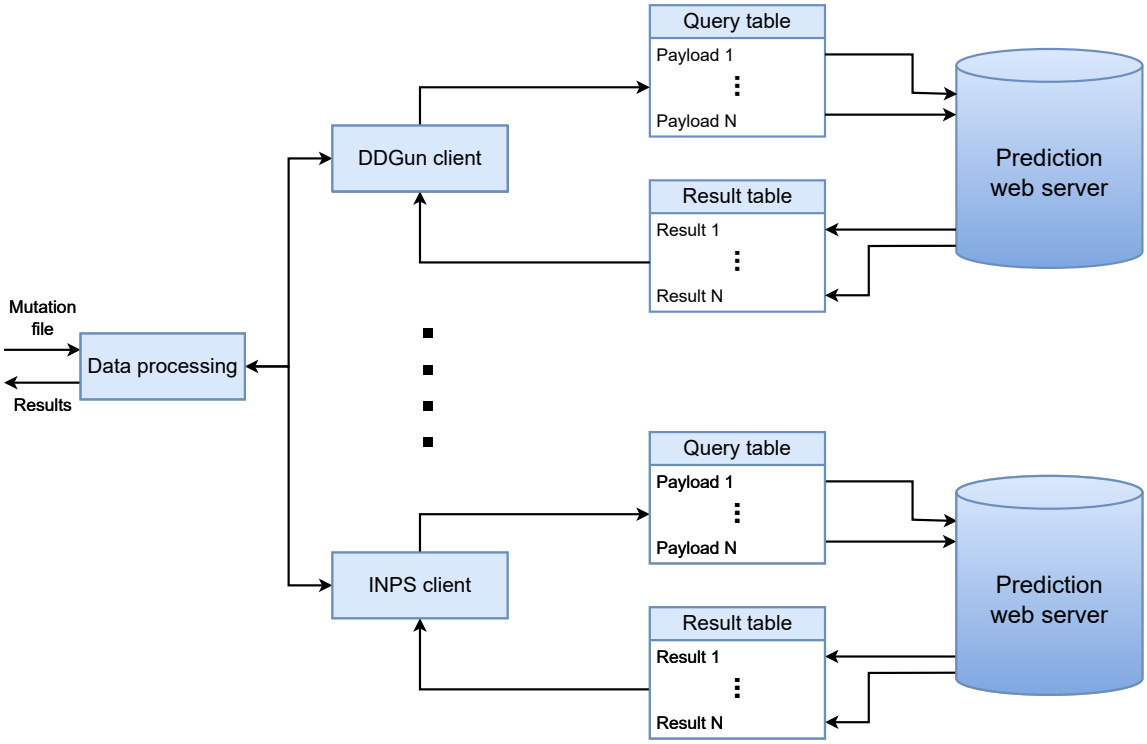


Figure 6.5: High-level diagram depicting the functionality of the BenchStab, including request-response dynamic between the protein stability predictor clients and their respective servers. At first, data are processed, then assigned to the appropriate `Predictor` clients, which then handle the communication with prediction web servers via asynchronous `GET` and `POST` requests in a single session. After collecting all responses from the server, the predictor client pieces the individual results together, assigns appropriate flags, and sends the data back to BenchStab. BenchStab then waits for all clients to finish their operations, concatenates their results, and presents the final predictions to the user.

Sometimes, if a large enough input dataset was provided, the sheer amount of asynchronous requests sent to the prediction web server at once could count as an unintended denial-of-service attack, which might have ended up causing the server to malfunction. As this may also have happened with Maestro, the feature called *batch size* was added, allowing the user to limit the maximum size of asynchronous requests sent simultaneously. Lastly, an attempt was made to calculate the appropriate waiting time between two status checks in `GET` predictors through geometric progression, but long pauses sometimes caused the user to log out, losing access to the prediction results; therefore, this feature was later discarded.

Chapter 7

Implementation

BenchStab is openly accessible on GitHub [6] and is developed using Python. It operates both as an independent Python library `BenchStab` and as a CLI tool. Additionally, it includes Python documentation [1] and a suite of 61 unique test cases. For asynchronous HTTP interactions, it employs `aiohttp`, uses `asyncio` for asynchronous handling of prediction clients, and utilizes the `pandas` library for data handling. BenchStab's architecture is structured into three main components: the primary loop, data preprocessing, and the management of prediction clients.

Main loop

The primary function of `BenchStab`, besides serving as a liaison between users and predictors, is to accurately determine the types of predictors from the data and then supervise their asynchronous operations. The mapping process is outlined in three stages:

1. `BenchStab` collects all versions of predictor client classes from the `predictors.web` module and organizes them into three groups according to the data type they support. The input data type for each predictor is deduced from the suffix of the class name. For example, for DDGun, the appropriate versions include `DDGunSequence`, `DDGunPdbFile`, and `DDGunPdbID`.

2. Should the parameters contain either an exclude or include filter, the function `filter_predictors` utilizes case-insensitive matching, disregarding suffixes, to either exclude or keep the specified predictors. When both types of filters are present, preference is given to the **include** filter.

3. Finally, every data point receives a specific set of predictors, constrained by its data type and the `allow` parameters. This ensures that errors such as assigning structure predictors to `Fasta` data points are avoided, even when `allow_struct_predictors` is enabled.

The main loop of BenchStab operates asynchronously, governed solely by the `asyncio` library. Each job submission triggers `compute`, which is then processed concurrently through `asyncio.gather` without blocking. This function, `gather`, handles execution until all tasks are completed, at which point control is transferred back to `BenchStab`. BenchStab then

[1] https://loschmidt.chemi.muni.cz/benchstab/details.html

processes and merges the results into the final data frame. Furthermore, during the operation of the predictor clients, after a specified duration (determined by `wait_interval`), all outcomes of the predictors are automatically captured and stored in the result file. This action protects against the potential loss of data due to an early termination of the process.

Parametrization

In addition to programmatic access, the BenchStab can also be accessed through the command line interface (CLI) `benchstab`. The script acts as a simple wrapper around the `BenchStab` class, providing the same level of parameter customization as the class. The usage hint can be accessed by calling the script together with the `--help` parameter, while the full list of supported predictors can be displayed by providing the `--list_predictors` standalone parameter. Both the CLI and the `BenchStab` library allow the user to customize these options (and many more):

- Specify the mutation file. (required)
- Provide a list of predictors to include/exclude. (optional)
- Provide a configuration file in the format `.json`, allowing the user to customize the global and per-predictor configurations. (optional)
- Specify the input data types of the included predictors. (required)

`config` parameter enables users to specify a custom predictor configuration in `JSON` format. Such configurations are directly integrated into predictor arguments, whether on a local or global scale. Configuration priorities are ranked as follows: 1. local (specific to the predictor), 2. global, and 3. default. Detailed CLI usage instructions are available in Appendix G, with a sample predictor configuration file presented in Appendix H.

7.1 Preprocessing

The class implementation of a preprocessing pipeline `Preprocessor` is stored in a `preprocessor.py` file. It has capabilities to be used both as a standalone tool for reading, processing, transforming, and then saving the data or as a component in larger applications (e.g., BenchStab). Each of the mentioned capabilities is the result of a sequence of the following preprocessing steps.

1. Gather all entries from the mutation file.
2. Determine the data type of each protein and obtain appropriate supplementary materials.
3. Obtain all lower-level protein formats (Figure 7.1).
4. Store the processed data as a row in the final data frame.

The first step in the preprocessing pipeline involves reading the user-provided source file line by line. Presently, only a single input format is accepted. This source file must adhere to specific formatting guidelines outlined in Section 6.1, with an illustrative example in Table 7.1. Lines that fail to meet these guidelines are omitted, and the remaining lines

are categorized by their protein input format into structures and sequences via the preprocessor's `parse` method. The classification of a protein as either a structure or a sequence depends on the user-supplied text format. If protein includes the `.fasta` extension, UniProt identifier, or raw FASTA sequence, it is automatically recognized as a sequence input, while others are classified as structures. Each category undergoes a distinct processing approach, with sequence processing handled by the `parse_sequence` method and structure processing by the `parse_struct` method.

Protein	**Mutation**	**Chain**	**Temperature (opt)**	**pH (OPT)**
1CSE	L45G	I		
./custom_seq.fasta	D420A		30	2.5
/custom_struct.pdb	N69L	A	25	7

Table 7.1: Example of mutation file. Protein, mutation, and chain parameters are mandatory, while temperature and pH levels are optional. Proteins in mutation records with `.fasta` suffix or fitting the UniProt ID/sequence regular expression are categorized as Fasta sequences, with `.pdb` suffix as a custom protein structure, while four letter codes as PDB entries.

Given that BenchStab supports various sequential input formats, it must possess the ability to identify each format accurately. For this purpose, it utilizes a collection of distinct regular expressions to differentiate between UniProt identifiers and raw sequences. The files are recognized by their `.fasta` extensions. When a record is designated as a FASTA file, it is also checked using BioPython [17] internal parser to confirm the file's integrity. Protein structure preprocessing mirrors the approach used for FASTA sequence analysis; PDB codes are validated using regular expressions, and structural files are recognized by their `.pdb` extension. The inclusion of the `.cif` format was contemplated; however, due to the lack of support from current predictors, this proposal was abandoned. Additionally, the integrity of PDB files is confirmed by using the BioPython PDB parser.

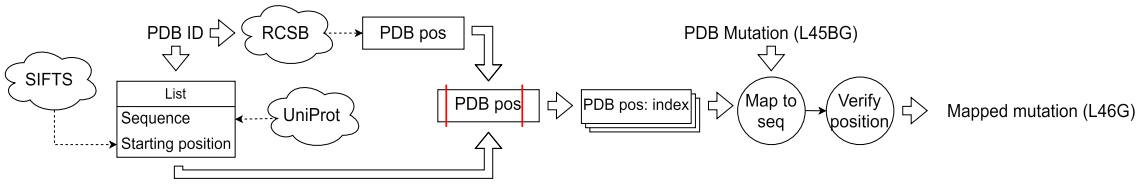


Figure 7.1: **Pipeline for sequential data retrieval from PDB ID entries**. The first step includes obtaining UniProt ID and initial sequence positions from SIFTS API, along with PDB position mapping from RCSB. Subsequently, the sequence of positions is trimmed at both ends to eliminate any custom mapping added by the RCSB entry authors. Ultimately, the positions from UniProt are synchronized with those from RCSB, and the mutant position is adjusted to match UniProt coordinates.

A unique capability of the Preprocessor is its competence in deducing protein input types at the sequence level from those at the structure level. This feature enables users to input the protein structure and engage the sequence-based predictor. When handling entries with a PDB ID, the Preprocessor prioritizes sequence retrieval from UniProt, and should this attempt be unsuccessful, it resorts to extraction from PDB. Sequences from PDB files are obtained directly from the file using BioPython's SeqIO. The problem with

mapping structures onto sequences is the difference in their amino acid position mapping. Employing mutation strings mapped to structures alongside UniProt sequences frequently leads to failures in predictions. To address this issue, I have developed a system that accurately aligns the PDB positions with those of UniProt (Figure 7.1).

If the parameters `verbose` or `outfolder` parameters are passed, `Preprocessor` will, in addition to data processing, perform a simple analysis via `Mapper` over the final data frame, calculating the number of:

- Mutations submitted.
- Unique proteins.
- Mutant amino acids with positive, negative, or no charge.
- Mutant amino acids with acidic, basic, or balanced side-chain properties.
- Polar/non-polar mutant amino acids.

`Mapper` class from `utils.amino acids` serves as a static container for storing information about amino acids. It holds data from about 20 naturally occurring amino acids in the proteins. Each stored record is defined by three-letter and one-letter amino acid codes and additional side-chain information, such as its polarity, acidity, and charge [61]. These records then allow `Mapper` to map between one and three-letter codes, which is required in some predictor implementations, and determine the side-chain properties from the supplied one-letter amino acid code.

The deployment of `Fasta`, `PDB`, and their superclass `File` from `utils.structure` significantly enhanced the code's readability and enabled additional customization to meet the requirements of FormData file formats. Specifically, the `File` class provides the `to_octet_stream` and `to_plain_text` methods, offering `BasePredictor` compatible outputs for *application/octet-stream* and *text/plain* formats. Additionally, by incorporating various built-in Python functions like `__eq__` and `__hash__`, these class objects can engage in `pandas` dataframe activities such as sorting, merging, and grouping. To address the issue of each new class instance initiating a database query or an I/O operation, a factory design pattern was utilized. This pattern enables the classes to maintain references to their objects, allowing them to return an existing reference instead of creating a new instance.

7.2 Predictor clients

The fundamental logic driving all predictor clients is derived from a single base class named `BasePredictor`, which is found in the `predictors/web/base.py` file. As illustrated in Figure 7.2, `BasePredictor` acts as both a blueprint to create specialized predictor clients and a framework for their core functionalities. The primary version of a predictor client lacks any HTTP communication capabilities, as these responsibilities are assigned to the subclasses that build on the `BasePredictor`. Rather, it comprehensively handles the logic required to convert the standardized dataset from the preprocessor into a specific format suitable for prediction web servers. Additionally, it incorporates advanced logic for tasks such as post-processing of results, dividing the dataset into batches as specified by the user, and generating regular reports on the progress of predictions.

Firstly, before any communication with the predictor begins, the data have to be transformed into a predictor-specific format. As already mentioned in Section 6.1, the **data transformation** pipeline consists of 2 optional steps and 1 mandatory step:

1. To match the structure of mutation lists accepted by prediction web servers, the user can allow (`group_mutations` parameter), and define the scale (`group_mutations_by` parameter) of dataset aggregation. For example, if the aggregation level is set to protein only, all mutation records with their specific chains will be aggregated into a single list. By default, if the aggregation is allowed but no scale is passed, the level is set to the protein, chain. (Optional)

2. For each transformed record, the `prepare_payload` method is applied. This method allows the user to define a custom `POST` request payload to fit both the needs of the prediction web server and the data type supported by the predictor. (Mandatory)

3. Lastly, if the prediction web server accepts only form-encoded data, the entire payload is converted to the shape accepted by `aiohttp` parser `FormData`, which subsequently encodes the data in a form-specific format. (Optional)

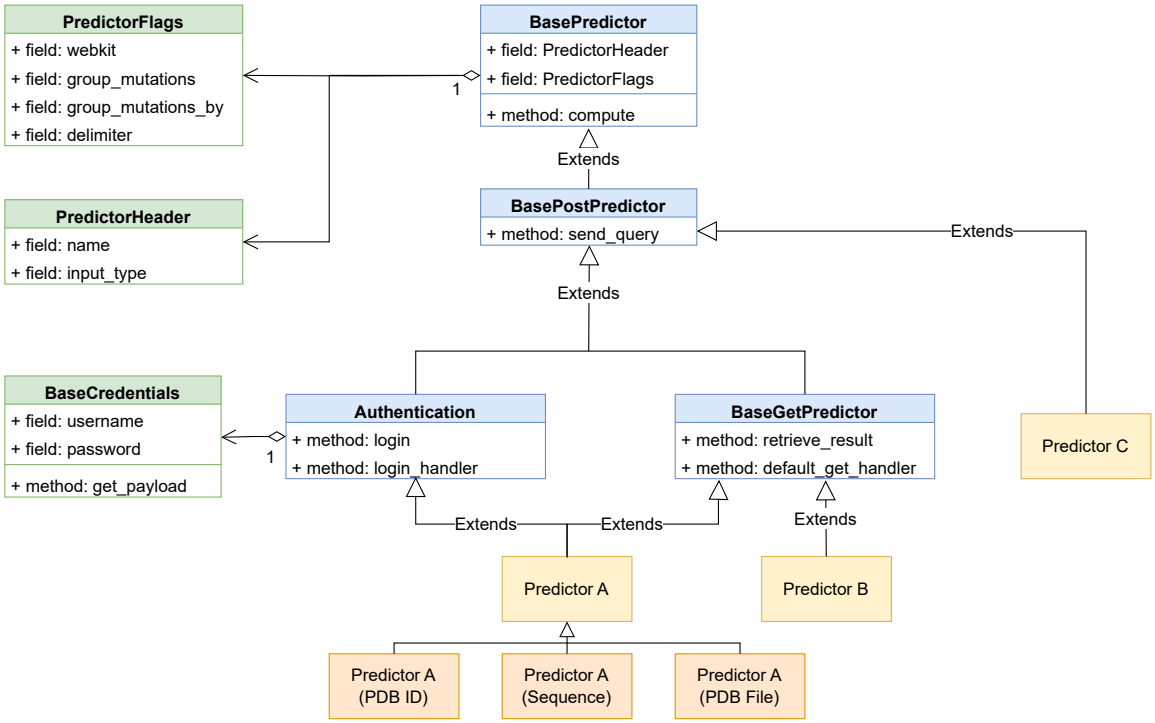


Figure 7.2: Class diagram illustrating the relationships and inheritance among all classes related to predictors, with `BasePredictor` positioned at the top of the hierarchy. `BasePredictor` functions as both a framework and an implementation of the basic predictor features, and its capabilities are enhanced by `BasePostPredictor`, which introduces communication through `POST` requests. Both `Authentication` and `BaseGetPredictor` further develop the post-predictor client by incorporating authentication mechanisms and a waiting loop for job-oriented predictors. These form the basis for client-side implementations of specific prediction tools, depicted symbolically as `Predictor A`, `B`, and `C`. Ultimately, these classes are enhanced by specifying their input data type format, such as `predictor A (PDB ID)`.

