BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

PLATFORM FOR BIOLOGICAL SEQUENCE ANALYSIS USING MACHINE LEARNING

PLATFORMA PRO ANALÝZU BIOLOGICKÝCH SEKVENCÍ S VYUŽITÍM STROJOVÉHO UČENÍ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **DÁVID LACKO**
AUTOR PRÁCE

SUPERVISOR **Ing. TOMÁŠ MARTÍNEK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Lacko Dávid**
Program: Informační technologie
Název: **Platforma pro analýzu biologických sekvencí s využitím strojového učení**
Platform for Biological Sequence Analysis Using Machine Learning
Kategorie: Bioinformatika
Zadání:

1. Study the basic principles of machine learning.
2. Learn about the field of protein engineering and the tasks to which machine learning has been successfully applied.
3. Analyze these tasks, identify the common parts, and design a flexible platform for solving them.
4. Implement the proposed platform and validate its functionality on selected examples from the field of protein engineering.
5. Evaluate the results obtained and discuss the future work.

Literatura:

- According to instructions of the supervisor.

Pro udělení zápočtu za první semestr je požadováno:

- Fulfilment of items 1 to 3 of the assignment.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Martínek Tomáš, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 29. října 2021

Abstract

Machine learning has many active areas and one of them is protein characterisation since experimental annotation is usually costly and time-consuming, and many datasets suitable for training predictors are currently being published. One of the recent methods, called innov'SAR, combines the Fourier transform with partial linear regression and has been used in several protein engineering applications. However, the code for the method is not freely available and the method itself was not statistically verified. The goal of this thesis is to address these limitations, implement and extend the method using Python language in an easy-to-use platform that allows training and testing of the models. The extensions include parallelization, Spearman scoring function and aligned sequence input. The statistical significance testing is also performed to verify the impact of the found dependencies between input sequences and properties of the proteins. The method proved to be statistically significant with strong dependencies found between inputs and outputs. Two newly collected halalkane dehalogenase datasets were used to train models and they have cross validation scores of $Q^2 = 0.54$ and $Q^2 = 0.77$ with almost double the improvement over the baseline models. Created models allow filtering of large sequence databases and scanning for potential improvements in the protein properties.

Abstrakt

Strojové učenie má veľa aktívnych odvetví a jedným z nich je charakterizácia proteínov pretože experimentálne získavanie charakteristík je drahé a časovo náročné, a taktiež preto, že každoročne sú publikované mnohé sady údajov vhodné na trénovanie takýchto prediktorov. Jedna z nedávno vyvinutých metód, nazývaná innov'SAR, ktorá bola použitá už v niekoľkých aplikáciách proteínového inžinierstva, kombinuje Fourierovu transformáciu z čiastočnou lineárnou regresiou. Avšak, jej implementácia nie je voľne dostupná a samotná metóda nebola štatisticky overená. Cieľom tejto práce je adresovať tieto nedostatky, implementovať túto metódu v jazyku Python, rozšíriť ju a zahrnúť do ľahko použiteľnej platformy, ktorá umožní trénovanie a testovanie modelov. Taktiež bolo vykonané testovanie štatistickej významnosti za účelom overenia dopadu nájdených závislostí medzi sekvenciami a vlastnosťami proteínov. Metóda sa osvedčila ako štatisticky významná so silnými závislosťami nájdenými medzi vstupmi a výstupmi. Novo zozbierané dátové sady haloalkán dehalogenáz sa použili na vytvorenie modelov s validačným skóre $Q^2 = 0.54$ a $Q^2 = 0.77$, čo je takmer dvojnásobné zlepšenie oproti základným modelom. Tieto modely majú potenciál na filtrovanie väčších databáz sekvencií a vyhľadávanie proteínov s potenciálne lepšími vlastnosťami.

Keywords

machine learning, protein engineering, bioinformatics, PLS, haloalkane dehalogenases

Kľúčové slová

strojové učenie, proteínové inžinierstvo, bioinformatika, PLS, haloalkán dehalogenázy

Reference

LACKO, Dávid. *Platform for Biological Sequence Analysis Using Machine Learning.* Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Tomáš Martínek, Ph.D.

Rozšírený abstrakt

Proteíny sú základným stavebným blokom živích organizmov. Zabezpečujú rôzne funkcie naprieč celým spektrom živočíchov, rastlín a húb ako napríklad katalyzovanie chemických reakcií v podobe enzýmov, manipulácia DNA, tvorba stien a výstuže buniek, transport látok a iné. Sú tvorené reťazcom amino kyselín rôznych dĺžok od niekoľko desiatok až po niekoľko desiatok tisíc. Proteíny majú štruktúru na úrovni aminokyselín, ale aj v 3D priestore a niekedy aj medzi sebou. Je viacero vlastností proteínov, ktoré sa študujú a je snaha ich vylepšiť, napríklad tepelná, či chemická stabilita, enzymatická aktivita a iné. Proteínové inžinierstvo sa zaoberá štúdiom známych proteínov a ich vlastností a snaží sa vylepšovať existujúce alebo hľadať úplne nové proteíny, ktoré by spĺňali určité vlastnosti vzhľadom na využitie.

V súčasnosti sa v najväčšej databáze proteínových sekvencií nachádza viac ako 250 miliónov sekvencií, avšak len malý zlomok z nich je anotovaný, či preskúmaný. Rýchlosť, ktorou toto množstvo sekvencií narastá je veľmi neúmerné tomu, ako rýchlo sa skúma funkcia, či vlastnosti týchto proteínov. Preto je potrebné vyvíjať nové a aplikovať existujúce nástroje zamerané na strojové učenie, ktoré by umožnili rýchlo analyzovať a kategorizovať tieto sekvencie. Jeden z týchto nástrojov je napríklad AlphaFold vyvinutý spoločnosťou DeepMind, ktorý je založený na hlbokých neurónových sieťach a umožňuje predpovedať 3D štruktúru zo sekvencií proteínov, čo je jedna z najťažších oblastí proteínového inžinierstva.

Avšak na predpoveď iných vlastností proteínov nemusí byť potrebné až tak komplexné riešenie, čo ukázali aj vedci zo spoločnosti Peaccel, ktorý predstavili metódu innov'SAR. Táto metóda využíva Fourierovu transformáciu a informácie z AAindex databázy na vytvorenie modelu založeného na metóde čiastočných minimálnych štvorcov. Tieto modely sa dajú natrénovať na akúkoľvek numerickú vlastnosť proteínov a ako vstupné dáta stačia sekvencie aminokyselín. Základom je zakódovanie sekvencie pomocou určitej vlastnosti z databázy AAindex a následnou aplikáciou fourierovej transformácie na túto numerickú sekvenciu pre získanie vstupných dát pre metódu strojového učenia. V tomto prípade bola zvolená metóda čiastočných minimálnych štvorcov – partial least squares (PLS). Dôvodom je jej jednoduchosť a rýchlosť, ale aj schopnosť pracovať so sadami dát, ktoré majú viacej parametrov na prvok ako samotných prvkov. Pre použitie PLS musia byť sady dát reprezentované ako matice o rozmeroch $m \times n$ a teda platí, že m môže byť väčšie ako n. Toto jej umožňuje pracovať s malými sadami dát a zároveň obsiahnuť dostatok informácii v parametroch pre natrénovanie silného modelu.

Predmetom tejto práce je vytvorenie platformy postavenej práve na metóde innov'SAR, ktorá umožní vytvorenie modelu, jeho ohodnotenie a použitie na predpoveď hodnoty vlastností iných proteínov. Táto práca nielen implementuje innov'SAR, ale ju aj rozširuje o paralelizmus, caching a hodnotiacu funkciu založenú na spearmanovej korelácii. Samotná platforma je implementovaná v jazyku Python s použitím viacerých knižníc. Užívateľ s finálnou platformou interaguje pomocou príkazového riadku, kde zadá požadovanú funkcionalitu, teda trénovanie, testovanie alebo predpoveď a potrebné dátové súbory. Výstup platformy je zase vo forme súboru modelu alebo výpisu dát do príkazového riadku, ktorý sa jednoducho dá presmerovať do súboru.

Súčasťou práce je aj testovanie výslednej platformy na dvoch rôznych sadách dát, čo zahŕňa trénovanie modelu a hodnotenie modelu pomocou testovacej sady pre obe hodnotiace funkcie – Q^2 aj spearmanovu koreláciu. Výsledky ukazujú, že pre tieto konkrétne sady dát predpoveď nie je úplne presná, avšak vytvorené modely sa dajú použiť na filtrovanie objemných databáz pre výber sekvencií s očakávanými hodnotami skúmanej vlastnosti. Každá sada dát je zameraná na iný typ vlastnosti, jedna sa zaoberá katalytickou aktivi-

tou enzýmov a druhá tepelnou stabilitou pri zahrievaní. Obe sady sa taktiež líšia v type sekvencií, pričom jedna sada reprezentuje sekvencie, ktoré sa líšia v dĺžke aj pozíciách amino kyselín a vyžaduje zarovnanie. Druhá sada obsahuje iba mutácie nad jednou sekvenciou a predmetom skúmania je teda efekt týchto mutácií na hodnotu tepelnej stability.

Pre otestovanie štatistickej významnosti metódy innov'SAR boli zvolené dva prístupy, kde každý hodnotí iný aspekt závislostí, ktoré táto metóda hľadá. Toto testovanie sa vykonáva za účelom zistenia, či nezmyselné dáta, ktoré však kopírujú štatistické vlastnosti reálnych dát sú schopné dosiahnuť rovnaké skóre modelu ako reálne dáta. Výsledkom je teda informácia, či model naozaj zachytáva relácie medzi vstupmi a výstupmi, alebo akékoľvek dáta sú schopné dosiahnuť vysoké skóre. Prvý test založený na permutácii výstupov poukazuje na to, či tieto výstupy naozaj závisia na vstupoch, teda či vstupy obsahujú informáciu diskriminujúcu inú hodnotu výstupu. Druhý test bol založený na vytvorení štatistických modelov z existujúcej sady dát a následnej generácií náhodných vstupných sekvencií, ktoré však kopírovali štatistické rozloženie reálnych dát. Oba tieto testy ukázali, že metóda je štatisticky významná, a že akokoľvek preusporiadané dáta produkujú modely s menším skóre.

Platform for Biological Sequence Analysis Using Machine Learning

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Tomáš Martínek, Ph.D. The supplementary information was provided by Stanislav Mazurenko, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Dávid Lacko
May 9, 2022

Acknowledgements

I would like to thank my external scientific supervisor Stanislav Mazurenko for patience and immense support throughout this thesis and while I was getting myself familiar with the field of bioinformatics and protein engineering. Thanks goes also to the whole team of Loschmidt laboratories for their help and warm welcome to the T-Team. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Contents

Chapter 1

Introduction

There are more than 250 million sequences in the UniProt [25] database, which is the largest and most commonly used database for protein sequences. However, only a small fraction of these proteins are annotated. Knowing the protein structure can usually help predict its function, but current databases (e.g. RCSB[1]) of protein structures cover only a minor fraction of those sequences. Thus, there is an urgent need to produce sequence-based predictors of protein characteristics, such as protein function or stability, especially for a specific protein family.

Multiple machine learning methods have been adapted or invented for the fields of bioinformatics and protein engineering. They range from simple statistical methods to advanced multi-layer neural networks, depending on the amount of data available for training. For instance, the deep neural network AlphaFold has recently demonstrated tremendous potential in predicting protein structures based on the available dataset of around 180 000 experimental structures deposited in the database. For the prediction of protein function, activity, or effects of mutations, typical datasets are much smaller and often contain dozens to hundreds of data points. This has motivated the development of simpler linear predictors, which are less likely to overfit such small datasets but might still provide useful guidance in navigating the vast available protein sequence space. One of these methods is called an innovative sequence activity relationship or innov'SAR. It was developed, published, and patented by the Peaccel company based in Paris, and it is a proprietary method. It uses the Fourier transform as the main novelty, which despite being linear, allows mixing together signals coming from different parts of the protein sequence. However, since the method is proprietary, the code is not available. Moreover, the method was not verified in terms of statistical significance and used only one scoring function.

The goal of this thesis is to address these limitations and design a platform for protein prediction tasks based on the innov'SAR method. This platform aims to provide users with an environment to create models for predicting various protein properties without the need for protein structure. Since it is expensive and time-consuming to obtain the 3D structure of proteins and is sometimes even not possible with current technologies, numerous machine learning approaches have been created and published to predict the protein structure. However, these are still not 100% precise, and using the output of such predictors for feature generation in other predictive approaches can further propagate the uncertainty and errors and decrease the final precision. Thus, the goal of the platform is also to create a functional predictor without the need for any kind of structural information.

[1] https://www.rcsb.org/

This platform has also been verified in terms of performance and statistical significance on two newly collected haloalkane dehalogenase datasets for training and testing. To do this, permutation testing and random sampling were executed to verify different hypotheses and evaluate the performance of the predictor on the independent subsets of these two datasets.

Structure of the thesis. Chapter 2 offers a brief introduction to the field of protein engineering and machine learning. In chapter 3, an overview of the state-of-the-art machine learning methods for protein engineering can be found, as well as highlights of their problems. Chapter 4 provides a detailed description of the design and implementation. The platform and algorithm testing and its results are provided in chapter 5, including model strength tests, as well as method robustness testing. The final chapter summarizes the results and offers some perspectives for future extensions.

Chapter 2

Summary of the current state

2.1 Proteins

Proteins are the universal building blocks of life. They facilitate many functions within living organisms, such as catalyzing reactions as enzymes, manipulating and repairing DNA, providing structural support to cells, transporting molecules, acting as a defense mechanism against foreign bodies, signaling, or muscle movement [2]. From a chemical point of view, proteins are macromolecules that comprise long chains of amino acids. These chains must also be specifically structured in the 3D space to be able to perform their function. Thus, there are multiple protein representations, from a string of letters representing amino acids to a set of 3D atom coordinates.

Amino acids are chemical compounds that consist of amino and carboxylate functional groups, along with side chains specific to each amino acid. Their chemical structure allows them to join together into a chain. In living organisms, proteins are encoded in DNA. In cells, DNA consists of genes that encode the amino acid sequences and are used as templates for mRNA, and proteins are produced by joining the required amino acids into a chain based on mRNA and then folding this chain in a specific 3D structure. The processes through which DNA is interpreted and proteins are synthesized are called transcription and translation. There are more than 500 different amino acids, however, only 20 of them appear in the genetic code and thus primarily constitute different proteins. Each of these amino acids has a letter code assigned to it (Table 2.1) to represent protein sequences as strings of

Amino acid	letter	Amino acid	letter	Amino acid	letter
Alanine	A	Glycine	G	Proline	P
Arginine	R	Histidine	H	Serine	S
Asparagine	N	Isoleucine	I	Threonine	T
Aspartate	D	Leucine	L	Tryptophan	W
Cysteine	C	Lysine	K	Tyrosine	Y
Glutamine	Q	Methionine	M	Valine	V
Glutamate	E	Phenylalanine	F		

Table 2.1: The most common amino acids occurring in proteins and the corresponding one-letter notation typically used in the field of protein engineering to represent protein sequences.