To effectively handle the requests directed to the predictor, the predictor client employs fixed-size asynchronous dynamic queues. The size of the queue indicates the number of

simultaneous requests that can be sent to the predictor server, and it can be adjusted by the user via the `batch_size` parameter. Once all predictions are completed, the queue is de-allocated, and the predictor client concatenates and returns all outcomes, including both failures and successes.

The primary cycle of the prediction client (Algorithm 1) operates in the following manner:

Algorithm 1 The main predictor loop of the predictor client.

```
while True do
    task ← queue.get()
    start_timer()
    if not login(task) then
        task ← done
        continue
    end if
    if not send_query(task) then
        task ← done
        continue
    end if
    while retrieve_result(task) do
        if not timeout then
            status ← Timeout
            break
        end if
        timeout ← timeout − 1
        continue
    end while
    task ← done
end while
```

In the standard implementation of `BasePredictor`, the methods `login()`, `send_query()`, and `retrieve_result()` all return True because their actual implementations are provided in the subclass. This design enables users to conveniently choose the components they want to incorporate into their specific predictor version. For instance, PoPMuSiC extends `BaseAuthentication`, handling the authentication process, and `BaseGetPredictor`, a derivative of `BasePostPredictor`, thereby managing both the sending of queries and the retrieval of results. Each method sets a distinct `status` for the prediction, indicating its stage in the prediction process. This feature is beneficial because it enables users to quickly determine the data points for which predictions are missing. Additionally, the status is also used in `retrieve_result()`, where all predictions already finished are exempted.

Therefore, as the process is uniform across all predictors, the essential methods a user needs to implement for adding a new predictor client are `prepare_payload`, `default_post_handler()`, which processes the response from the `POST` request made by `send_query()`, and potentially `default_get_handler()`, which processes the `GET` request from `retrieve_result()`. Since the responses are usually in HTML format, BenchStab includes `HTMLParser`, providing valuable tools for XPath and HTML table parsing.

Chapter 8

Results and discussion

This chapter conducts a comparative review of key protein stability predictors, assessing their effectiveness through diverse perspectives and multiple metrics. Through an examination of their predictive capability, dependability, and computational speed, I aim to provide insights that help researchers choose the optimal predictor for their specific needs. In addition, beyond the technical details of the review, I also explore potential conclusions that can be drawn from the benchmarking of these predictors.

In the benchmark analysis, a total of 27 predictors were initially considered, as detailed in Chapter 5. However, due to the issues described in Chapter 6, only 18 of these predictors were implemented and evaluated. Subsequent adjustments to this subset involved the exclusion and inclusion of certain predictors:

+ **FoldX4**, **FoldX5** [76]: two representatives of the state-of-the-art stability predictors to provide comparison between online and offline predictors.

– **SDM**: unavailability of the predictor at the time of benchmarking.

– **sRIDE**: the output of the predictor (stabilizing wild-type residues in the protein) could not be subjected to the same analysis as the outputs of the other predictors.

Figure 8.1 offers a detailed comparison of the predicted predictors evaluated and chosen, encompassing both their sequential and structural forms. As a benchmark dataset, I used the Benchstab dataset (Chapter 4). The dataset includes 289 unique mutants in 36 proteins that are not well known ($< 50\%$ sequence identity) to predictors. In addition, it includes several unusual protein structures that might challenge some predictors. To normalize all $\Delta\Delta G$ predictions across all predictors, all results were adjusted to follow the following format:

- **Destabilizing**: predictions with $\Delta\Delta G < 0$.
- **Stabilizing**: predictions with $\Delta\Delta G > 0$.

The analysis of the results was conducted from three different angles: considering the average prediction time, visualizing the results in various graph formats, and finally assessing the results using diverse metrics.

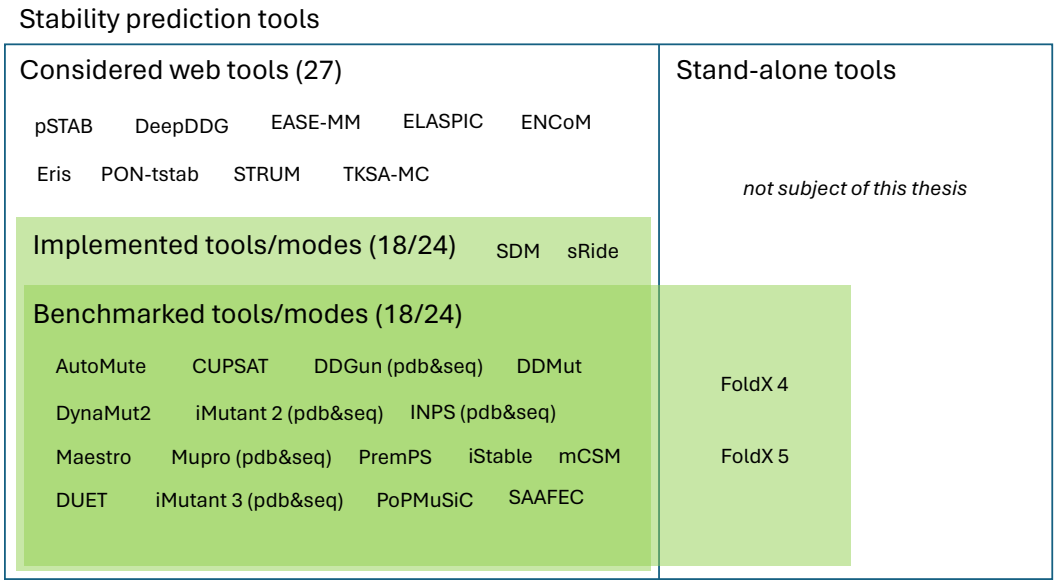


Figure 8.1: **Venn diagram of prediction tools** investigated in this work. Some of the web tools were not implemented in BenchStab, e.g. due to their unavailability. The numbers for the respective groups are counts of the predictor tools/predictor modes.

8.1 Prediction time

A crucial factor influencing the overall effectiveness of web-based predictors is the speed of their predictions. Quicker prediction times lead to increased submission capacity, reduced queue lengths, and decreased waiting periods, enhancing the user experience. To evaluate the performance of the implemented predictors in terms of prediction speed, the average prediction time for all predictors tested was measured using the Benchstab dataset.

Given that the Benchstab includes prediction time measurement as a feature, external timing was unnecessary. Utilizing the internal clock of the Benchstab offers the benefit that the recorded prediction time excludes any preprocessing and internal (client) queueing delays, as the timing begins with the prediction submissions. Therefore, the prediction time comprises the following.

$$t_{prediction} = t_{submission} + t_{queue} + t_{prediction} \tag{8.1}$$

where t_{queue} denotes the time predictors spend in the external queue, with a value of 0 for POST-only predictors. A challenge with this method was that certain predictors (e.g. Dynamut, INPS) process multiple mutations in a single job, which means that the recorded time reflects the duration for a batch of mutations. To equalize the prediction times between jobs that handle single mutations and those that manage mutation batches, the times for batch jobs were averaged according to the number of mutations they included.

The threshold values for evaluating prediction time were established as follows: the predictor with $t_{prediction}$ <= 120 seconds was considered “fast”, while the predictor with $t_{prediction}$ >= 10 minutes was considered “slow”. The results presented in Table 8.1 show that most of the predictors are fast, except for SAAFEC, where most of the predictions

submitted failed on the internal Benchstab timeout. At the time of writing this thesis, the problem with SAAFEC persists since any subsequent automatic submissions by Benchstab were also timed out, and manual submissions hung indefinitely.

Predictor	Prediction time per mutant	Predictor	Prediction time per mutant
Automute	17 s	DDMut	119 s
INPS	4.5 min	CUPsat	7 s
DDGun	82 s	Maestro	20 s
DUET	41 s*	mCSM	23 s
I-Mutant2	80 s	PoPMuSiC	6 min
iStable	2 s*	PremPS	8.3 min
SAAFEC-SEQ	39 min	MUPro	5 s
DynaMut2	2.7 min	sRIDE	92 s

*The predictor utilizes caching, i.e., provides faster results for repeated inputs.

Table 8.1: Average prediction time per single mutant for all benchmarked web-based predictors. I measured the prediction time in October 2023 for all successful (without timeouts or failures) predictions on our dataset. The time may have been affected by the predictor queue sizes at the query time (including wait time). The color in the last column highlights fast (green, <2 min) and slow (orange, >10 min) predictors.

One of the techniques some predictors utilize to procure fast predictions is result caching, with the most notable examples being iStable and DUET. In the case of iStable, the average prediction time for a protein/mutation combination already seen was < 1 second, while the time for the new predictions was significantly longer, around 40 seconds.

8.2 Visual analysis

This section analyzes the measured protein stability predictor $\Delta\Delta G$ results, incorporating correlation graphs, ROC curves, PR curves, kernel density estimates, and heat maps. These visual tools collectively offer a comprehensive overview of the predictive performance of the benchmarked predictors.

The correlation graphs project the relationship between the predicted and measured protein stabilities, providing insights into the consistency and accuracy of our predictions. For visibility purposes, outliers that do not belong to the range $[-7, 7]$ $\Delta\Delta G$ were removed from the graphs. ROC and PR curves further assess classification performance, highlighting trade-offs between true positive rate, false positive rate, precision, and recall. Kernel density estimates provide a detailed overview of the $\Delta\Delta G$ prediction distributions relative to the actual distributions, further supporting the data displayed in the correlation graphs.

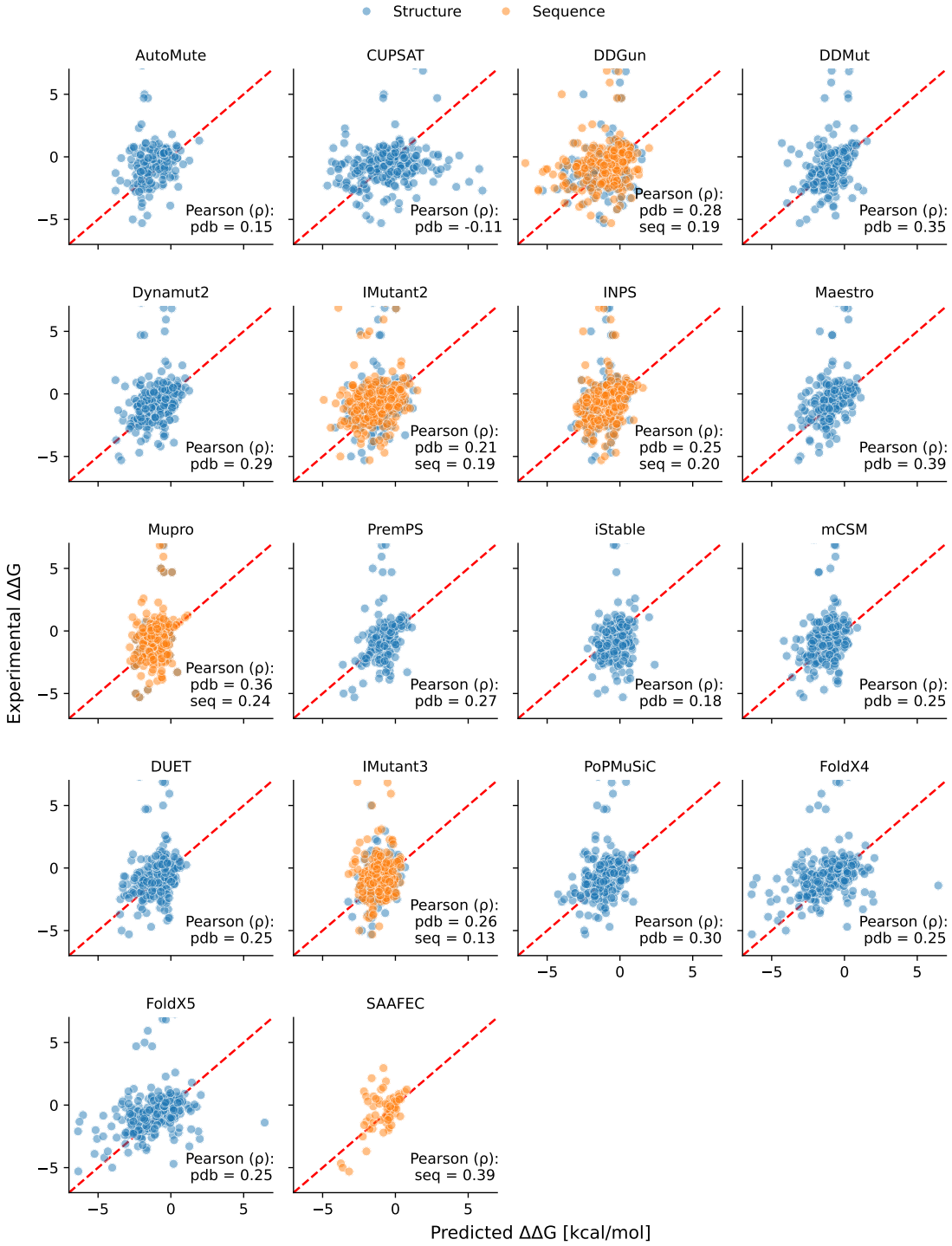


Figure 8.2: **Correlation graphs between predicted ΔΔG(kcal/mol) and the ground truth**. Each subgraph corresponds to one tool. Each dot corresponds to a prediction for a particular mutant either from a protein sequence (orange, seq) or a protein structure (blue, pdb). The red dashed line represents the ideal prediction. For the presentation purpose, the subgraphs are clipped to range [-7, 7], but the outlying points are still included in the calculation of the Pearson correlation coefficients. From the positions of the central clusters which are shifted above the diagonals, you can observe that many predictors predict more negative effects than is the ground truth, e.g. AutoMute, mCSM. This suggests a bias towards the prediction of destabilizing mutations (negative $\Delta\Delta G$). Also notably, the predictors mispredict a cluster of phenylalanine and cysteine mutants (middle-top in the graphs), which have very high stabilizing experimental $\Delta\Delta G$ (5 kcal/mol), as neutral ($\Delta\Delta G$ close to 0).

Lastly, I have used heat maps to assess whether certain distinctive traits of amino acids/proteins influenced the overall prediction score. I examined the balanced accuracy and mean absolute error of the prediction outcomes in various categories:

- **Chemical properties** of the amino acid molecules.
- **Volume** of the amino acid molecules.
- **Hydropathy**, **polarity**, and **charge** of the amino acids.
- **SCOP class** belonging to the queried proteins. For each protein in the dataset, the class was obtained via the SIFTS API [21] by its protein PDB identifier.

where my focus was not on the properties of the amino acids per se, but rather on the changes in properties due to mutation ($property_{wt} => property_{mut}$). The mapping of amino acid properties was obtained from the international ImMunoGeneTics information system [61]. A predictor/property combination was considered statistically significant only if it involved more than 5 results.

Correlation graphs

The initial observation in the correlation graphs (Figure 8.2) reveals that the predictions of all predictors noticeably deviate from the ideal predictor (red dashed line), with ρ in the range $0.2 < \rho < 0.4$. For Automute and the sequential variant of I-Mutant 3, the correlation coefficients are notably lower ($0.1 < \rho < 0.2$), while for CUPSAT even negative ($\rho = -0.11$), indicating a greater deviation from the experimental results. These figures agree with the results of a study by *Khan et al.* [39], which reported comparable outcomes for these predictors. Furthermore, it is important to highlight that the number of successful predictions is the only difference between the sequential and structural forms of Mupro, suggesting that the additional structural data did not affect the prediction results for the Benchstab dataset.

A more detailed examination shows that the majority of the predicted $\Delta\Delta G$ values (x-axis) tend to gather near 0, with a slight incline towards negative values. This observation suggests two possible interpretations:

1. The inability of most predictors to accurately forecast substantial changes in the $\Delta\Delta G$.
2. The dominance of destabilizing mutations within the training datasets influences the predictors to anticipate the destabilizing change in $\Delta\Delta G$ values more frequently.

The $\Delta\Delta G$ distributions illustrated in the **kernel density estimates** (Figure D.1) support these observations. Furthermore, the KDE graph confirms the validity of these observations in both structural and sequential variants of the predictors.

Another interesting topic for discussion involves the unique cluster of outliers found in the upper regions of the correlation graphs for many predictors (such as Mupro, PremPS, and PoPMuSiC). The protein/mutation examples observed most frequently in this outlier group are shown in Table 8.2:

A common characteristic of all datapoints in the outlier cluster is their exceptionally high experimentally measured $\Delta\Delta G$ values. This suggests that these mutations significantly enhance protein stability and may lead to the formation of new bonds, presenting challenges in modeling. For example, at datapoints with protein 1FEP, all WT amino acids mutate specifically to cysteine, suggesting a likely formation of new disulfide bridges.

PDB ID	Wild type	Position	Mutant
1CS2	Methionine (M)	69	Alanine (A)
1FEP	Glutamic acid (E)	248	Cysteine (C)
	Glycine (G)	252	Cysteine (C)
	Leucine (L)	244	Cysteine (C)
	Serine (S)	246	Cysteine (C)
	Serine (S)	254	Cysteine (C)
2ADA	Tryptophan (W)	117	Phenylalanine (F)
	Tryptophan (W)	161	Phenylalanine (F)
	Tryptophan (W)	272	Phenylalanine (F)

Table 8.2: The most frequently occurring datapoints from the outlier cluster in the upper echelon of correlation graphs from Figure 8.2.

ROC/PR

Examination of ROC curves (Figure B.1) and PR curves(Figure C.1) reveals that for many threshold settings, the performance of CUPsat and Automute closely mirrors that of a random predictor (indicated by a red dashed line). In contrast, all other benchmarked predictors significantly surpass the random predictor. The PR curve also indicates that the sequential versions of these predictors are more effective in predicting stabilizing mutations than their structural counterparts. It is unexpected considering that, theoretically, the additional context learned from the structural data is supposed to enhance the performance of the predictors. However, this observation is consistent with findings from a different research conducted by *Pucci et al.* [56], which implies that the structural information may not have been used effectively.

Visualizing the default predictor thresholds $\Delta\Delta G = 0\ kcal/mol$ (shown as a cross on the graph) has shown that several predictors (such as INPS, Maestro, FoldX) might positively benefit from choosing different threshold values.

Heat maps

Analysis of the effects of amino acid property transitions such as **hydropathy**, **polarity**, and **charge** on predictor results reveals that both in the balanced accuracy heat map (Figure E.3) and in the mean absolute error heat map (Figure F.3), these changes in properties had no significant impact on the final prediction scores. As expected, scenarios where properties had changed (e.g. positive charge => uncharged) resulted in higher BA and lower MAE compared to scenarios with no changes (uncharged => uncharged).

Most amino acid transitions, in terms of their **chemical properties**, do not show any significant deviations from the average according to balanced accuracy (Figure E.2). The only notable finding is that, with the exceptions of DDMut and iStable, the predictors generally fail to accurately forecast the transition from basic to aromatic amino acids. In contrast, the mean absolute error heat map (Figure F.2) indicates that, on average, all predictors accurately determine the precise change in $\Delta\Delta G$ for transitions involving the aliphatic group, with an average $MAE < 1\ kcal/mol$. Amino acids such as leucine, alanine, and glycine, which are part of the aliphatic group, rank as some of the most prevalent amino acids in proteins [96]. Consequently, these amino acids are likely to be predominant in protein stability datasets, providing more information from which predictors can learn.

At first glance, Figure F.1 appears to suggest that the predictors do not accurately predict the precise $\Delta\Delta G$ values for transitions towards *small* mutants (medium => small, large => small, very small => small). This observation is supported by the balanced accuracy results shown in Figure E.1. However, the sample size is insufficient and the observed trend does not seem rational from a **volume** perspective, considering that the transitions involve both smaller and larger wild types of amino acids. This suggests the presence of additional factors that influence the overall accuracy, beyond just volume.

Comparative heat maps of predictor performance, classified by the **SCOP classification** of proteins, have shown conflicting results between their MAE (Figure F.4) and BA (Figure 8.3) metrics. Although the balanced-accuracy heat map indicates that rarer classes such as coils and membrane/surface proteins are more difficult to classify, the MAE heat map demonstrates that predictors tend to accurately determine the exact change in their $\Delta\Delta G$ (average $MAE < 1\ kcal/mol$). In particular, low MAE scores combined with poor BA scores suggest that the estimated values of $\Delta\Delta G$ tend to stay close to zero. Consequently, mutations in these protein classes are unlikely to result in significant changes in $\Delta\Delta G$, suggesting that improved classification thresholds could lead to substantially improved BA scores.

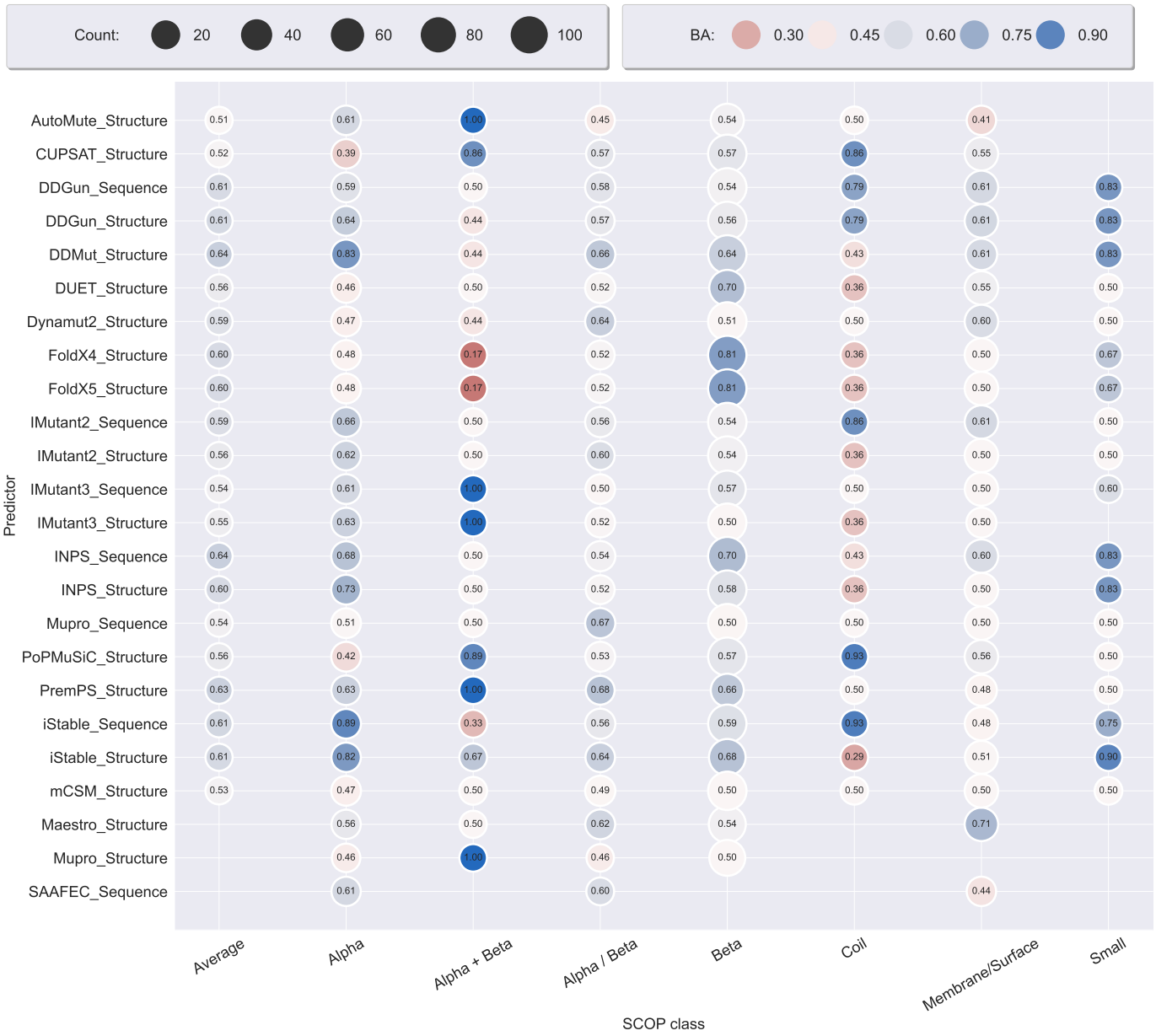


Figure 8.3: Average balanced accuracy (BA) based on the protein SCOP [45] class, per predictor/type combination. The color of the circle determines the balanced accuracy value, while the size of the circle number of datapoints belonging to respective predictor/SCOP class combination. Each combination had to have at least 5 successful predictions to be present in the graph. It can be seen that protein classes, which are more difficult to model, such as **coils** or **membrane/surface proteins** also proved to be more difficult to predict in comparison to **alpha** or **beta** proteins.

8.3 Quantitative analysis

This section encapsulates a comprehensive quantitative analysis of the predictor results based on the various discrete and continuous metrics discussed in Chapter 5. In obtaining each respective metric, no datapoint was excluded from the calculation. Table 8.4 displays the complete count of successful predictions for each predictor and their specific input variants, under the column **Results**. Remarkably, except for mCSM, only the sequential predictor models were able to predict all 289 datapoints in the BenchStab dataset successfully. This probably stems from the inclusion of uncommon protein structures, such as 1EL1, which are difficult for some predictors to model or analyze efficiently. A prominent example is AutoMute, which failed to tessellate the protein structure [80] at 118 of the 289 datapoints. However, the predictor with the lowest success rate, SAAFEC, predominantly failed due to excessive queue times that lasted more than two days.

The prevalence of negative values in the **MSD** column indicates that the predictors tend to forecast mutations as more destabilizing than they are actually. This supports the data seen in the correlation and KDE graphs and reinforces the conclusions of previous research ([88, 73]). The observed **BA** values, ranging from 51% to 64%, suggest that these predictors are somewhat effective for mutation prioritization[1]. Among the competitors, the top three, listed in ascending order, were PremPS, INPS (sequence), and DDMut. Interestingly, each of these leading predictors utilizes machine learning, indicating that the use of machine learning to predict protein stability appears promising.

Method	Type	#	MSD	ACC	PCC	MCC	BA	MAE	R^2
MA	Structure	1046	-0.269	0.703	0.266	0.212	0.592	1.168	0.071
MA	Sequence	283	N/A	0.717	N/A	0.248	0.614	N/A	N/A
ML	Structure	1818	-0.435	0.738	0.252	0.218	0.576	1.148	0.064
ML	Sequence	1193	-0.595	0.739	0.199	0.257	0.582	1.294	0.039
FE	Structure	1375	-0.417	0.686	0.151	0.165	0.578	1.415	0.023
FE	Sequence	289	-0.457	0.73	0.194	0.249	0.605	1.446	0.038

Cut-offs of good and bad performers: 1.2 < MAE < 1.5, 0.25 < |MSD| < 0.6, Accuracy > 0.7, BA > 0.6, PCC/MCC > 0.2, R2 > 0.1

Table 8.3: Assessment of the web-based protein stability prediction tool methods on my benchmark dataset derived from FireProtDB. The following metrics were used: mean absolute error (**MAE**), mean signed deviation (**MSD**), highlighting a potential bias toward destabilizing/stabilizing mutations, accuracy and balanced accuracy (**BA**), Pearson correlation coefficient (**PCC**), Matthews correlation coefficient (**MCC**), and R-squared ($\mathbf{R^2}$). The respective methods are as follows: machine learning (**ML**), free energy calculations (**FE**), and meta approach (**MA**). I used the cut-off of 0 kcal/mol for the binary classification.

In contrast, most of the tools demonstrated significantly poor performance in regression[2], with a $\mathbf{R^2}$ ranging from 0.01 to 0.15. Regrettably, this claim is not corroborated by either **MAE** (which varies from 1 $kcal/mol$ to 2 $kcal/mol$) or **RMSE** (which varies from 1.5 $kcal/mol$ to 3.1 $kcal/mol$). For example, considering mutations with minimal impact

[1] This refers to the process where an initial automated assessment is employed to lessen the number of variants requiring manual review.

[2] Predicting the exact change in protein stability.

(e.g. $-1\ kcal/mol < \Delta\Delta G < 1\ kcal/mol$), the mean absolute error calculations reveal that all predictors grossly misestimate the change in $\Delta\Delta G$ post-mutation. Conversely, for mutations causing substantial alterations in $\Delta\Delta G$ ($> 5\ kcal/mol$), a deviation of approximately 1.5 $kcal/mol$ might still be deemed relatively accurate.

The observed **PCC** primarily indicates a weak correlation ($pcc = (0, 0.3]$) and occasionally a moderate correlation ($pcc = (0.3, 0.5]$) between the predicted and the experimental results. The leaders in the performance chart among the predictors are SAAFEC, Maestro, MuPro (structure), and DDMut, highlighting the superiority of machine learning-based predictors. Much like the PCC, the **Matthews** correlation coefficient predominantly shows a weak correlation, with sporadic instances of moderate correlation. They also reinforce the supremacy of machine learning predictors, the top performers being SAAFEC, PremPS, INPS, and DDMut.

An alternative viewpoint on the predictive performance of $\Delta\Delta G$ is illustrated in Table 8.3. Rather than focusing on the predictors, this table displays the efficacy of various methods and protein input format combinations. Regrettably, it is not possible to derive any conclusions from the sequential versions of either the meta approach (iStable) or the free energy (DDGun) predictors, due to their significantly smaller sample sizes compared to other combinations. Moreover, it is probable that the underperforming outliers (CUPSAT, AutoMute) adversely influenced their respective groups. Taking these elements into account, it is evident that sequential machine learning predictors excel in discriminative metrics, surpassing others in terms of accuracy, balanced accuracy, and the Matthews correlation coefficient. Conversely, the structural meta-approach and structural machine learning methods surpass other techniques in continuous metrics such as PCC, MSD, or MAE.

Predictor	Type	Method	Results	Accuracy	BA	PCC	MCC	R^2	RMSE	MAE	MSD
AutoMute	Structure	Machine Learning	171	0.643	0.513	0.15	0.04	0.023	1.981	1.31	-0.632
CUPSAT	Structure	Free Energy	260	0.596	0.524	-0.111	0.045	0.012	3.145	1.948	-0.22
DDGun	Sequence**	Free Energy	289	0.73	0.605	0.194	0.249	0.038	2.223	1.446	-0.457
DDGun	Structure	Free Energy	276	0.714	0.607	0.279	0.231	0.078	1.931	1.193	-0.106
DDMut	Structure	Machine Learning	270	0.73	0.638	0.349	0.302	0.122	1.894	1.109	-0.247
DUET	Structure	Meta approach	263	0.696	0.557	0.247	0.148	0.061	2.018	1.221	-0.415
Dynamut2	Structure	Meta approach	289	0.73	0.586	0.285	0.222	0.082	1.931	1.144	-0.397
FoldX4	Structure	Free Energy	277	0.7	0.602	0.255	0.212	0.065	2.296	1.406	-0.628
FoldX5	Structure	Free Energy	277	0.7	0.602	0.255	0.212	0.065	2.296	1.406	-0.628
IMutant2	Structure	Machine Learning	276	0.728	0.562	0.207	0.173	0.043	2.039	1.225	-0.579
IMutant2	Sequence**	Machine Learning	289	0.74	0.593	0.189	0.249	0.036	2.194	1.422	-0.795
IMutant3	Sequence*	Machine Learning	262	0.721	0.541	0.128	0.179	0.016	2.113	1.352	-0.683
IMutant3	Structure*	Machine Learning	218	0.775	0.551	0.256	0.173	0.065	1.526	0.939	-0.252
INPS	Structure	Machine Learning	277	0.762	0.602	0.254	0.296	0.065	1.915	1.081	-0.315
INPS	Sequence**	Machine Learning	289	0.772	0.638	0.203	0.357	0.041	1.984	1.17	-0.379
Maestro	Structure	Meta approach	244	0.717	0.622	0.388	0.289	0.151	1.919	1.153	-0.469
Mupro	Structure	Machine Learning	126	0.786	0.509	0.36	0.049	0.129	1.807	1.086	-0.57
Mupro	Sequence**	Machine Learning	289	0.74	0.543	0.243	0.2	0.059	1.948	1.248	-0.524
PoPMuSiC	Structure	Free Energy	285	0.712	0.559	0.303	0.157	0.092	1.917	1.163	-0.488
PremPS	Structure	Machine Learning	191	0.759	0.632	0.266	0.358	0.071	2.207	1.213	-0.328
SAAFEC	Sequence*	Machine Learning	64	0.656	0.615	0.395	0.39	0.156	2.127	1.238	-0.623
iStable	Sequence**	Meta approach	283	0.717	0.614	N/A	0.248	N/A	N/A	N/A	N/A
iStable	Structure	Meta approach	250	0.664	0.608	0.179	0.208	0.032	1.915	1.155	0.225
mCSM	Structure	Machine Learning	289	0.727	0.533	0.249	0.134	0.062	1.996	1.222	-0.62

* Reported as collected in the previous run (April 2023), due to the unavailability of the predictor in October 2023. ** Collected in February 2024

Cut-offs of good and bad performers: 1.8 < RMSE < 2.0, 1.2 < MAE < 1.5, 0.25 < |MSD| < 0.6 , 0.76 > Accuracy > 0.7, BA > 0.6, 0.3 > PCC/MCC > 0.2, R2 > 0.1

Table 8.4: Assessment of the web-based protein stability prediction tools on my benchmark dataset derived from FireProtDB. The following metrics were used: root mean squared error (**RMSE**), mean absolute error (**MAE**), mean signed deviation (**MSD**), highlighting a potential bias toward destabilizing/stabilizing mutations, binary accuracy and balanced accuracy (**BA**), Pearson correlation coefficient (**PCC**), Matthews correlation coefficient (**MCC**), and R-squared ($\mathbf{R^2}$). I used the cut-off of 0 kcal/mol for the binary classification. Results were collected in October 2023 if not stated otherwise. For the calculation of FoldX results, I used the 2024 Mac binaries with the RepairPDB module, followed by the PssmStability module setting pH equal to 7 and –water=CRYSTAL. The green or orange color represents a good or bad (see below for the cut-offs) performer, respectively, for a given metric. The best performers are marked by their place in parentheses.