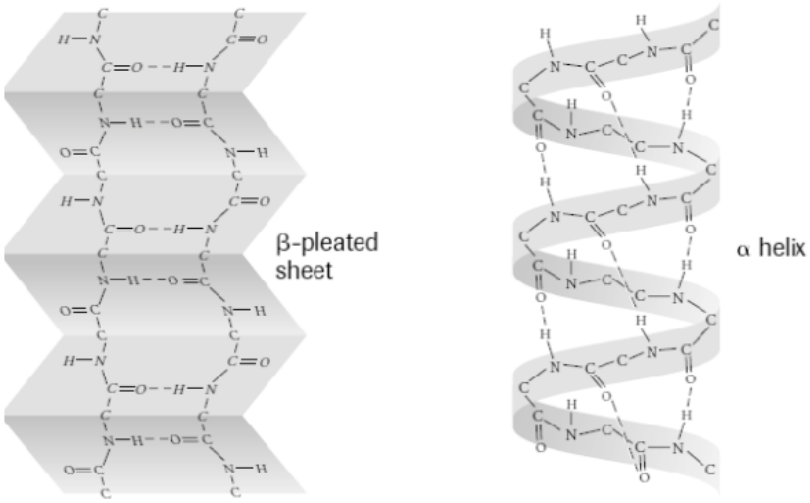


Figure 2.1: Example of secondary protein structure – α-helix and β-sheet. (Taken from [8])

letters, which is representation of the primary structure and is the most widely used protein representation.

During protein folding, individual side chains of amino acids form local structures stabilized by hydrogen bonds, the most common examples being the α-helix, β-sheet and turns, all of which form the secondary structure (see Figure 2.1). The tertiary structure of a protein is the placement of individual residues in the 3D space, which represents the overall shape of the macromolecule. The correct function of the protein is greatly impacted by this structure. For instance, a protein in a denatured state still typically has the same primary and partially secondary structure, but it usually loses its function. The denatured state of a protein is defined as a state when the protein loses its native fold due to some external circumstances such as change in acidity, temperature, pressure, or other factors.

Proteins are widely used throughout the industry, such as in chemical, medical, pharmaceutical, and biotechnological fields. Humans are indirectly taking advantage of proteins, mostly enzymes, e.g. by using cells that produce them. The most prominent use is the production of alcohol and dough with yeast, which has been around for millennia. One of the earliest out-of-cell usages of proteins was in the 1960s as an additive to detergents for home and industry use. Another example of industry usage of proteins is for improving stability of lubricants for machines. In the medical field, insulin, a hormone responsible for the absorption of sugar from the blood, is used in its pure form. Another example is the penicillin acylase antibiotic [9]. Some enzymes are used for biosensing of different substrates or degradation, such as PET plastic degradation [17].

All proteins bind to other molecules, generally called ligands. These can be other proteins, simple inorganic or more complex organic molecules. **Enzymes** are proteins that not only bind to other ligands, but also speed up reactions by lowering the energy barrier required for the reaction. In this process, the enzymes themselves typically do not change and thus act as catalysts. They allow cells and organisms to control chemical processes, produce essential chemicals, and thus sustain themselves. The ligands that the enzymes bind to are called *substrates*, and after the chemical reaction occurs, the resulting compound is called a *product*.

Proteins have different properties that affect their function and usability outside their natural environment – a cell. All proteins are characterized in terms of thermal stability,

which is usually expressed in terms of melting temperature. This temperature is defined in temperature scanning experiments, in which the temperature of a sample is gradually increased, and the fractions of folded and unfolded protein molecules are calculated. The melting temperature is then defined as the temperature at which the concentration of denatured and folded proteins is the same. In other words, it is the temperature at which half of the initial protein sample is denatured. This property is important for the industrial and laboratory use of proteins because the working temperatures of machines and processes can be higher than those in nature. Furthermore, proteins are exposed to more destabilizing factors when used outside of their natural environments. For example, some enzymes can be used as biosensors in chemical or pharmaceutical production, which might require higher temperature tolerances for the enzyme or higher pH resistance. Moreover, higher thermal stability is usually associated with longer storage times.

Another property that is interesting in enzymes is their activity, or how well or fast they facilitate certain reactions. Some enzymes might have high activity, but only catalyze a particular reaction, others might be slower, but catalyze multiple different reactions. In this case, the requirements on the enzyme depend on the process in which it is used.

In protein engineering, the goal is to modify the protein sequence to improve or change these properties in such a way that the protein would satisfy the requirements for industrial or chemical use. The goal is also to provide tools to facilitate these improvements and make it easier to customize proteins for different uses.

Haloalkane Dehalogenases

These proteins are enzymes that catalyze chemical reactions in which the halogen in a halide molecule is replaced by a hydroxyl group derived from water. Halides are often toxic and pose a risk to the environment, and they do not degrade naturally and stay in the soil for a long time. For example, one of such pollutants is β-hexachlorocyclohexane (β-HCH), which was used extensively as a pesticide in the past and is associated with Parkinson's and Alzheimer's disease [3]. Although it is now forbidden, traces of it can still be found in the soil and water. Haloalkane dehalogenases can be used as biosensors for such pollutants. They can also be used for bioremediation, e.g. to degrade the pollutants or clean up warfare agents.

2.1.1 Protein engineering

Currently, in protein engineering, the most common strategy to create the desired protein is to take a naturally existing protein that has the required functionality and change its structure to improve a certain property. A large group of methods for protein engineering is rational design, which usually includes taking 3D structures of given enzymes, then analyzing catalytic or tunnel residues, and suggesting means to alter a specific catalytic activity [21]. An alternative group of approaches is directed evolution, in which mutagenesis is used to create a diverse random library of enzyme variants, which is then screened for a specific catalytic activity, and the successful candidates are used as templates for the next iteration. While the directed evolution has been proven efficient [13], it is usually an expensive and time-consuming method that requires special laboratory equipment. In either group of methods, the change in structure is typically made in the amino acid sequence by substituting, removing, or inserting one or several amino acids. The substitutions are often also called mutations, the original protein template is called wild type if it occurs in the nature, and the resulting protein is called a mutant or variant.

Despite the major success of these two groups of strategies to date, it is very common that with improving one property, other properties change too. For example, improving enzyme activity might lead to lowering the melting temperature. Therefore, improving or altering the desired properties without compromising other important features of the protein is one of the biggest challenges in this field.

Another challenge when designing mutants comes with the number of possible combinations of amino acids. Usually, the protein sequences are hundreds of amino acids long. Then, considering that a sequence is, for example, about 300 amino acids long and in each position there can be 20 different amino acids, there are $20^{300} = 2 * 10^{390}$ different mutants which is more than the atoms in the observable universe. Thus, it is not possible to explore all the mutants exhaustively and the need for more directed exploration is needed.

This problem arises also from the fact that experimentally exploring even a few dozen mutants might take several weeks and is associated with significant challenges, e.g. when something goes wrong during the experiment, it needs to be restarted. And while the laboratory experiments are necessary to confirm that the designed protein has indeed the targeted properties, they could be at the end of in silico (computational) design process, which may help select only a subset of the most viable mutants for laboratory testing.

The computational tools for such tasks include various modelling environments, molecular dynamics simulations, or statistical methods based on large databases. Many such tools can achieve high precision, but they usually need to be manually targeted to a specific protein or protein family and require manual configuration and tuning. Thus, they cannot change the field dramatically when it comes to, e.g., annotating more than 250 million sequences that are available in databases. Lately, the focus has turned to machine learning approaches. This is due to the fact that machine learning seems to thrive in other fields of computer science and provides good performance and precision. It has also already been applied in protein engineering and has demonstrated a great potential in assisting in various protein engineering tasks [18].

2.2 Machine learning

Machine learning (ML) [24] is a field of computer science that includes algorithms for identification of patterns in data. The applications of machine learning are more and more common in everyday life, e.g. for spam filtering, customized web searching, image and speech recognition, and others. It also shows huge potential for processing and analyzing the results of wet lab experiments. Usually, these algorithms are trained on a sample of data that is called training data or training dataset. Machine learning encompasses many approaches and areas, but the dimensionality reduction and supervised learning are at the core of this project, and so they are explained in more details below.