Chapter 9

Conclusion

The main objective of this Thesis was to create a software tool that allows for automated querying of protein stability predictors. Subsequently, this tool should have been used to establish a benchmark of the available web predictors. My first step towards this goal was to summarize all relevant information into the theoretical part of the Thesis.

Its first Chapter 2 aims to introduce the reader to the realm of proteins, explaining concepts such as amino acids and protein structures. Next, I delved into the subject of protein stability (Chapter 3), examining various methods of acquisition and prediction. Following this, I investigated various protein stability datasets (Chapter 4) and discussed their limitations.

One such limitation is their likely inclusion in the training or validation sets of various predictors. To ensure an impartial evaluation, I created a new dataset. I chose FireProtDB as the source due to its benefits, such as manual curation and rigorous submission guidelines. Next, I followed several steps (Section 4.2), such as extra manual filtering, compiling all training datasets from the evaluated predictors, and mapping UniProt IDs to UniRef50 clusters. The resulting BenchStab is freely available in Zenodo [91] and contains 289 mutations in 36 proteins. It also boasts the unique feature that each protein has less than 50% sequential identity with any protein in the training sets of the evaluated predictors.

After securing the data source for the benchmark, the next step was to develop the software tool itself. To ensure that it will be accessible on nearly every device, I have endeavored to reduce both the computational demands and the network traffic involved when accessing the predictors. In addition, my goal was to provide a very straightforward process for the integration of new predictors. To achieve this, Python was selected as the programming language. During the initial stages of development, it was necessary to examine the operational processes of protein stability predictors, which included monitoring all network communications and examining all potential configurations (Chapter 6). Armed with this knowledge, I developed and launched robust software called BenchStab. Due to the unavailability and malfunctions of the predictor webservers, Benchstabs implements 18 of the 27 predictors considered. This tool is thoroughly documented and available for free on GitHub [6] in two ways: as a standalone Python library and as a CLI application (Appendix G).

BenchStab consists of two primary processes: preprocessing (Section 7.1) and communication with predictors (Section 7.2). The preprocessing component enables users to specify the protein query in any popular format, from raw Fasta sequences to PDB identifiers. Furthermore, BenchStab's preprocessing includes a cascade data acquisition system, which automatically deduces the lower-level format (sequence) when a higher-level format (struc-

ture) is provided. This feature facilitates accurate mapping of mutation positions between PDB structures and UniProt sequences, allowing the query of sequential predictors using structural data. To minimize overhead in the predictor-querying section of BenchStab, I have not chosen to use any web scraping frameworks and instead opted for exclusive use of HTTP requests. Moreover, to prevent the usual context-blocking tendencies of the predictors, all communications are executed asynchronously. This approach guarantees a semiparallel handling of queries across different predictors, enhancing the speed and reducing the load of BenchStab's internal operations. Finally, to demonstrate the straightforward scalability of BenchStab, PonSol2, a solubility predictor for proteins, has been integrated.

During the benchmarking process, both the newly created dataset and the software tool were used. The performance of the predictors was evaluated from three perspectives: prediction time (Section 8.1), graphical plot analysis (Section 8.2), and quantitative evaluation using different metrics (Section 8.3). When comparing average prediction times (Table 8.1), it was found that except for SAAFEC, PremPS, and PoPMuSiC, all predictors delivered their results within a reasonable time frame (2 to 180 seconds). The correlation graph (Figure 8.2) reveals that the predicted values $\Delta\Delta G$ show only a weak to moderate correlation with the experimentally measured values. Furthermore, the plot suggests a potential bias towards destabilizing mutations and a lack of accuracy in the predictors' ability to predict significant changes in $\Delta\Delta G$. Kernel density graphs also support these findings. Analysis of ROC curves (Appendix B) indicates that although most predictors perform adequately, the performance of CUPSAT and AutoMute closely mirrors that of a random predictor. PR curves of the predictors (Appendix C) of reveal that the sequential variants generally outperform the structural variants in terms of predictive performance. Finally, evaluating the impact of various amino acid/protein properties on the total prediction score (Figure 8.3, Appendices E F) revealed no notable results compared to the mean prediction performance of the predictors.

Quantitative analysis (Table 8.4) confirmed the results discussed previously, with the observed balanced accuracy between 51% and 64% and a weak to moderate correlation demonstrated by the Matthews and Pearson correlation coefficients. Almost all the tools showed a particularly poor performance in regression, i.e. predicting the exact change in protein stability (the worst and best R^2 equal to 0.01 and 0.15, respectively). Furthermore, the vast majority of the tools tested showed a bias towards destabilizing mutations (Appendix D.1), also shown by the mean signed deviation ranging from -0.79 to -0.11.

In this work, I implemented BenchStab, a fast and automatic tool for acquiring predictions from predictor websites. To achieve this, I created a unique data set with fewer than 50% sequence identity compared to the training data of the predictors. Using this dataset and BenchStab, I analyzed and evaluated the predictive ability of 18 different predictors. The results confirmed the practical applicability of the predictors, while also highlighting some minor deficiencies that could be targets for future improvements.

Bibliography

[1] Ahmad, F. Protein stability [determination] problems. *Frontiers in Molecular Biosciences*. 2022, vol. 9. DOI: 10.3389/fmolb.2022.880358. ISSN 2296-889X. Available at: https://www.frontiersin.org/articles/10.3389/fmolb.2022.880358.

[2] Alan D. McNaught compiled by and Wilkinson, A. *Compendium of chemical terminology : IUPAC recommendations*. Second edition. Oxford [Oxfordshire] ; Malden, MA : Blackwell Science, 1997., 1997. At head of title: International Union of Pure and Applied Chemistry.;Bibliography: pages 447-450. Available at: https://search.library.wisc.edu/catalog/999830115402121.

[3] Alberts, B., Johnson, A. and Lewis, J. e. a. *Molecular Biology of the Cell. 4th edition*. New York: Garland Science, 2002. Available at: https://www.ncbi.nlm.nih.gov/books/NBK26830/.

[4] Barlow, D. and Thornton, J. Ion-pairs in proteins. *Journal of Molecular Biology*. 1983, vol. 168, no. 4, p. 867–885. DOI: https://doi.org/10.1016/S0022-2836(83)80079-5. ISSN 0022-2836. Available at: https://www.sciencedirect.com/science/article/pii/S0022283683800795.

[5] BENESCH, R., BENESCH, R. E., BOYER, P. D., KLOTZ, I. M., MIDDLEBROOK, W. R. et al., ed. AUTHOR INDEX. In: BENESCH, R., BENESCH, R. E., BOYER, P. D., KLOTZ, I. M., MIDDLEBROOK, W. R. et al., ed. *Sulfur in Proteins*. Academic Press, 1959, p. 443–452. DOI: https://doi.org/10.1016/B978-0-12-395705-4.50034-X. ISBN 978-0-12-395705-4. Available at: https://www.sciencedirect.com/science/article/pii/B978012395705450034X.

[6] Berezný, M. and Velecký, J. *BenchStab* [https://github.com/loschmidt/BenchStab]. GitHub, 2024.

[7] Bigman, J. L. and Reinhardt, K. A. Chapter 11 - Monitoring of Chemicals and Water. In: Reinhardt, K. A. and Kern, W., ed. *Handbook of Silicon Wafer Cleaning Technology (Third Edition)*. Third Editionth ed. William Andrew Publishing, 2018, p. 619–657. DOI: https://doi.org/10.1016/B978-0-323-51084-4.00011-3. ISBN 978-0-323-51084-4. Available at: https://www.sciencedirect.com/science/article/pii/B9780323510844000113.

[8] Blaabjerg, L. M., Kassem, M. M., Good, L. L., Jonsson, N., Cagiada, M. et al. Rapid protein stability prediction using deep learning representations. *BioRxiv*. Cold

Spring Harbor Laboratory. 2022. DOI: 10.1101/2022.07.14.500157. Available at: https://www.biorxiv.org/content/early/2022/08/07/2022.07.14.500157.

[9] Brändén, C. and Tooze, J. *Introduction to Protein Structure*. Garland Pub., 1999. ISBN 9780815323051. Available at: https://books.google.sk/books?id=zsWcpqrgG74C.

[10] Caldararu, O., Blundell, T. L. and Kepp, K. P. A base measure of precision for protein stability predictors: structural sensitivity. *BMC Bioinformatics*. Feb 2021, vol. 22, no. 1, p. 88. DOI: 10.1186/s12859-021-04030-w. ISSN 1471-2105. Available at: https://doi.org/10.1186/s12859-021-04030-w.

[11] Cao, H., Wang, J., He, L., Qi, Y. and Zhang, J. Z. DeepDDG: Predicting the Stability Change of Protein Point Mutations Using Neural Networks. *Journal of Chemical Information and Modeling*. American Chemical Society. Apr 2019, vol. 59, no. 4, p. 1508–1514. DOI: 10.1021/acs.jcim.8b00697. ISSN 1549-9596. Available at: https://doi.org/10.1021/acs.jcim.8b00697.

[12] Capriotti, E., Fariselli, P. and Casadio, R. I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic Acids Res.* Oxford University Press (OUP). july 2005, vol. 33, Web Server issue, p. W306–10.

[13] Chen, C.-W., Lin, J. and Chu, Y.-W. IStable: off-the-shelf predictor integration for predicting protein stability changes. *BMC Bioinformatics*. Jan 2013, vol. 14, no. 2, p. S5. DOI: 10.1186/1471-2105-14-S2-S5. ISSN 1471-2105. Available at: https://doi.org/10.1186/1471-2105-14-S2-S5.

[14] Chen, Y., Lu, H., Zhang, N., Zhu, Z., Wang, S. et al. PremPS: Predicting the impact of missense mutations on protein stability. *PLoS Comput. Biol.* Public Library of Science (PLoS). december 2020, vol. 16, no. 12, p. e1008543.

[15] Cheng, J., Randall, A. and Baldi, P. Prediction of protein stability changes for single-site mutations using support vector machines. *Proteins*. Wiley. march 2006, vol. 62, no. 4, p. 1125–1132.

[16] Clark, J. *Electronic Spectroscopy*. 2022. The Beer-Lamber Law. Available at: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Spectroscopy/Electronic_Spectroscopy/Electronic_Spectroscopy_Basics/The_Beer-Lambert_Law.

[17] Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J. et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*. Oxford University Press. 2009, vol. 25, no. 11, p. 1422–1423.

[18] Connie, R., Robert, W., Vladimir, J., Jean, D., Jung, C. et al. *Biology*. OpenStax, 2016.

[19] Contessoto, V., Oliveira, V. Martins de, Fernandes, B., Slade, G. and Leite, V. TKSA-MC: A Web Server for rational mutation through the optimization of protein charge interactions. november 2017. DOI: 10.1101/221556.

[20] Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. M. et al. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *Journal of the American Chemical Society*. American Chemical Society. May 1995, vol. 117, no. 19, p. 5179–5197. DOI: 10.1021/ja00124a002. ISSN 0002-7863. Available at: https://doi.org/10.1021/ja00124a002.

[21] Dana, J. M., Gutmanas, A., Tyagi, N., Qi, G., O'Donovan, C. et al. SIFTS: updated Structure Integration with Function, Taxonomy and Sequences resource allows 40-fold increase in coverage of structure-based annotations for proteins. *Nucleic Acids Res.* Oxford University Press (OUP). january 2019, vol. 47, D1, p. D482–D489.

[22] Danev, R., Yanagisawa, H. and Kikkawa, M. Cryo-Electron Microscopy Methodology: Current Aspects and Future Directions. *Trends in Biochemical Sciences*. 2019, vol. 44, no. 10, p. 837–848. DOI: https://doi.org/10.1016/j.tibs.2019.04.008. ISSN 0968-0004. Available at: https://www.sciencedirect.com/science/article/pii/S096800041930088X.

[23] Dehouck, Y., Grosfils, A., Folch, B., Gilis, D., Bogaerts, P. et al. Fast and accurate predictions of protein stability changes upon mutations using statistical potentials and neural networks: PoPMuSiC-2.0. *Bioinformatics*. august 2009, vol. 25, no. 19, p. 2537–2543. DOI: 10.1093/bioinformatics/btp445. ISSN 1367-4803. Available at: https://doi.org/10.1093/bioinformatics/btp445.

[24] Esquivel, R., Molina Espiritu, M., Salas, F., Soriano Correa, C., Barrientos Salcedo, C. et al. Decoding the Building Blocks of Life from the Perspective of Quantum Information. In:. April 2013. DOI: 10.5772/55160. ISBN 978-953-51-1089-7.

[25] Fang, J. A critical review of five machine learning-based algorithms for predicting protein stability changes upon mutation. *Briefings in Bioinformatics*. july 2019, vol. 21, no. 4, p. 1285–1292. DOI: 10.1093/bib/bbz071. ISSN 1477-4054. Available at: https://doi.org/10.1093/bib/bbz071.

[26] Fersht, A. R. and Requena, Y. Free energies of hydrolysis of amides and peptides in aqueous solution at 25 degrees. *J Am Chem Soc*. Jul 1971, vol. 93, no. 14, p. 3499–3504. [DOI:10.1021/ja00743a034] [PubMed:5560469].

[27] Folkman, L., Stantic, B., Sattar, A. and Zhou, Y. EASE-MM: Sequence-Based Prediction of Mutation-Induced Stability Changes with Feature-Based Multiple Models. *Journal of Molecular Biology*. 2016, vol. 428, no. 6, p. 1394–1405. DOI: https://doi.org/10.1016/j.jmb.2016.01.012. ISSN 0022-2836. Available at: https://www.sciencedirect.com/science/article/pii/S0022283616000310.

[28] Frappier, V., Chartier, M. and Najmanovich, R. J. ENCoM server: exploring protein conformational space and the effect of mutations on protein function and stability. *Nucleic Acids Res.* Oxford University Press (OUP). july 2015, vol. 43, W1, p. W395–400.

[29] Frappier, V., Chartier, M. and Najmanovich, R. J. ENCoM server: exploring protein conformational space and the effect of mutations on protein function and

stability. *Nucleic Acids Research*. april 2015, vol. 43, W1, p. W395–W400. DOI: 10.1093/nar/gkv343. ISSN 0305-1048. Available at: https://doi.org/10.1093/nar/gkv343.

[30] Giollo, M., Martin, A. J., Walsh, I., Ferrari, C. and Tosatto, S. C. NeEMO: a method using residue interaction networks to improve prediction of protein stability upon mutation. *BMC Genomics*. May 2014, vol. 15, no. 4, p. S7. DOI: 10.1186/1471-2164-15-S4-S7. ISSN 1471-2164. Available at: https://doi.org/10.1186/1471-2164-15-S4-S7.

[31] Gopi, S., Devanshu, D., Krishna, P. and Naganathan, A. N. pStab: prediction of stable mutants, unfolding curves, stability maps and protein electrostatic frustration. *Bioinformatics*. october 2017, vol. 34, no. 5, p. 875–877. DOI: 10.1093/bioinformatics/btx697. ISSN 1367-4803. Available at: https://doi.org/10.1093/bioinformatics/btx697.

[32] Gregorio, N. E., Levine, M. Z. and Oza, J. P. A user's guide to cell-free protein synthesis. *Methods Protoc.* MDPI AG. march 2019, vol. 2, no. 1, p. 24.

[33] Gromiha, M. M. Chapter 6 - Protein Stability. In: Gromiha, M. M., ed. *Protein Bioinformatics*. Singapore: Academic Press, 2010, p. 209–245. DOI: https://doi.org/10.1016/B978-8-1312-2297-3.50006-0. ISBN 978-81-312-2297-3. Available at: https://www.sciencedirect.com/science/article/pii/B9788131222973500060.

[34] Havranek, J. J. and Harbury, P. B. Tanford-Kirkwood electrostatics for protein modeling. *Proc. Natl. Acad. Sci. U. S. A.* Proceedings of the National Academy of Sciences. september 1999, vol. 96, no. 20, p. 11145–11150.

[35] Hopkins, E., Sanvictores, T. and Sharma, S. Physiology, Acid Base Balance. In: *StatPearls*. Treasure Island (FL): StatPearls Publishing, January 2022.

[36] Hoppe, C. and Schomburg, D. Prediction of protein thermostability with a direction- and distance-dependent knowledge-based potential. *Protein Sci.* Wiley. october 2005, vol. 14, no. 10, p. 2682–2692.

[37] Ivakhnenko, A. G. Polynomial Theory of Complex Systems. *IEEE Transactions on Systems, Man, and Cybernetics*. 1971, SMC-1, no. 4, p. 364–378. DOI: 10.1109/TSMC.1971.4308320.

[38] Jubb, H. C., Higueruelo, A. P., Ochoa Montaño, B., Pitt, W. R., Ascher, D. B. et al. Arpeggio: A web server for calculating and visualising interatomic interactions in protein structures. *J. Mol. Biol.* february 2017, vol. 429, no. 3, p. 365–371.

[39] Khan, S. and Vihinen, M. Performance of protein stability predictors. *Human Mutation*. 2010, vol. 31, no. 6, p. 675–684. DOI: https://doi.org/10.1002/humu.21242. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.21242.

[40] Khatun, J., Khare, S. D. and Dokholyan, N. V. Can contact potentials reliably predict stability of proteins? *J. Mol. Biol.* Elsevier BV. march 2004, vol. 336, no. 5, p. 1223–1238.

[41] Laimer, J., Hofer, H., Fritz, M., Wegenkittl, S. and Lackner, P. MAESTRO - multi agent stability prediction upon point mutations. *BMC Bioinformatics*. Apr 2015, vol. 16, no. 1, p. 116. DOI: 10.1186/s12859-015-0548-6. ISSN 1471-2105. Available at: https://doi.org/10.1186/s12859-015-0548-6.

[42] Li, B., Yang, Y. T., Capra, J. A. and Gerstein, M. B. Predicting changes in protein thermodynamic stability upon point mutation with deep 3D convolutional neural networks. *PLoS Comput. Biol.* Public Library of Science (PLoS). november 2020, vol. 16, no. 11, p. e1008291.

[43] Li, G., Panday, S. K. and Alexov, E. SAAFEC-SEQ: A sequence-based method for predicting the effect of single point mutations on protein thermodynamic stability. *Int. J. Mol. Sci.* MDPI AG. january 2021, vol. 22, no. 2, p. 606.

[44] Lipman, D. J. and Pearson, W. R. Rapid and sensitive protein similarity searches. *Science*. American Association for the Advancement of Science (AAAS). march 1985, vol. 227, no. 4693, p. 1435–1441.

[45] Lo Conte, L., Ailey, B., Hubbard, T. J., Brenner, S. E., Murzin, A. G. et al. SCOP: a structural classification of proteins database. *Nucleic Acids Res.* Oxford University Press (OUP). january 2000, vol. 28, no. 1, p. 257–259.

[46] Magyar, C., Gromiha, M. M., Pujadas, G., Tusnády, G. E. and Simon, I. SRide: a server for identifying stabilizing residues in proteins. *Nucleic Acids Res.* Oxford University Press (OUP). july 2005, vol. 33, Web Server issue, p. W303–5.

[47] Mazurenko, S. Predicting protein stability and solubility changes upon mutations: data perspective. *ChemCatChem*. 2020, vol. 12, no. 22, p. 5590–5598. DOI: https://doi.org/10.1002/cctc.202000933. Available at: https://chemistry-europe.onlinelibrary.wiley.com/doi/abs/10.1002/cctc.202000933.

[48] Modarres, H. P., Mofrad, M. R. and Sanati Nezhad, A. Protein thermostability engineering. *RSC Adv.* The Royal Society of Chemistry. 2016, vol. 6, p. 115252–115270. DOI: 10.1039/C6RA16992A. Available at: http://dx.doi.org/10.1039/C6RA16992A.

[49] Montanucci, L., Capriotti, E., Frank, Y., Ben Tal, N. and Fariselli, P. DDGun: an untrained method for the prediction of protein stability changes upon single and multiple point variations. *BMC Bioinformatics*. Jul 2019, vol. 20, no. 14, p. 335. DOI: 10.1186/s12859-019-2923-1. ISSN 1471-2105. Available at: https://doi.org/10.1186/s12859-019-2923-1.

[50] Musil, M., Konegger, H., Hon, J., Bednar, D. and Damborsky, J. Computational Design of Stable and Soluble Biocatalysts. *ACS Catalysis*. 2019, vol. 9, no. 2, p. 1033–1054. DOI: 10.1021/acscatal.8b03613. Available at: https://doi.org/10.1021/acscatal.8b03613.

[51] Musil, M. *Computational Design of Stable Proteins*. Brno, CZ, 2021. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/phd-thesis/967/.

[52] Nair, P. S. and Vihinen, M. VariBench: A Benchmark Database for Variations. *Human Mutation*. 2013, vol. 34, no. 1, p. 42–49. DOI: https://doi.org/10.1002/humu.22204. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.22204.

[53] Nikam, R., Kulandaisamy, A., Harini, K., Sharma, D. and Gromiha, M. M. ProThermDB: thermodynamic database for proteins and mutants revisited after 15 years. *Nucleic Acids Research*. november 2020, vol. 49, D1, p. D420–D424. DOI: 10.1093/nar/gkaa1035. ISSN 0305-1048. Available at: https://doi.org/10.1093/nar/gkaa1035.

[54] Ophardt, C. E. *Virtual ChemBook*. Elmhurst college, 2003. Available at: http://chemistry.elmhurst.edu/vchembook/index.html.

[55] Pace, C. N., Fu, H., Lee Fryar, K., Landua, J., Trevino, S. R. et al. Contribution of hydrogen bonds to protein stability. *Protein Sci.* Wiley. may 2014, vol. 23, no. 5, p. 652–661.

[56] Pancotti, C., Benevenuta, S., Birolo, G., Alberini, V., Repetto, V. et al. Predicting protein stability changes upon single-point mutation: a thorough comparison of the available tools on a new dataset. *Briefings in Bioinformatics*. january 2022, vol. 23, no. 2, p. bbab555. DOI: 10.1093/bib/bbab555. ISSN 1477-4054. Available at: https://doi.org/10.1093/bib/bbab555.

[57] Parthiban, V., Gromiha, M. M. and Schomburg, D. CUPSAT: prediction of protein stability upon point mutations. *Nucleic Acids Res.* Oxford University Press (OUP). july 2006, vol. 34, Web Server issue, p. W239–42.

[58] Pires, D. E. V., Ascher, D. B. and Blundell, T. L. DUET: a server for predicting effects of mutations on protein stability using an integrated computational approach. *Nucleic Acids Res.* Oxford University Press (OUP). july 2014, vol. 42, Web Server issue, p. W314–9.

[59] Pires, D. E. V., Ascher, D. B. and Blundell, T. L. mCSM: predicting the effects of mutations in proteins using graph-based signatures. *Bioinformatics*. Oxford University Press (OUP). february 2014, vol. 30, no. 3, p. 335–342.

[60] Polizzi, K. M., Bommarius, A. S., Broering, J. M. and Chaparro Riggers, J. F. Stability of biocatalysts. *Curr. Opin. Chem. Biol.* Elsevier BV. april 2007, vol. 11, no. 2, p. 220–225.

[61] Pommié, C., Levadoux, S., Sabatier, R., Lefranc, G. and Lefranc, M.-P. IMGT standardized criteria for statistical analysis of immunoglobulin V-REGION amino acid properties. *J. Mol. Recognit.* Wiley. january 2004, vol. 17, no. 1, p. 17–32.

[62] Ponnuswamy, P. and Gromiha, M. On the Conformational Stability of Folded Proteins. *Journal of Theoretical Biology*. 1994, vol. 166, no. 1, p. 63–74. DOI: https://doi.org/10.1006/jtbi.1994.1005. ISSN 0022-5193. Available at: https://www.sciencedirect.com/science/article/pii/S0022519384710058.

[63] Privalov, P. L. and Gill, S. J. Stability of protein structure and hydrophobic interaction. *Adv Protein Chem*. 1988, vol. 39, p. 191–234. [DOI:10.1016/s0065-3233(08)60377-0] [PubMed:3072868].

[64] Pucci, F., Bernaerts, K. V., Kwasigroch, J. M. and Rooman, M. Quantification of biases in predictions of protein stability changes upon mutations. *Bioinformatics*. Oxford University Press (OUP). november 2018, vol. 34, no. 21, p. 3659–3665.

[65] Pucci, F., Bernaerts, K. V., Kwasigroch, J. M. and Rooman, M. Quantification of biases in predictions of protein stability changes upon mutations. *Bioinformatics*. april 2018, vol. 34, no. 21, p. 3659–3665. DOI: 10.1093/bioinformatics/bty348. ISSN 1367-4803. Available at: https://doi.org/10.1093/bioinformatics/bty348.

[66] Pucci, F., Bourgeas, R. and Rooman, M. Predicting protein thermal stability changes upon point mutations using statistical potentials: Introducing HoTMuSiC. *Scientific Reports*. Mar 2016, vol. 6, no. 1, p. 23257. DOI: 10.1038/srep23257. ISSN 2045-2322. Available at: https://doi.org/10.1038/srep23257.

[67] Reddy, M. K. Amino acid. In: *Encyclopedia Britannica*. November 2022.

[68] Refaeilzadeh, P., Tang, L. and Liu, H. Cross-Validation. In: LIU, L. and ÖZSU, M. T., ed. *Encyclopedia of Database Systems*. Boston, MA: Springer US, 2009, p. 532–538. DOI: 10.1007/978-0-387-39940-9_565. ISBN 978-0-387-39940-9. Available at: https://doi.org/10.1007/978-0-387-39940-9_565.

[69] Revard, B. C., Tipton, W. W. and Hennig, R. G. Structure and Stability Prediction of Compounds with Evolutionary Algorithms. In: Atahan Evrenk, S. and Aspuru Guzik, A., ed. *Prediction and Calculation of Crystal Structures: Methods and Applications*. Cham: Springer International Publishing, 2014, p. 181–222. DOI: 10.1007/128_2013_489. ISBN 978-3-319-05774-3. Available at: https://doi.org/10.1007/128_2013_489.

[70] Rodrigues, C. H. M., Pires, D. E. V. and Ascher, D. B. DynaMut2: Assessing changes in stability and flexibility upon single and multiple point missense mutations. *Protein Sci.* Wiley. january 2021, vol. 30, no. 1, p. 60–69.

[71] Sali, D., Bycroft, M. and Fersht, A. R. Stabilization of protein structure by interaction of alpha-helix dipole with a charged side chain. *Nature*. Oct 1988, vol. 335, no. 6192, p. 740–743. [DOI:10.1038/335740a0] [PubMed:3173493].

[72] Samaga, Y. B. L., Raghunathan, S. and Priyakumar, U. D. SCONES: Self-Consistent Neural Network for Protein Stability Prediction Upon Mutation. *The Journal of Physical Chemistry B*. American Chemical Society. Sep 2021, vol. 125, no. 38, p. 10657–10671. DOI: 10.1021/acs.jpcb.1c04913. ISSN 1520-6106. Available at: https://doi.org/10.1021/acs.jpcb.1c04913.

[73] Sanavia, T., Birolo, G., Montanucci, L., Turina, P., Capriotti, E. et al. Limitations and challenges in protein stability prediction upon genome variations: towards future applications in precision medicine. *Comput. Struct. Biotechnol. J.* Elsevier BV. july 2020, vol. 18, p. 1968–1979.

[74] Savojardo, C., Fariselli, P., Martelli, P. L. and Casadio, R. INPS-MD: a web server to predict stability of protein variants from sequence and structure. *Bioinformatics*. april 2016, vol. 32, no. 16, p. 2542–2544. DOI: 10.1093/bioinformatics/btw192. ISSN 1367-4803. Available at: https://doi.org/10.1093/bioinformatics/btw192.

[75] Savojardo, C., Martelli, P. L., Casadio, R. and Fariselli, P. On the critical review of five machine learning-based algorithms for predicting protein stability changes upon mutation. *Briefings in Bioinformatics*. december 2019, vol. 22, no. 1, p. 601–603. DOI: 10.1093/bib/bbz168. ISSN 1477-4054. Available at: https://doi.org/10.1093/bib/bbz168.

[76] Schymkowitz, J., Borg, J., Stricher, F., Nys, R., Rousseau, F. et al. The FoldX web server: an online force field. *Nucleic Acids Res.* Oxford University Press (OUP). july 2005, vol. 33, Web Server issue, p. W382–8.

[77] Shafee, T. *Protein structure (full)*. 2016. File: Protein structure (full).png. Available at: https://foundation.wikimedia.org/wiki/File:Protein_structure_%28full%29.png.

[78] Siloto, R. M. and Weselake, R. J. Site saturation mutagenesis: Methods and applications in protein engineering. *Biocatalysis and Agricultural Biotechnology*. 2012, vol. 1, no. 3, p. 181–189. DOI: https://doi.org/10.1016/j.bcab.2012.03.010. ISSN 1878-8181. Available at: https://www.sciencedirect.com/science/article/pii/S1878818112000734.

[79] Singh, M. K. and Singh, A. Chapter 14 - Nuclear magnetic resonance spectroscopy. In: Singh, M. K. and Singh, A., ed. *Characterization of Polymers and Fibres*. Woodhead Publishing, 2022, p. 321–339. The Textile Institute Book Series. DOI: https://doi.org/10.1016/B978-0-12-823986-5.00011-7. ISBN 978-0-12-823986-5. Available at: https://www.sciencedirect.com/science/article/pii/B9780128239865000117.

[80] Singh, R. K., Tropsha, A. and Vaisman, I. I. Delaunay tessellation of proteins: four body nearest-neighbor propensities of amino acid residues. *J. Comput. Biol.* Mary Ann Liebert Inc. 1996, vol. 3, no. 2, p. 213–221.

[81] Smyth, M. S. and Martin, J. H. X ray crystallography. *Mol. Pathol.* BMJ. february 2000, vol. 53, no. 1, p. 8–14.

[82] Soderberg, T. and BCcampus. *Organic Chemistry with a Biological Emphasis*. Timothy Soderberg, 2012. Online access: BCampus.ca BC Open Textbook Collection. Available at: https://books.google.cz/books?id=M0vQoAEACAAJ.

[83] Stauffer, E., Dolan, J. A. and Newman, R. CHAPTER 3 - Review of Basic Organic Chemistry. In: Stauffer, E., Dolan, J. A. and Newman, R., ed. *Fire Debris Analysis*. Burlington: Academic Press, 2008, p. 49–83. DOI: https://doi.org/10.1016/B978-012663971-1.50007-5. ISBN 978-0-12-663971-1. Available at: https://www.sciencedirect.com/science/article/pii/B9780126639711500075.

[84] Stourac, J., Dubrava, J., Musil, M., Horackova, J., Damborsky, J. et al. FireProtDB: database of manually curated protein stability data. *Nucleic Acids Research*. november 2020, vol. 49, D1, p. D319–D324. DOI: 10.1093/nar/gkaa981. ISSN 0305-1048. Available at: https://doi.org/10.1093/nar/gkaa981.

[85] Suzek, B. E., Wang, Y., Huang, H., McGarvey, P. B., Wu, C. H. et al. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity

searches. *Bioinformatics*. Oxford University Press (OUP). march 2015, vol. 31, no. 6, p. 926–932.