In supervised learning, the algorithm is presented with input data, called features, and output data, called labels. The goal of the algorithm is to make predictions of labels using the features, based on the examples it is provided during learning. The idea is similar to how humans learn basic things, e.g. when children see a car for the first time and the parents tell them what that object is called, they are able to distinguish this object later on, even if it is not exactly the same color or shape. Here, the input features represent the image of the car, and the input labels represent the name of the object – a car. With enough data, the algorithm should be able to learn the connections (if there are any) between the features and the labels, and when presented with data that do not have the labels, it should be able to deduce the labels based on the learned correlations.

There are two types of labels and information that they can represent. The first one is categorization where the labels represent a category to which the input features correspond. An example of this kind of labels is the type of an object in an image, where a model can be created, e.g., to distinguish cars in a particular image. The second type of labels has the form of continuous variable. An example can be an estimation of the distance between some object in the picture and the camera. The problems that can be described by the first type of labels are called classification problems and problems with the second type of labels are called regression problems. The problems being solved in this project belong mostly to the regression category. The result of supervised regression algorithms are models that were trained on the training data to be able to predict labels for the new data similar to the training data. The goal is to create a model that is able to estimate labels with high confidence.

In contrast to supervised learning, in which the data is usually labeled, the dimensionality reduction is an unsupervised technique used for extracting the most valuable information for a set of data while also using the least possible number of values, and in a way it is similar to compression. The usual input for dimensionality reduction is a matrix of features, and the output is a vector or a smaller matrix.

Principal component analysis – PCA [4] [16] is one of the simplest dimensionality reduction techniques. PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system. Its main goal is to reduce the number of features and eliminate the linear correlation between them by producing new independent features. This is achieved by finding new base vectors that best capture the variation in the original feature set but are orthogonal to each other, and then projecting original features to the new orthogonal basis made from these vectors, thus creating new independent features.

PCA can be seen as an iterative process, whereby for each component k (except the first one) new data matrix X_k is computed by subtracting data contained in the previous components w_k from the original data X:

$$X_k = X - \sum_{s=1}^{k-1} X w_s w_s^T. \tag{2.1}$$

Then the eigenvector w_k corresponding to the largest eigenvalue of the covariance matrix $X_k^T X_k$ is computed for the data matrix X_k according to the following equation:

$$w_k = arg\ max_{w \neq 0} \left\{ \frac{w^T X_k^T X_k w}{w^T w} \right\} \tag{2.2}$$

Ideally, when the original feature set is correlated, just a subset of all orthogonal vectors is needed to explain most of the data variation in the original feature set. And while some details are lost, the decrease of original dimensionality is more valuable. One of the common strategies to find the eigenvectors is to use Singular Value Decomposition.

Dataset size problem

In machine learning, especially deep learning, datasets in the size of multiple thousand entries are often required to produce a high-quality predictor. Some notable examples of such models include AlphaFold, which was trained on 29000 datapoints to predict protein structures, or DEEPre for predicting EC (Enzyme Commision) number, trained on more than 20000 entries. However, in many applications in the field of protein engineering, it is

time-consuming and expensive to obtain such large amounts of data. It may take several weeks to produce a mutant, and then even more time to measure the property. Moreover, even when there are large datasets available, they usually consist of heterogeneous data, and, for example, stability predictors trained on larger datasets often have poor results for a particular protein. There are modern experimental methods that can significantly reduce the time for measurements, such as microfluidics [27], but the time required for synthesis of the mutant is still quite long. Therefore, machine learning methods that can work with small datasets (size in tens or hundreds of entries) are required to make use of the available data. These methods are usually more simple, such as linear regression, and the emphasis during the development of such tools is typically placed on designing appropriate features and good data representation, e.g. by embedding or encoding.

Proteins can be represented in many ways: as an amino acid sequence, secondary structure sequence, or coordinates of atoms in 3D structure. Some of these representations have numerical values naturally (coordinates), but others are character sequences, and they need to be transformed to some kind of numerical values. One of the simplest ways of representing the amino acid sequence is one-hot encoding where each position is substituted by a vector of size 20 of 1s or 0s, and each position in this vector represents a different amino acid, the 1 is on the position in the vector that corresponds to the amino acid found in the particular position in the sequence. However, this encoding results in quite sparse representation, i.e. many entries being zeros. Another way to represent the sequence is to pick some chemical characteristic that can be measured for an amino acid and translate the letters of amino acids into numbers using this characteristic. However, there are many of these chemical characteristics to choose from (see subsection 2.2.1). Regardless of the selected representation, in most cases, the resulting feature vector is much larger than the number of labeled datapoints, and a careful selection of algorithms is required to handle such imbalanced representations.

2.2.1 AAindex

The AAindex[1] is a database of amino acid properties. It contains 566 properties at the time of writing. Many of these properties are redundant. Each entry has a table of all amino acids and a numerical value of the given property for the particular amino acid. The entries also have a unique identifier, description, list of similar properties, and authors who contributed the entry.

Using the AAindex, the amino acid sequence can be translated into a numerical sequence of some property. This translation is used in the pipeline to transform the sequences into numerical representations that can be used in the analysis.

2.3 Partial least squares – PLS

Partial least squares is one of the linear regression techniques that can be easily used with these representations. It finds relations between two matrices X (features) and Y (labels) by projecting the features and labels to a new space. PLS creates a multivariate linear prediction model further referred to as model or predictor. Its main advantage is that it works well even when the number of features exceeds the number of datapoints, and thus it is widely used in chemometrics and bioinformatics. It also works well with noisy data

[1] https://www.genome.jp/aaindex/

and data that is collinear [28]. Some examples of use for PLS are predicting diesel blend properties [12] or analyzing stream of wood particles for cellulose, lignin and glucan [14]. However, the disadvantage is that each input feature contributes separately to the output label and this poses a problem when it is used with the sequence representations, because there usually is some behavior that comes from the amino acid vicinity in 3D space, but in the PLS, this would not be taken into account. A partial solution might be to produce features for combinations of 2, 3, or generally K neighboring amino acids, which would solve this problem for amino acids that are next to each other in the sequence, but will have limited effects for the interactions in 3D space between amino acids that are not next to each other in the sequence.

Often, while designing a predictor, signal processing functions can be used on the existing data to offer a different point of view and account for such cross-talks between different features. For example, these cross-talks might be solved by using the spectra representations by applying the Fourier transform. The idea behind using the Fourier transform is to capture the structural dependencies between the amino acids that might arise from their spatial vicinity in the 3D structure without the need for 3D structure data. It also might allow for more complex combinations of individual mutation effects than a simple linear combination because some mutations might interfere with each other, and their effect on the protein might overlap or even multiply, which is well represented by different frequency signal interference.

2.4 Fourier transform

The Fourier transform is a mathematical operation that translates the original function $f(x)$ or signal into its frequency spectrum $\hat{f}(s)$:

$$\hat{f}(s) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x s} dx, \quad \forall s \in \mathbb{R} \tag{2.3}$$

The result of such a transformation is a complex-valued function of frequency. The magnitude of the function value for a given frequency is the amplitude of the given frequency in the original signal. The angle of the same complex value is the phase offset. The requirements on the transformation and input signal are Dirichlet conditions.