[86] Teng, S., Srivastava, A. K. and Wang, L. Sequence feature-based prediction of protein stability changes upon amino acid substitutions. *BMC Genomics*. Nov 2010, vol. 11, no. 2, p. S5. DOI: 10.1186/1471-2164-11-S2-S5. ISSN 1471-2164. Available at: https://doi.org/10.1186/1471-2164-11-S2-S5.

[87] Thornton, J. Disulphide bridges in globular proteins. *Journal of Molecular Biology*. 1981, vol. 151, no. 2, p. 261–287. DOI: https://doi.org/10.1016/0022-2836(81)90515-5. ISSN 0022-2836. Available at: https://www.sciencedirect.com/science/article/pii/0022283681905155.

[88] Usmanova, D. R., Bogatyreva, N. S., Ariño Bernad, J., Eremina, A. A., Gorshkova, A. A. et al. Self-consistency test reveals systematic bias in programs for prediction change of stability upon mutation. *Bioinformatics*. Oxford University Press (OUP). november 2018, vol. 34, no. 21, p. 3653–3658.

[89] Uversky, V. N. Intrinsically disordered proteins in overcrowded milieu: Membrane-less organelles, phase separation, and intrinsic disorder. *Curr. Opin. Struct. Biol.* june 2017, vol. 44, p. 18–30.

[90] Van Rossum, G. and Drake, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

[91] Velecký, J., Berezný, M., Musil, M., Damborsky, J., Bednar, D. et al. *The BenchStab dataset: a dataset for benchmarking mutational predictors for stability*. Zenodo, 2024.

[92] Wehrl, A. General properties of entropy. *Rev. Mod. Phys.* American Physical Society. Apr 1978, vol. 50, p. 221–260. DOI: 10.1103/RevModPhys.50.221. Available at: https://link.aps.org/doi/10.1103/RevModPhys.50.221.

[93] Witvliet, D. K., Strokach, A., Giraldo Forero, A. F., Teyra, J., Colak, R. et al. ELASPIC web-server: proteome-wide structure-based prediction of mutation effects on protein stability and binding affinity. *Bioinformatics*. january 2016, vol. 32, no. 10, p. 1589–1591. DOI: 10.1093/bioinformatics/btw031. ISSN 1367-4803. Available at: https://doi.org/10.1093/bioinformatics/btw031.

[94] Worth, C. L., Preissner, R. and Blundell, T. L. SDM–a server for predicting effects of mutations on protein stability and malfunction. *Nucleic Acids Res.* Oxford University Press (OUP). july 2011, vol. 39, Web Server issue, p. W215–22.

[95] wwPDB consortium. Protein Data Bank: the single global archive for 3D macromolecular structure data. *Nucleic Acids Res.* Oxford University Press (OUP). january 2019, vol. 47, D1, p. D520–D528.

[96] Xiang, X. and Liu, H. IDPM: An online database for ion distribution in protein molecules. *BMC bioinformatics*. march 2018, vol. 19, p. 102. DOI: 10.1186/s12859-018-2110-9.

[97] Yang, Y., Urolagin, S., Niroula, A., Ding, X., Shen, B. et al. PON-tstab: Protein Variant Stability Predictor. Importance of Training Data Quality. *International Journal of Molecular Sciences*. 2018, vol. 19, no. 4. DOI: 10.3390/ijms19041009. ISSN 1422-0067. Available at: https://www.mdpi.com/1422-0067/19/4/1009.

[98] Yin, S., Ding, F. and Dokholyan, N. V. Eris: an automated estimator of protein stability. *Nature Methods*. Jun 2007, vol. 4, no. 6, p. 466–467. DOI: 10.1038/nmeth0607-466. ISSN 1548-7105. Available at: https://doi.org/10.1038/nmeth0607-466.

[99] Zhou, Y., Pan, Q., Pires, D. E. V., Rodrigues, C. M. and Ascher, D. DDMut: predicting effects of mutations on protein stability using deep learning. *Nucleic Acids Research*. june 2023, vol. 51, W1, p. W122–W128. DOI: 10.1093/nar/gkad472. ISSN 0305-1048. Available at: https://doi.org/10.1093/nar/gkad472.

Appendix A

Excluded datasets

Predictor/Dataset	S2648	S1603	P53	S350	S1765	S1538	S1948	S1925	S1615	S2298	S4022	S9028
CUPSat	×	✓	×	×	×	✓	×	×	×	×	×	×
DUET	✓	×	×	×	×	×	×	×	×	×	×	×
I-Mutant2.0	×	×	×	×	×	×	✓	×	×	×	×	×
iStable	✓	×	×	×	×	×	✓	×	×	×	×	×
MUpro	×	×	×	×	×	×	×	×	✓	×	×	×
DynaMut2	✓	×	×	✓	×	×	×	×	×	✓	✓	×
DDMut	×	×	×	×	×	×	×	×	×	×	×	✓
INPS	✓	×	✓	×	×	×	×	×	×	×	×	×
Maestro	✓	×	×	✓	✓	×	×	✓	×	×	×	×
mCSM	✓	×	×	✓	×	×	×	✓	×	×	×	×
PoPMuSiC	✓	×	×	×	×	×	×	×	×	×	×	×
PremPS	✓	×	×	×	×	×	×	×	×	×	×	×
SDM 2.0	✓	×	×	✓	×	×	×	×	×	×	×	×
SAAFEC-SEQ	✓	×	✓	✓	×	×	×	×	×	×	×	×
AutoMute	N/A, extracted from FireProtDB											
I-Mutant3.0	N/A, extracted from FireProtDB											
DDGun	Untrained											

Table A.1: List of training (machine learning), or training and validation (force-fields) datasets used during the development of the benchmarked predictors. They were all excluded from our benchmark dataset based on sequence similarity (using UniRef50 clusters of the UniProt release 2023-01).

Appendix B

ROC curves

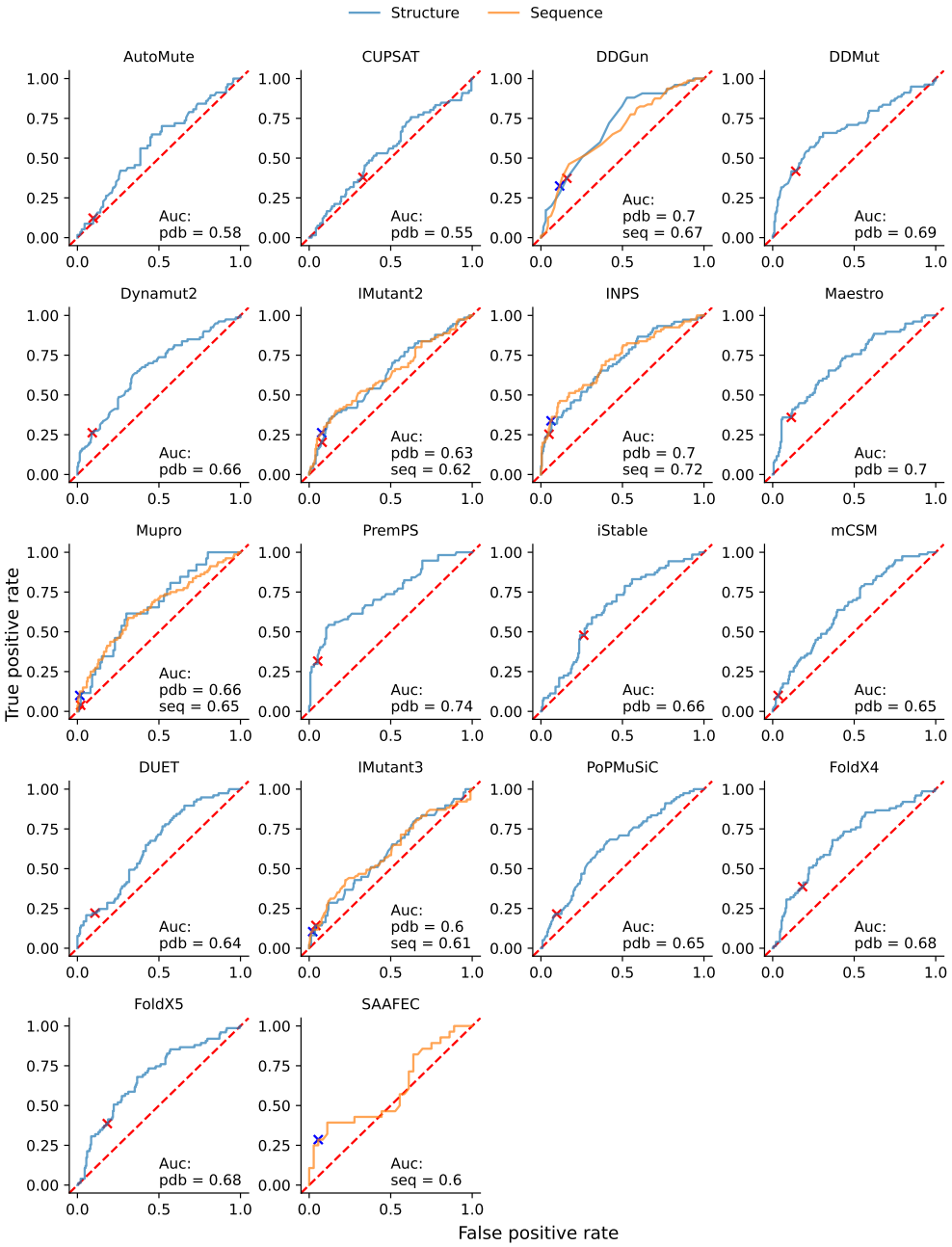


Figure B.1: **ROC curves depicting the relationship between false positive rate and true positive rate** (sensitivity/recall) at different thresholds to classify mutations as stabilizing or destabilizing. The blue curve represents the structural mode of the predictor, the orange curve is its sequential mode, and the red dashed line is the random predictor. The default threshold (0 kcal/mol) is marked with a red or blue cross for structural or sequential modes respectively.

Appendix C

PR curves

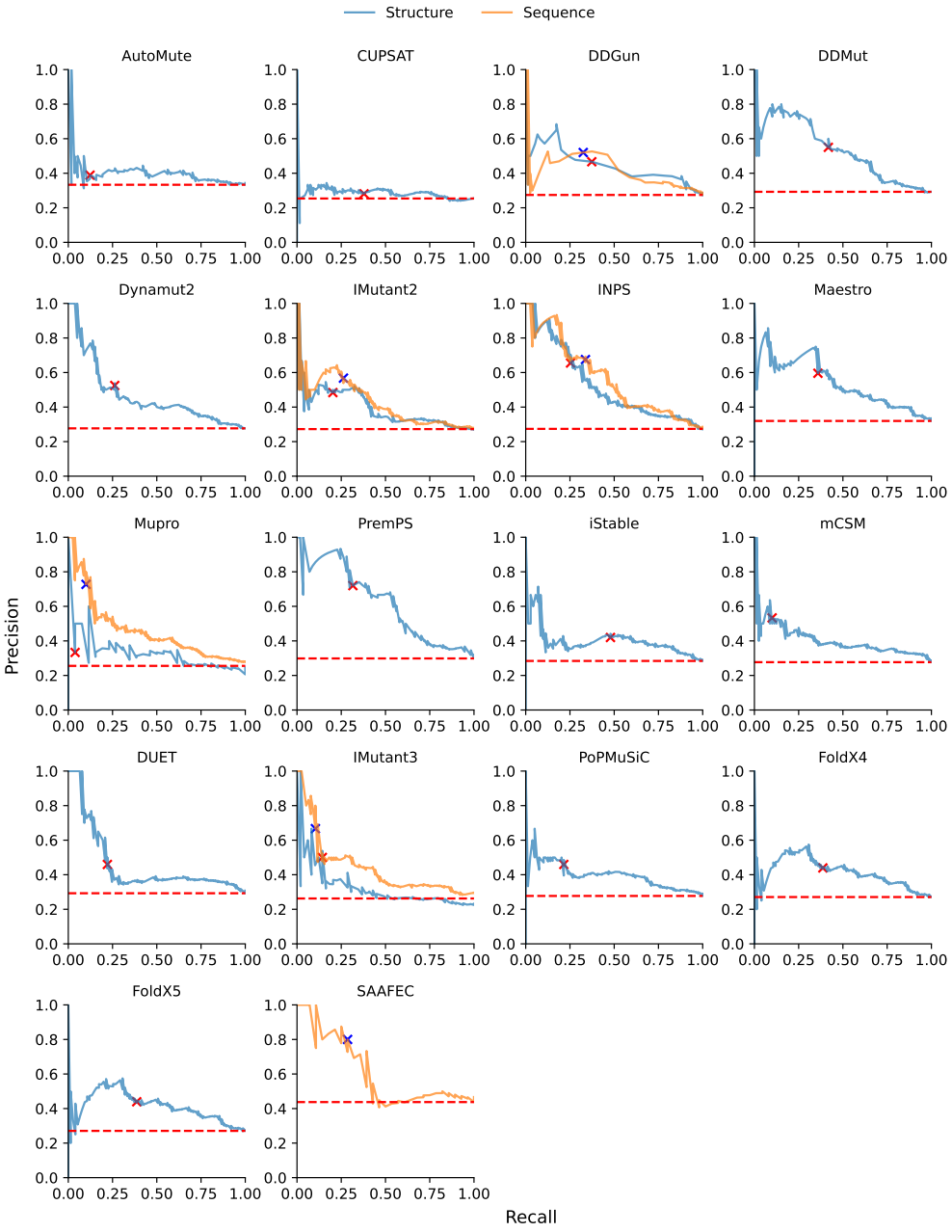


Figure C.1: **Precision-recall curves depicting the relationship between precision and recall** at different thresholds to classify mutations as stabilizing or destabilizing. The blue curve represents the structural mode of the predictor, the orange curve is its sequential mode, and the red dashed line is the random predictor adjusted to the pool of data successfully processed by the given tool. The default threshold (0 kcal/mol) is marked with a red or blue cross for structural or sequential modes respectively.

Appendix D

Kernel density estimate

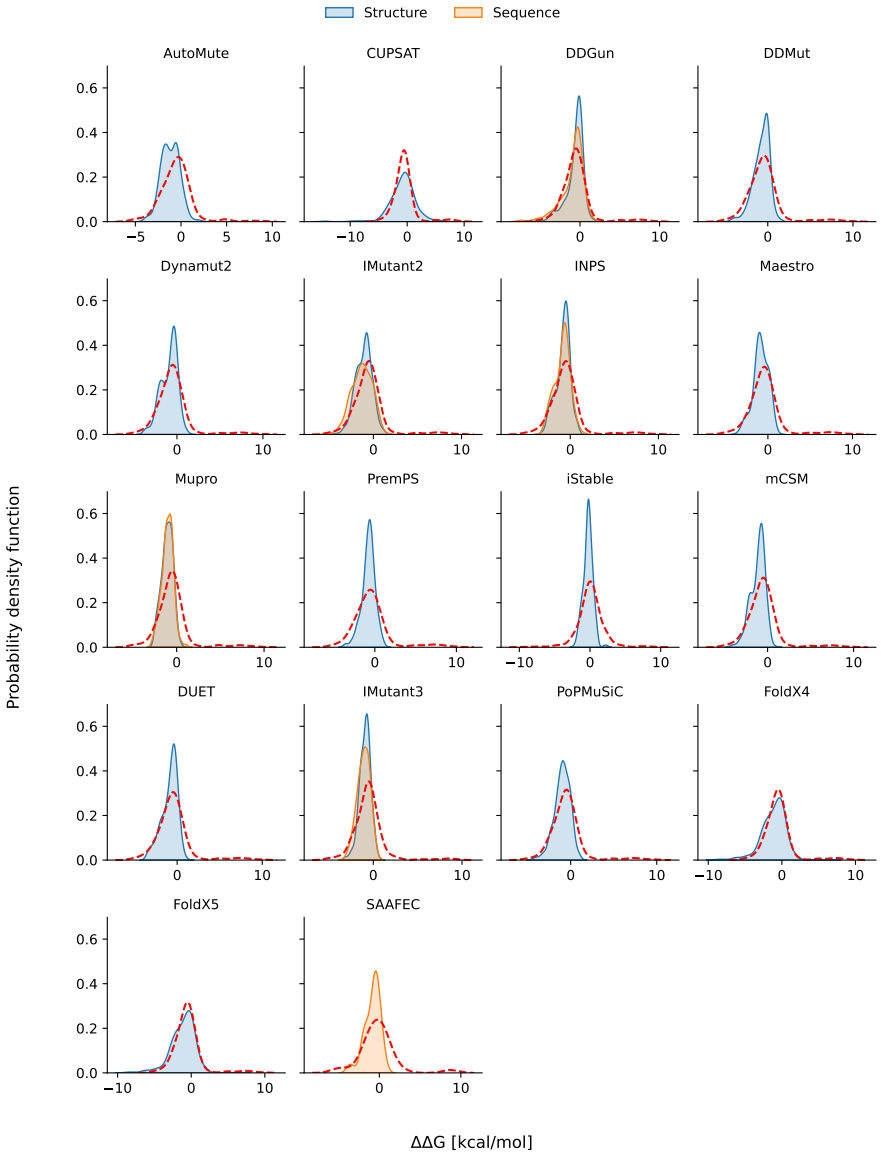


Figure D.1: **Kernel density estimate plot depicting the predicted ΔΔ G distributions per predictor and input type**. Orange represents sequential predictors, while blue represents tools predicting from structural inputs. The red dashed line shows the ΔΔ G distribution of a subset of ground truth labels belonging to successfully predicted data points. Most of the tools displayed a substantial shift of their distributions leftwards to their respective ground truth distribution, i.e. pointing on a bias towards destabilizing predictions (negative ΔΔ G values).