For discrete signals, such as values measured by laboratory equipment, the Discrete Fourier transform (DFT) is used. This variation of the Fourier transform does not produce a continuous spectrum function, but rather a vector of finite length:

$$X[k] = \frac{1}{N} \sum_{j=0}^{N-1} x[j] e^{-ijk2\pi/N}, \quad k = 0, \ldots, N-1. \tag{2.4}$$

Here, X is the spectrum vector, x is the input signal, and N is its length. The variable k indexes the relative frequencies of the spectrum, and i is the imaginary unit of a complex number. The length of X is the same as N, but in the case of real-valued input signal, only half of X is considered, because the values in the other half mirror the first half. The Fourier transform is widely used in signal processing, informatics, bioinformatics, chemometrics, etc. [6]. For example, it is used in image compression, image processing, voice compression in telephony, and much more. An example of how it is used in this platform and in the innov'SAR method can be seen in Figure 2.2.

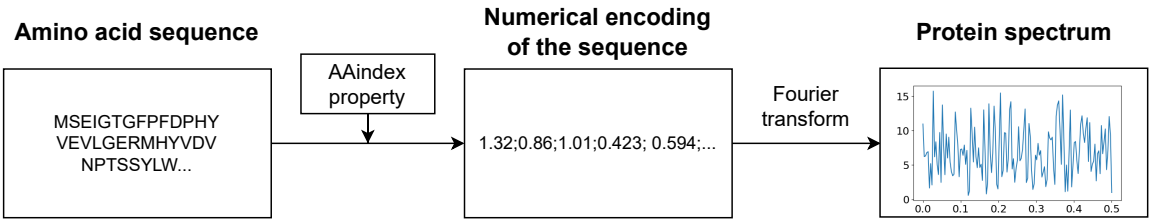


Figure 2.2: Encoding of the protein sequence - the sequence is encoded using a property from the AAindex and then the Fourier transform is used to convert the numerical sequence to a spectrum.

Chapter 3

Related work

In protein engineering, some typical tasks for the use of machine learning include predicting protein structure or catalytic activity. Predicting protein function from protein sequence is another challenge in this field, because the number of proteins that have only a known sequence, but an unknown function is huge. Protein solubility poses yet another problem, especially for the structural biology or pharmaceutical industry, where it is crucial to distribute a protein in solution for experiments or medication. Another example is directed protein evolution where the effect of individual mutations is studied and multiple-point mutants are designed by combining the explored mutations to achieve better properties.

Some notable examples of using classical Machine Learning algorithms in protein engineering is predicting enzyme specificity profiles, which capture how well given enzyme interacts with different substrates. In [29], the authors used the K-nearest Neighbours and Decision Tree algorithms for glycosyltransferase activity prediction. Although promising, this method used some manual adjustment and 3D structures for more precise prediction. Another recent approach [23] predicts the activity and substrate specificity of OleA enzymes. The authors used a range of methods, namely the Random Forest, Elastic Net, and Multivariate Adaptive Regression. Again, some structural features were used to create the predictor. Yet another method [22] includes using large neural networks and self-supervised learning on very large publicly accessible protein data sets. Although they showed some promise, these methods were applied to a specific group of enzymes, and their generalizability to other enzymatic families is an open question. In addition, they rely on structural information, which is not available in many cases, especially when using enzyme mining techniques in protein sequence databases, such as the recently published EnzymeMiner[1] [10].

Another recent ML approach in protein engineering for creating protein predictors without using structural data is using natural language processing (NLP) neural network architectures and workflows and adapting them for amino acid sequences. This method is deduced from the similarities between natural human languages and DNA and amino acid sequences because amino acid sequences can be seen as sentences. It expresses the protein as a sequence of words, each amino acid representing a single word or a letter. This sequence is interpreted in the cells, and proteins are synthesized based on the encoding of amino acid sequences in the DNA and the RNA. The research of machine learning in the field of protein engineering mimics the evolution of natural language processing. It started with Singular Vector Machines (SVMs) and Hidden Markov Models (HMMs). Then it

[1] https://loschmidt.chemi.muni.cz/enzymeminer/

continued with convolutional neural networks, recurrent neural networks like Long-Short term memory (LSTM). Each step presented an improvement over the previous method and offered new possibilities. In particular, with LSTMs it was possible to capture long-term dependencies. With the latest success of AlphaFold [15], Google released the architecture called Transformer [26], which is based on attention mechanisms.

Although these methods provide better accuracy, they are much more complex and require large datasets for training, which is often not the case in many protein engineering applications, especially those targeting a particular family of enzymes. Training times are also much higher [7] – in the range of days with the high performance computing option. Thus, despite the success of deep learning in protein engineering, simpler machine learning algorithms are still being developed. One of the recent examples is the innov'SAR method, which is covered in the next section.

3.1 Innov'SAR method

Innovative Sequence Activity Relationship or innov'SAR method allows creating prediction models only from sequences of proteins and amino acid properties and does not require any structural information [6]. For the prediction, it uses the partial least squares (PLS) method, which provides speed and is resistant to overfitting, making it suitable for small datasets. This framework is also quite general and can be used to predict various labels such as the catalytic activity, thermal stability, or other properties of protein mutants.

The method proceeds as follows. It first encodes the protein sequence as a numerical sequence of certain property from the AAindex database, further called *index representation*. Then the Fourier transform is performed on the *index representation* and the *spectra representation* is obtained. Next, the PLS model is trained using the spectra representation and label values, and a *predictor* or *model* is obtained. Finally, the previous steps are repeated for each AAindex, searching for the indices that produce models with the best scores. The entire process is visualized in Figure 3.1.

The idea behind using the Fourier transform is that changing a single amino acid in the protein sequence can greatly impact protein properties, such as the catalytic activity, and the spectrum produced by Fourier transform has similar behavior. In reality, the properties of a protein can be impacted by amino acids that are distant in the sequence by the effect of cross-talk, caused either by their spatial vicinity in the 3D structure or by molecular forces distributed along the structure. These interactions cannot be captured by a simple linear regression only from the *index representation*, but with the *spectra representation* it might be possible. This is due to the fact that without the spectra representation the change in the numerical sequence is minimal and localized, although the change in the predicted property can be substantial. However, the spectra representation of a certain sequence can greatly vary, meaning the frequency spectrum changes considerably, by changing just one value in the sequence. This ultimately represents the real-life behavior of enzymes, where single point mutations on certain places can dramatically affect the catalytic activity. The authors showed that this representation by Fourier transform can substantially improve the prediction model, and this claim was supported by applications on different datasets [5].

Another challenge innov'SAR addresses is the process of choosing the best indices from the AAindex, that is, the properties of amino acids that will help predict the property of interest the best. In the original innov'SAR paper [6], a combinatorial approach was chosen. First, the model for each property from AAindex is trained. For each model, scoring functions are calculated, and the model with the best score is selected. The whole

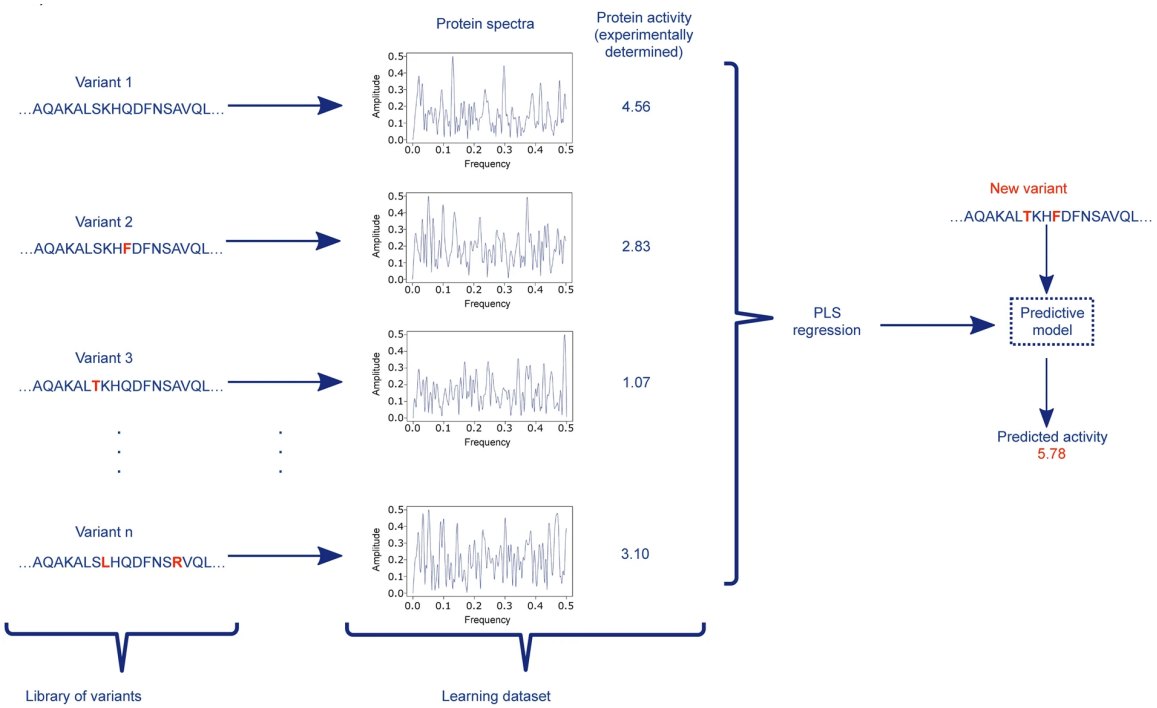


Figure 3.1: Process of the innov'SAR method - the spectra and the measured values are the input to the PLS which creates a model capable of predicting property for the sequences that were not in the training set. (Taken from [6])

process is repeated iteratively, each time adding new AAindex property to the previously chosen ones while improving the model score, i.e. in each iteration, the properties that had the best scores in the previous iterations are combined with each one of the not yet used AAindices. This process can be repeated any number of times, and in the original paper the authors stated that they carried out the combination of up to three indices or their spectral representations.

Despite great promise, the original innov'SAR method has several limitations. The major drawback is that the implementation of the method is proprietary and not available to the scientific community. However, the authors published the innov'SAR method publicly and described it in great detail, which motivated the reimplementation of this method in this project. The second limitation is that all of the experiments conducted in the original publications were performed on mutant datasets. However, theoretically, it is possible to apply the innov'SAR method to aligned sequences of different proteins, e.g. those sharing similarities in the sequences. The third limitation of the method was the large times required to train the model. In the original paper, the authors did not explore any means of parallelization. However, some parts of the training process can be easily parallelized to improve performance and training times. Finally, the authors did not explore other scoring functions than the cross validation score – Q^2. However, predicting exact labels based on small datasets seems to pose a significant challenge. And in many protein engineering tasks, e.g. the enzyme discovery problem, scientists are often more interested in ranking the protein candidates rather than predicting the exact values. This is due to the fact that the selected enzyme variants will still need to be validated experimentally. Therefore, the

goal of this project was to address those limitations and test the improved method on two new datasets, which is covered in more detail in the next two chapters.

Chapter 4

Algorithms and implementation

The major goal of this project is to implement the innov'SAR method and build a tool for using this method for custom data analysis and model creation. The innov'SAR method is also extended by a **prediction of aligned sequences of related proteins**, and means to preprocess the data by Principal Component Analysis. It also offers a possibility to create an archive containing all necessary files for MetaCentrum training. The original innov'SAR method was published on the case study of mutants. Moreover, the model was trained on mutants with few mutations and applied to mutants generated with these mutations. The novelty presented in this work is to use multiple sequence alignment with this method and to train the model with multiple-point mutants to see if the model is able to capture mixed mutations.

Furthermore, the original publication did not consider any computational optimizations for the algorithm, although there is a possibility for parallelization of the training process. **Parallelization is applied in this platform to speed up the training process.** Caching is also utilized to add ability to resume the training and to provide quick results for already trained models.

The next addition to the platform is **the possibility to choose between two scoring functions – Q^2 and the Spearman correlation coefficient.** The original paper used only Q^2, which scores how precise the predictions are with respect to the exact values. However, the Spearman correlation coefficient measures correctness of the label ranking and can, therefore, be used to score whether the predicted protein is expected to have a better or worse property than other evaluated proteins. This information might be useful for mass-selecting proteins for further lab experiments or more computationally heavy predictions.

Since PLS was mainly developed for real values, the complex Fourier transformed spectra needs to be converted to the real numbers. We tried multiple representations of complex numbers by real values, e.g. combining amplitudes and phase shifts, amplitudes multiplied by phase shifts, changing the sign of the amplitudes by the sign of phase shift. Ultimately, the best performance was gained by using only the amplitudes.

The innov'SAR paper provided more complex sequence encoding, where the protein spectra were combined with the index encoding in each step. This allowed both types of information to be explored during the training. However, during the implementation of this platform, we run some experiments to test this approach, and it proved to not improve the final model, if not degrade it. All the experiments point to the conclusion that the spectra encoding has more valuable information for the predictions. During the AAindex combination process of the innov'SAR method, not only the *spectra encodings* but also the *index encodings* were included in the property selection for each step. The

combination process never selected the index encoding and always preferred the spectra representation. Thus, we decided to not include the *index encoding* in the final platform to increase performance and training times at the cost of a possible decrease in precision in some cases.

The next section provides details on Partial least squares method and algorithm used for its computation and section 4.2 provides insights into the scoring functions used to score the PLS models and how the scores differ. In section 4.3 the general workflow of this platform is explained, and in section 4.4 details on implementation and challenges encountered during implementation are mentioned. Finally, in section 4.5, the general complexity of the algorithm and the time required for training are evaluated.

4.1 Partial least squares – PLS

The NIPALS algorithm is used for PLS regression, with one hyperparameter – the number of components. The following equations capture how outer relations are computed, which is similar to Principal Component Analysis. If X denotes an $n \times m$ matrix of features and Y is an $n \times p$ matrix of labels, PLS computes the following representation:

$$X = TP^T + E \tag{4.1}$$

$$Y = UQ^T + F \tag{4.2}$$

where T and U are projections of X and Y, P and Q are orthogonal loading matrices, and E and F are error terms. When computing the PLS, the goal is to maximize the covariance between T and U. To capture the dependencies between X and Y, the following equation is used.

$$Y = TBQ^T + F \tag{4.3}$$

Where B is the matrix of sensitivities, and the goal is to minimize a norm of F. In this work the PLS method from the scikit-learn[1] library was used, precisely the `PLSRegression` class.

4.2 Scoring functions

A scoring function provides a numerical score for a model or predictor. It usually penalizes the discrepancy between the predicted and true values. It can be used either to report the strength of a model or a method so that different methods can be objectively compared, or, as in this case, to select the best model from a set of possible candidates.

Coefficient of determination – R^2

For model evaluation, the R^2 score or coefficient of determination is often used in Machine learning to describe the fitness of the model.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{4.4}$$

[1] https://scikit-learn.org/1.0/modules/generated/sklearn.cross_decomposition.PLSRegression.html

$$y_i = \text{the i-th actual measured value}$$
$$\hat{y}_i = \text{the predicted value}$$
$$\bar{y} = \text{average of the measured values}$$

The coefficient takes values no greater than one, with one being the exact prediction, zero indicating the performance similar to that of a constant predictor (i.e. the predictor that ignores the features and simply returns the average label of the training set), and negative value representing the performance worse than that of the constant predictor. The implementation for this coefficient function is used from the scikit-learn library – `sklearn.metrics.r2_score`.