Appendix E

Balanced accuracy heat maps

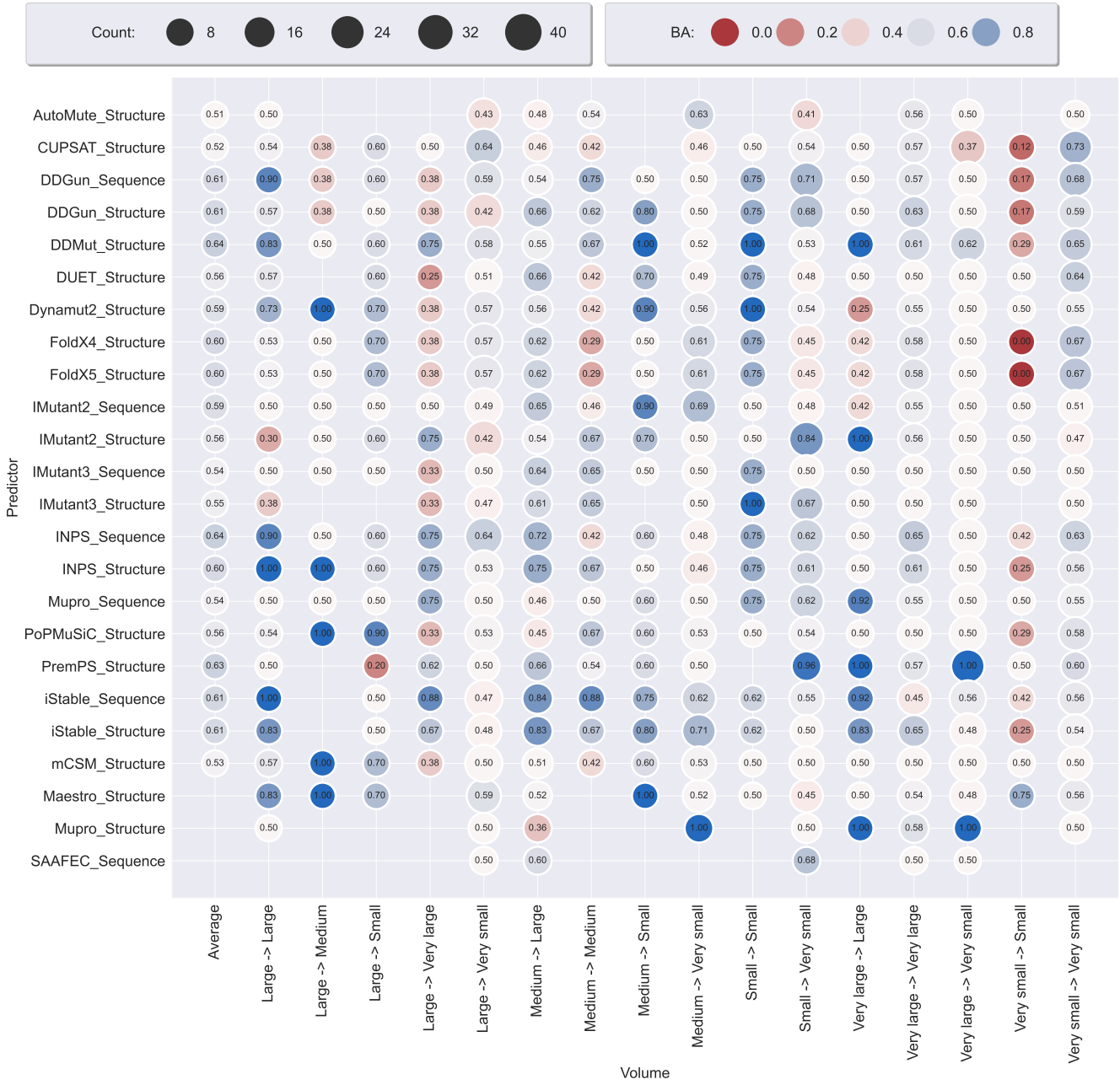


Figure E.1: Average balanced accuracy (BA) based on the size of the substituted amino acids ($size_{WT}$ -> $size_{MUT}$), per predictor/type combination. The color of the circle determines the balanced accuracy value, while the size of the circle number of datapoints belonging to respective substitution/predictor combination. Each combination had to have at least 5 successful predictions to be present in the graph.

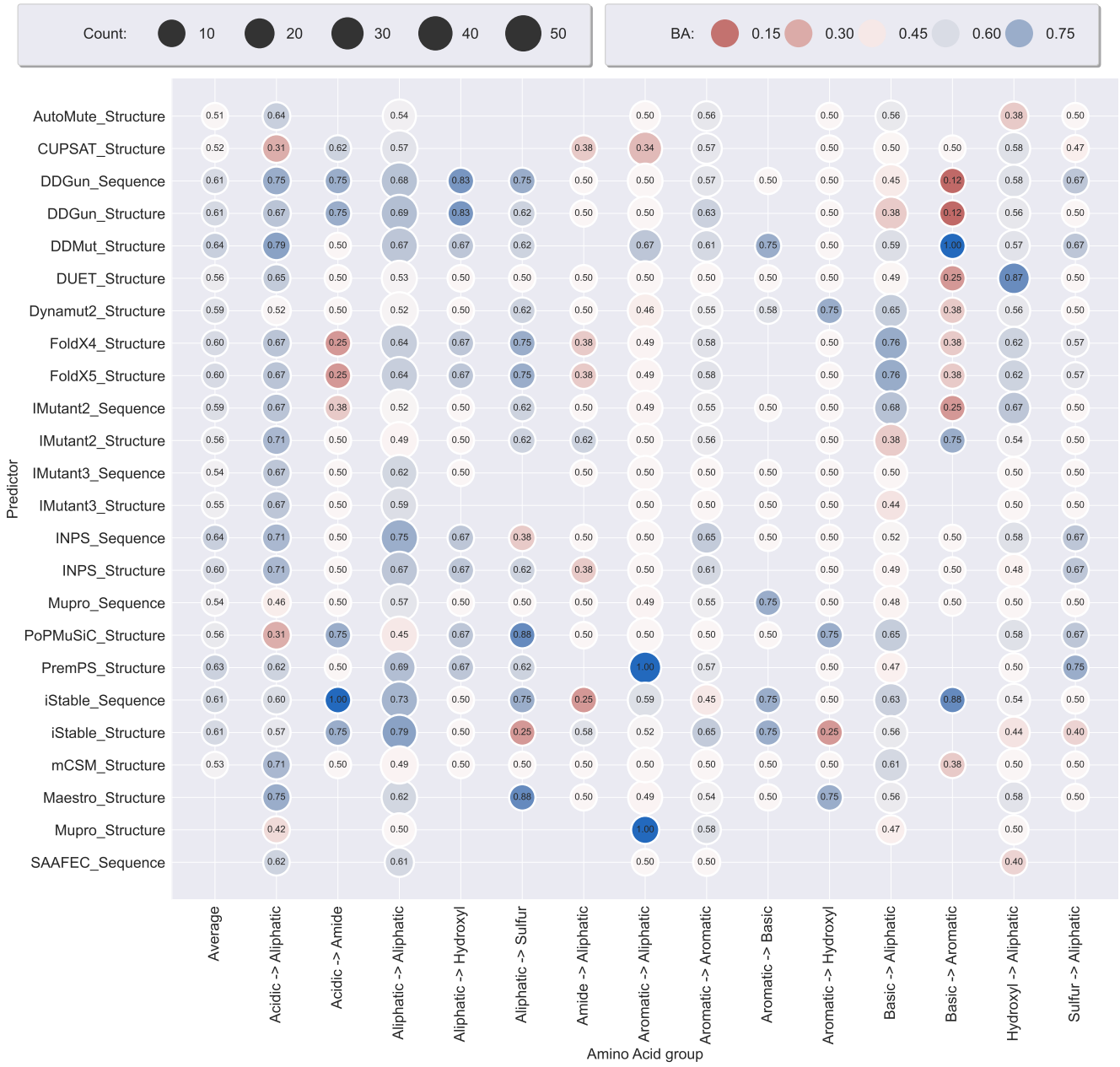


Figure E.2: Average balanced accuracy (BA) based on the chemical properties of the substituted amino acids ($chem_{WT}$ -> $chem_{MUT}$), per predictor/type combination. The color of the circle determines the balanced accuracy value, while the size of the circle number of datapoints belonging to respective substitution/predictor combination. Each combination had to have at least 5 successful predictions to be present in the graph.

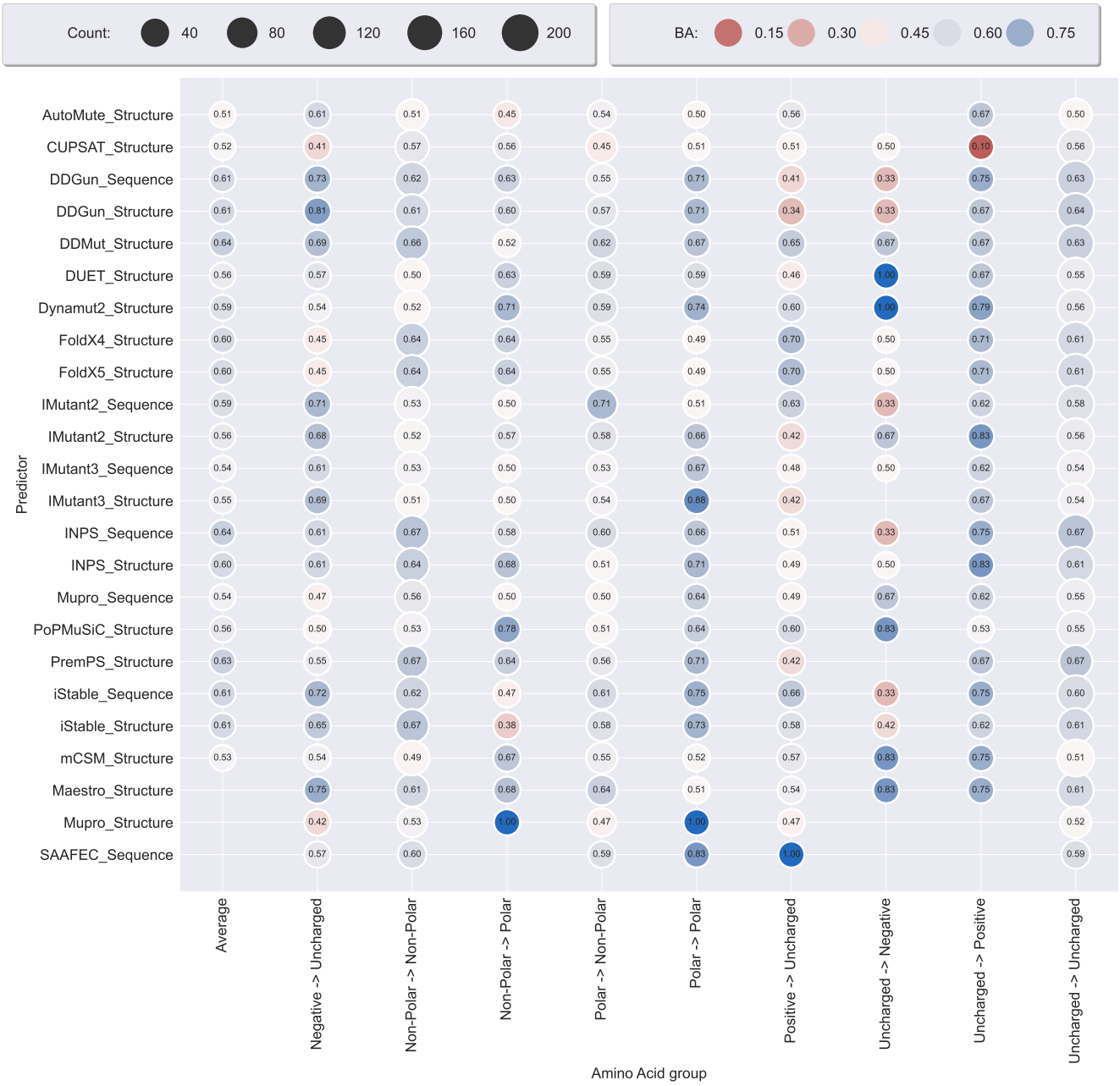


Figure E.3: Average balanced accuracy (BA) based on the polarity, hydropathy, and charge properties of the substituted amino acids ($prop_{WT}$ -> $prop_{MUT}$), per predictor/-type combination. The color of the circle determines the balanced accuracy value, while the size of the circle number of datapoints belonging to respective substitution/predictor combination. Each combination had to have at least 5 successful predictions to be present in the graph.

Appendix F

Mean absolute error heat maps

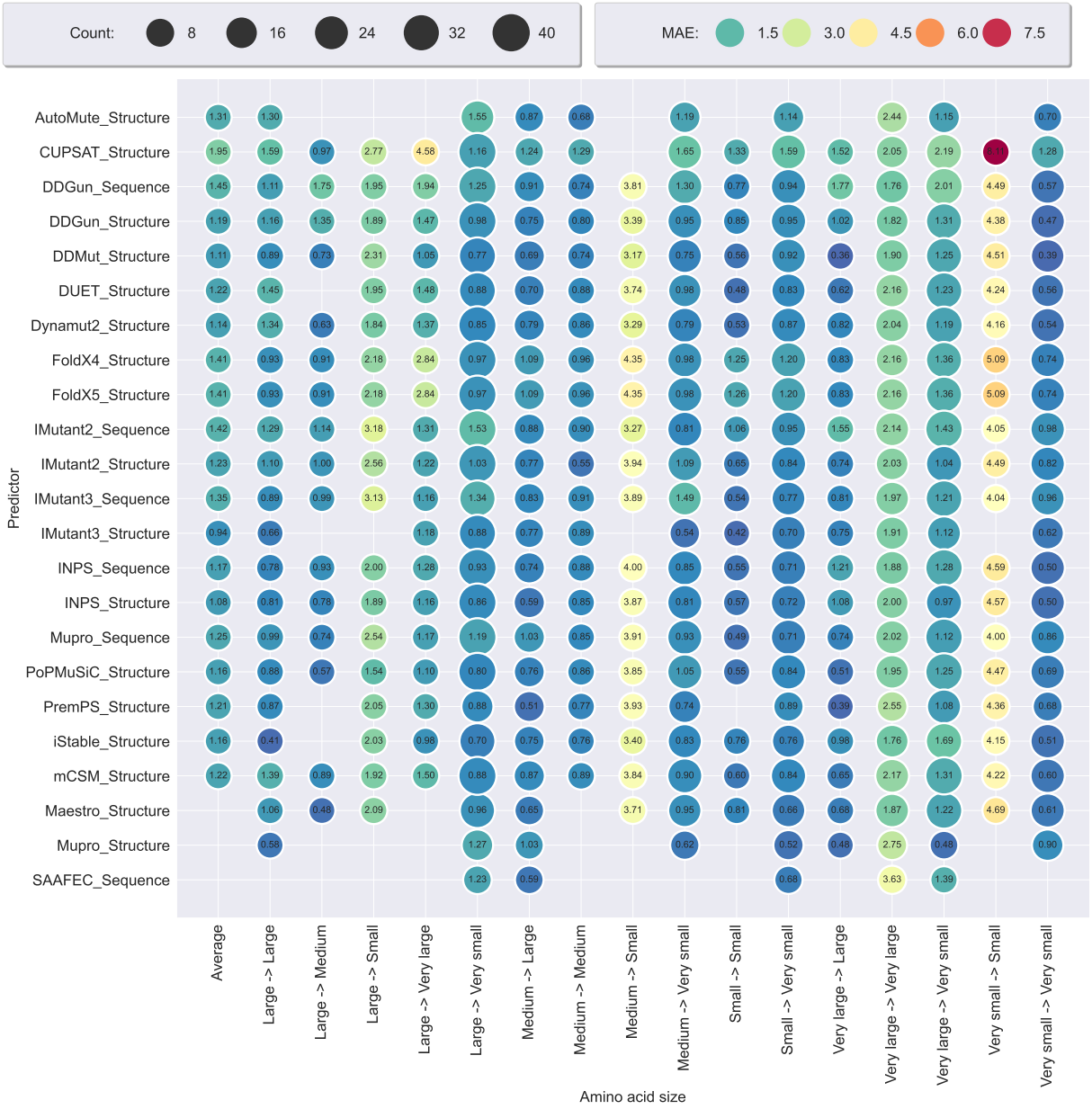


Figure F.1: Average mean absolute error (MAE) based on the size of the substituted amino acids ($size_{WT}$ -> $size_{MUT}$), per predictor/type combination. The color of the circle determines the mean absolute error value, while the size of the circle number of datapoints belonging to respective substitution/predictor combination. Each combination had to have at least 5 successful predictions to be present in the graph.

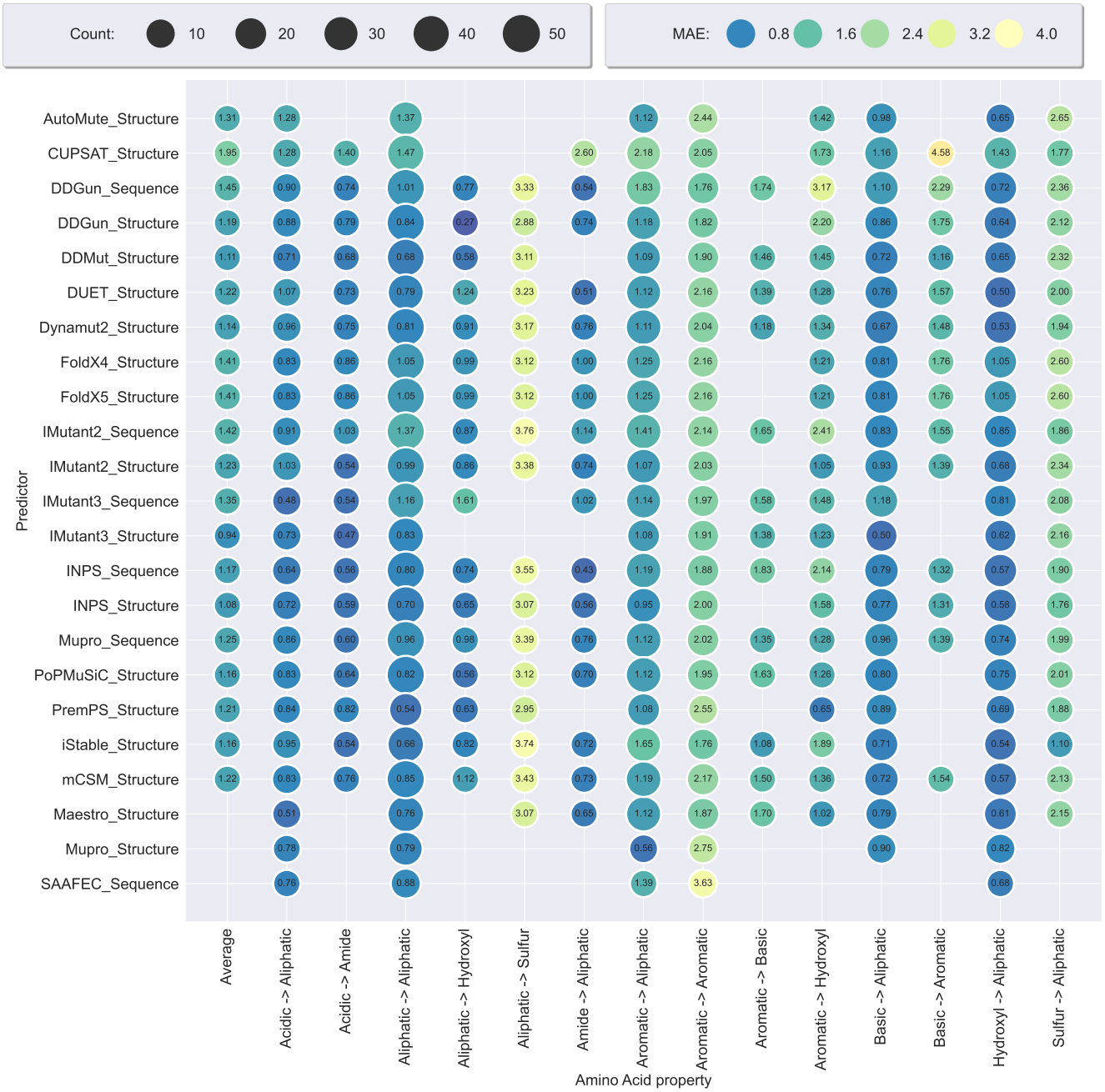


Figure F.2: Average mean absolute error (MAE) based on the chemical properties of the substituted amino acids ($chem_{WT}$ -> $chem_{MUT}$), per predictor/type combination. The color of the circle determines the mean absolute error value, while the size of the circle number of datapoints belonging to respective substitution/predictor combination. Each combination had to have at least 5 successful predictions to be present in the graph.

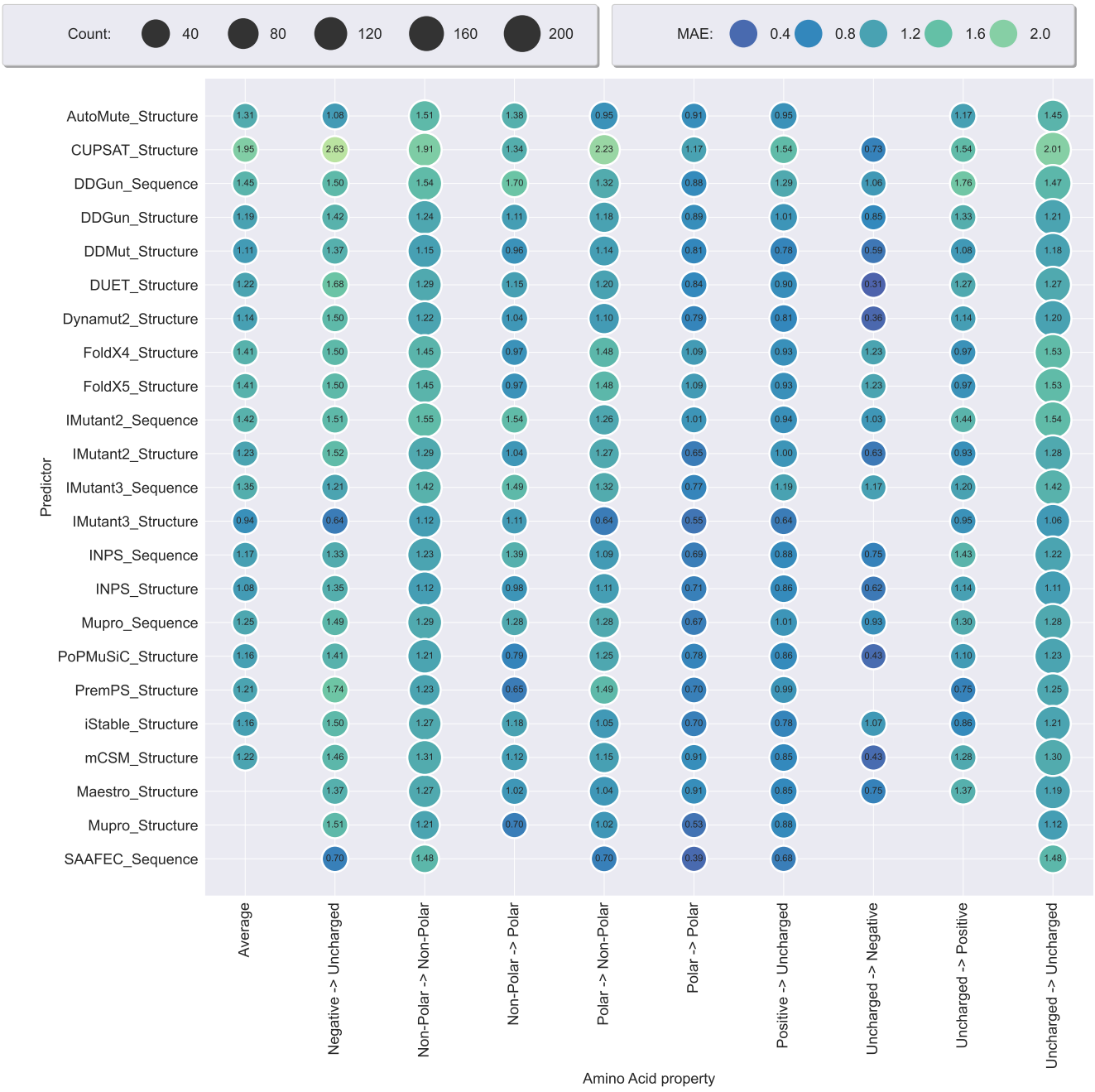


Figure F.3: Average mean absolute error (MAE) based on the polarity, hydropathy, and charge properties of the substituted amino acids ($prop_{WT}$ -> $prop_{MUT}$), per predictor/-type combination. The color of the circle determines the mean absolute error value, while the size of the circle number of datapoints belonging to respective substitution/predictor combination. Each combination had to have at least 5 successful predictions to be present in the graph.

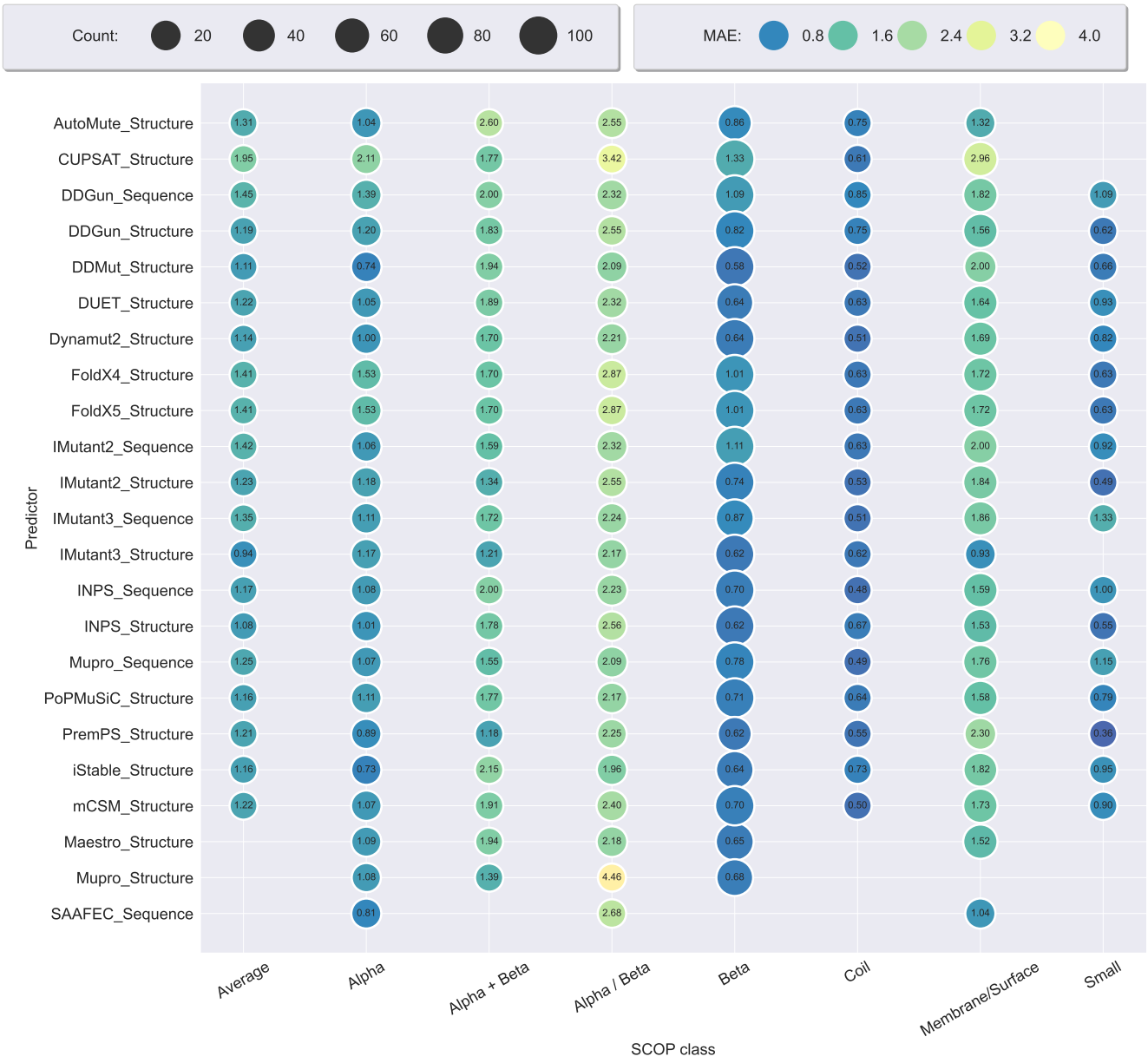


Figure F.4: Average mean absolute error (MAE) based on protein SCOP [45] class. The color of the circle determines the mean absolute error value, while the size of the circle number of datapoints belonging to respective substitution/predictor combination. Each combination had to have at least 5 successful predictions to be present in the graph.

Appendix G

CLI usage hint

The application is freely available on GitHub.

```
usage: benchstab.py [-h] [--include INCLUDE [INCLUDE ...]]
                    [--exclude EXCLUDE [EXCLUDE ...]]
                    [--pred-type PRED_TYPE [PRED_TYPE ...]]
                    [--config CONFIG] [--source SOURCE]
                    [--output OUTPUT] [--verbose]
                    [--quiet] [--list-predictors]
                    [--permissive] [--dry-run] [--skip-header]

Multi-Client application for efficient acquisition
of thermodynamical values from protein stability prediction servers.

optional arguments:
  -h, --help show this help message and exit
  --include INCLUDE [INCLUDE ...], -i INCLUDE [INCLUDE ...]
                        allows the user to specify a subset of predictors,
                        separated by a comma, from which the predictions
                        will be acquired.
  --exclude EXCLUDE [EXCLUDE ...], -e EXCLUDE [EXCLUDE ...]
                        Subset of predictors, separated by space,
                        excluded in the process of acquiring predictions.
                        If both include and exclude are supplied,
                        exclude will be ignored.
  --pred-type PRED_TYPE [PRED_TYPE ...], -t PRED_TYPE [PRED_TYPE ...]
                        Specifies which variations
                        of predictors should be used.
                        Available options: sequence, structure.
  --config CONFIG, -c CONFIG
                        Config file, with predictor parameters
                        in .json format. For exact structure,
                        please refer to the documentation.
  --source SOURCE, -s SOURCE
                        Mutation file, where has to be in required format:
                        '(pdb_id|fasta_file|pdb_file) MUT CHAIN'.
```

```
--output OUTPUT, -o OUTPUT
                    File path, where folder with results will be created.
                    If not supplied, application will not attempt to
                    store the prediction results.
--verbose, -v
                    Prints additional information about
                    the process of acquiring predictions.
--quiet, -q
                    Suppresses all output except for errors.
                    Overrides --verbose.
--list-predictors, -l
                    List all available predictors.
--permissive, -p
                    Performs a permissive run,
                    where errors are ignored and application
                    will attempt to acquire predictions
                    from all available predictors.
--dry-run Performs a dry run,
                    without actually sending any requests to the servers.
--skip-header
                    Skip header in input file.
```

Appendix H

Example of predictor configuration file

```
{
    "username": "email@email.com",
    "password": "randompassword",
    "wait_interval": 60,
    "batch_size": 20,
    "max_retries": 600,
    "popmusic": {
        "batch_size": 5,
        "max_retries": 100,
        "password": "test123"
    },
    "deepddg": {
        "batch_size": 5
    },
    "ddgun": {
        "batch_size": 5
    },
    "premps": {
        "batch_size": 5,
        "pdb_model": "bio"
    },
    "saafec": {
        "batch_size": 5
    },
    "inps": {
        "batch_size": 5
    },
    "dynamut2": {
        "batch_size": 5
    }
}
```