Cross validation score – Q^2

However, more important metric is the Q^2 score or cross validation score, which is specific to the PLS algorithm. This score is computed similarly to the R^2 score, except that the predicted values are not taken from the training set, but from the leave-one-out cross validation. This means that in each iteration, one datapoint is taken from the training set, and the rest are used to train a PLS model. The model is then applied to the datapoint that was left out to calculate the predicted label marked $\hat{y}_i$. This process is repeated for each entry in the dataset. Finally, the same equation as for the R^2 is used (Equation 4.4). Although this score is useful for tuning the parameters and experimenting with different dataset splits, it is not a good independent score of the final model, because the whole training set is not used to calculate this score. The best way to independently test and score the created model is to have an independent test set.

Spearman correlation - ρ

The Spearman correlation coefficient is a statistical measure of the monotonicity of the relationship between two variables:

$$\rho = \frac{cov(R(X), R(Y))}{\sigma_{R(X)} \times \sigma_{R(Y)}}, \tag{4.5}$$

where X and Y are two variables, $cov(\cdot)$ is a function that computes the covariance of the variables, $R(\cdot)$ is a function that returns the rank for the provided variable, and σ is the standard deviation. The rank is computed by sorting the variable values and indexing them from the smallest to the highest. The Spearman correlation coefficient is thus equal to the Pearson correlation coefficient between the ranks of the two variables [19]. It takes values from the interval $[-1.0, 1.0]$, indicating how close the ranks follow the same order (the value of 1.0) or the inverse order (the value of -1.0). The difference between the Spearman and Pearson correlation coefficients is that the latter is more strict and measures how well the relationship between two variables fits a simple linear function. The Spearman correlation coefficient is more lenient in terms of the shape of the function, scores the monotonicity of the relationship, and is not sensitive to the shape of the particular function, as can be seen in Figure 4.1.

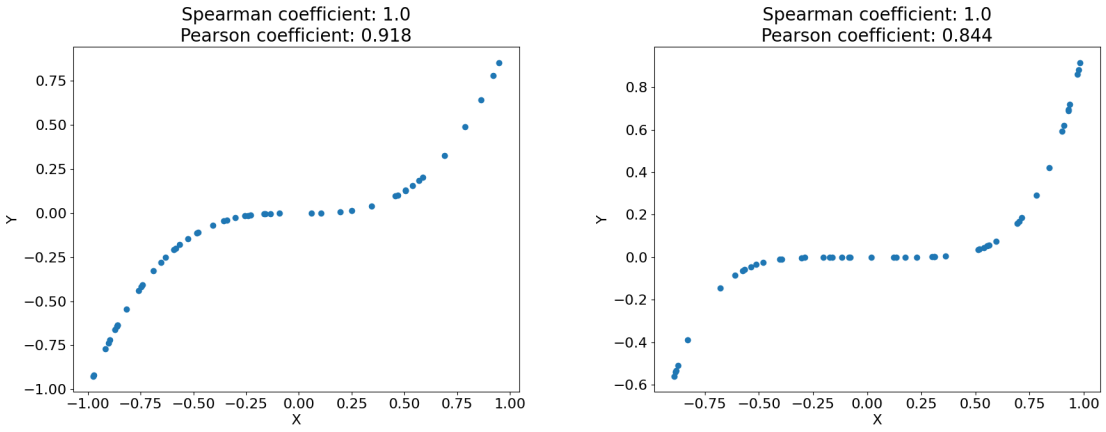


Figure 4.1: Comparison of the Spearman coefficient and the Pearson coefficient on similar data. As can be seen on the graphs the function that is more similar to a simple linear function has higher Pearson coefficient, but the Spearman coefficient stays the same, as both functions are increasing.

4.3 Workflow

The tool contains multiple levels, as can be seen in Figure 4.2. These levels represent the separate actions the user can take. The individual levels are explained in more detail in Figure 4.2b.

In order to create a model, the user has to provide training data to the tool. The users decide on the split for the input dataset into the training and testing sets for model validation themselves. This is left at the users' discretion since various tasks will imply different train/test splits. The tool accepts two input data formats: aligned protein sequences (*family*) or mutational data (*mutants*).

The first *family* type requires datapoints in the form of a multiple sequence alignment. This format is used for family datasets where sequences can differ in many positions or sequence lengths. However, it is important to align these sequences first before the training. The alignment is left to the user and is not in the scope of this platform, as there are many algorithms for aligning multiple sequences, and many parameters can be fine-tuned. There are also many tools to achieve this task, even available online for free[2]. The user should check whether the alignment looks reasonable and is without unnecessary gaps or other common errors. However, it is not crucial for the performance of the model to have the best possible alignment. More important is that it does not contain long gaps. For the correct testing, it is advised to align the training sequences separately and later co-align the test sequences to them. This workflow reflects the typical use case where the predicted sequences have to be co-aligned to the training sequences before the prediction because they are not known during the training of the model. The tool accepts a csv file with a predefined format (see Appendix B) as input of the sequences and the necessary data.

For the *mutants* type, the input file format is different. Two input files must be submitted to the tool. The first is the amino acid sequence of the template protein in a plain text file. The second file must contain a list of mutations in the csv file format in which the first column contains the mutation name, and the second column contains the list of muta-

[2] https://mafft.cbrc.jp/alignment/server/add.html

tions separated by a semicolon (the full format can be seen in Table B.2). The individual mutation has the following format:

$$XnnnY;$$

where X is the source amino acid, nnn is the position within the template protein sequence, and Y is the mutating amino acid. This essentially means that the amino acid X gets replaced by amino-acid Y at the position nnn. If X cannot be found at the nnn position, an error is reported. The mutation file must also contain the labels for individual mutations. The fourth column is optional, but recommended, when the same mutation appears more than once within the file. This column is used to aggregate the same mutants and average the labels. Its primary purpose is to specify a measurement method for the particular label value. Basically, the combination of mutations column and this column should form a unique key for each measurement.

If there are multiple output labels, the user must reduce the labels to a single value, e.g. by selecting a specific column or keeping only the first component after using the Principal component analysis (PCA). Using the *pca* command and providing the input file, the tool performs PCA and outputs the results in a file where each result is in a new line.

The last parameter necessary for the model training is the scoring function that will be optimized. There are two options:

- Q^2 – Leave-one-out cross validation score
- ρ – Spearman correlation coefficient

The Q^2 scoring should be used when the precision of the predicted values is important, but it should be noted that the model can achieve lower scores of R^2 depending on the dataset. The Spearman correlation coefficient can achieve higher final prediction scores (final Spearman correlation coefficient), but the output labels should not be considered as precise output values, but rather as an indication whether the input sequence is expected to have higher or lower label value than other proteins in the set. When using the Spearman coefficient, the results of the predictor can be used for ranking the sequences with unknown properties for further lab testing or more computationally intensive simulations. Thus, it can also be seen as a comparator mode. By default, the Q^2 score is used, but it can be changed by providing a command line argument.

After providing all the necessary input data, model training begins. The overall training time can take quite a long time, depending on the size of the input dataset. If the size is up to hundred data points, the training should take around ten minutes on a standard desktop computer. The output from the training is a model in a custom format file that is later provided to the tool for testing and prediction. The calculations are performed in parallel whenever possible (see section 4.6).

After the training, the model is validated, its performance checked, and the model score is calculated. For this reason, the tool provides a testing command - *test*. Here, two input files are required: the model that was trained in the previous level and the test set file. The format of the test set file differs depending on the type of model that was trained, similarly to the training input files: it must contain co-aligned sequences in the case of a *family*-type model, but otherwise the formats are the same as for the training set. Testing is a fast operation, and the results are usually available within a few seconds. On the basis of the results, the user could decide to further tune some parameters or redistribute the training and testing sets for better results.

If the validation of the model is satisfactory, the user can continue with predictions for new sequences, or the model can be distributed to other users. In order to predict a label for new sequences, the model file and the file with the new sequences must be provided. In the case of the *family*-type model, it is important to co-align the new sequences to the training set. It is worth noting that the co-aligning process, which aligns new sequences to an existing alignment, might shorten the input sequence if it is longer than the aligned training sequences. The prediction operation itself is also fast, and the results are available within a few seconds. During the interpretation of the results, users should be careful: the exact labels may be inaccurate when the model was optimized for ranking, which is the case when using the Spearman correlation coefficient. In this case, the value simply indicates whether the sequence is expected to have better or worse score than the other sequences.

4.4 Implementation details

The implementation of the method and the platform is done in Python language with the use of numpy (v1.22), scikit-learn [20] (v1.0.2) and pymol aaindex libraries. To download and access the AAindex database, we used the PyMOL library called aaindex[3], which downloads the database from the Web and provides a dictionary object in the Python code. The code of the platform is separated into multiple modules based on functionality and uses object-oriented programming.

The platform is designed with a command-line user interface with multiple modes of operation and optional arguments. For processing the arguments, the Python argparse library is used, especially its subparser capabilities. There are multiple subcommands – *train, test, predict, pca*, each facilitating a different level in the workflow.

In all the subcommands, either input or output files are needed. All files provided by the user must be in the csv format with the colon as the separator. The only file that has a custom format is the model file, which is produced after the training. It is a binary encoded object using the Python library **pickle** which allows exporting Python objects to files. Each type of input (family, mutants) have their own input file formats as mentioned in section 4.3. These input files are parsed and converted into Python objects for computation.

In the case of training, the instance of the `InnovSAR` class must be created and provided with the training data and settings. When initializing, the training sequence encodings and spectra are created for each AAindex entry and saved to a local dictionary with the AAindex ID as the key. This leads to shortening the execution time as they do not have to be computed every time they are required. Then the training starts, and it consists of multiple steps. In each step, a combination process starts and the *base* (the AAindices selected in the previous steps) is combined with one of the indices from the *pool* which contains not yet selected AAindices, this process explores all the possible combinations of the base with items from the pool. A model is trained on the spectra created from these combined AAindices and its score is saved. After trying all the combinations, the combination that created the best score is selected as the base for the next step and the process can be repeated. At the beginning the base is empty.

The the pool is divided between available processes and the model scoring is done in parallel. The number of simultaneous processes (computations) can be specified using a command line argument, and the default value is 8.

[3] https://github.com/Pymol-Scripts/Pymol-script-repo

Since the platform implements caching for the computation steps, first a check is preformed whether such cache files exist. If they do exist, they are used instead of running the computation for the training step. Usually there are three training steps, and each of these steps also represents a caching point. The number of training steps can be adjusted by a command line argument. The step must successfully finish in order to create a cache file, thus when the training process is stopped in the middle of the step, there is no caching for that particular step.

Finally, after the training is finished, the model is created and saved as a file using the `pickle.dump` function. This file does not contain the model created by the PLS algorithm, but rather contains the data necessary to recreate the model. This has two reasons, first one being the file size and second access to the training data during testing. Usually, the datasets that should be used with this platform are not very large and only protein sequences are required, which do not occupy a lot of space. If the matrices obtained by training a model using PLS were saved, they would occupy more space because one side of them is always $1.5x$ the length of the sequence and the other size depends on the number of components (on average 7), but there is multiple of these matrices. In some cases it is possible that these matrices would occupy less space than the raw sequences, but it is less likely, especially when saving floating point numbers requires more bytes than simple characters. Thus, in the model file, the training sequences and labels are saved as well as the training parameters. The PLS matrices are **not** explicitly saved, but the PLS model can be easily recreated from the training data. Retraining does not pose a problem since the PLS is quite simple and fast algorithm, and the retraining is performed within seconds.

The testing part of the software is much simpler. It parses the test set input file, creates a PLS predictor from the model file, and predicts the values for the test set. All the metrics are computed, and a graph is shown to the user, depicting real values on the x-axis and predicted values on the y-axis. This type of graph allows for simple depiction and evaluation of prediction accuracy, where the diagonal marks a hypothetical ideal predictor.

Code structure

There are multiple classes implemented that comprise the platform, as can be seen in Figure 4.3. The input data are represented by the classes derived from the `Dataset` abstract class. These classes can parse and process the input files based on what the user selected in the command-line arguments. The `Dataset` class must contain a list of sequences and their labels, and additionally the `MutationalDataset` contains the base sequence to which the mutations are applied.

To execute the training, the `InnovSAR` class instance needs to be created, and all the necessary settings should be set. Otherwise they will have the default values. Most importantly, the `dataset` attribute must have an assigned value of the `Dataset` class instance. Running the `compute()` method starts the training, and the final model is returned by this method as the `Model` class instance.

Within the `InnovSAR` class, there are multiple private methods, but two require more explanation. The first one is `_combinations_of_aaindices()`, and this method implements the parallelization and caching by writing and loading the cache files. If no such file exists, the parallel computation is run with `InnovSAR.process_num` processes. This function is called recursively for each step of the computation, and the number of these steps can be set by `InnovSAR.depth`. The number of steps determines how many AAindex-encoded spectra will be joined together for the final model. The user can also choose

to explore more models with *n_best* performing AAindices by setting `InnovSAR.n_best`. The recursion allows a tree-like exploration of the model space, by not only selecting one best AAindex model in one step, but selecting n_best models and exploring more combinations (see Figure 4.4). The second private function `_compute_for_all_aaindices()` is executed by each subprocess spawned in the `_combinations_of_aaindices()` function. It creates PLS models for each assigned AAindex and computes their scoring functions. The `PLSToolset` class implements all the functions that work with PLS, such as `q2` and `predict`. The `PLSToolset.pls_and_q2_component_search()` also performs a hyperparameter search for the optimal number of PLS components. This function returns the `PLSResult` instance which is used in the `InnovSAR._compute_for_all_aaindices()`.

4.5 Complexity estimates

By the description of the innov'SAR method and considering the fact that it uses a combinatorial approach, the training times for the method can be significant. To put it simply, first, for each AAindex, the method iterates over the training sequences, calculating their individual spectra. Then multiple processes start fitting PLS models for each AAindex encoding and calculating their scores, resulting in

$$N = s * M * c * (n + 1) \tag{4.6}$$

where, N is the number of models trained, s is number of steps, M is the number of AAindex entries, c is the number of PLS components that is being explored, and n is the training dataset size. The number of steps is the final number of AAindices that will be joined together. Since the PLS has one hyperparameter – number of components – that needs to be tuned during training and in order to do this, a cross validation is computed for each model with different number of components.

As an example of exact values, the family dataset with 24 entries will be used. First, the spectra database is build, that makes it $24 * 566 = 13584$ different spectra and thus that many Fourier transforms. Then, the innov'SAR method fits a model for each AAindex entry, and for every one of the computed models a scoring function with leave-one-out cross validation is calculated. By substituting values into Equation 4.6 the number of PLS fittings per training is $3 * 566 * 10 * (24 + 1) = 424500$, if we explore 10 different components and join up to 3 AAindices. This is quite significant number of models, and it means that for each AAindex entry $10 * (24 + 1) = 250$ different PLS models need to be fit. The $+1$ represents the model with none of the sequences left out.

As the number of the input features increases, the training times for PLS also increase as well as the number of PLS fittings. Putting all this together means that one full run of this method on a personal machine with 2.4GHz 4-core 8-thread CPU takes about 8-15 minutes for the family dataset. This does not pose a problem when training just one model, however to perform the statistical validation it was necessary to execute several hundreds of such trainings. In order to compute such large amounts of models more power and time is required than a typical desktop computer can provide, thus services of MetaCentrum were used.

4.6 Parallelization

Although the innov'SAR method does not allow for a complete parallelism, it is possible to do some parts in parallel. At certain points, all the results from the parallel computations

need to be collected, and based on the results, the next iteration of the parallel computations can be launched. The disadvantage of using Python as the main implementation language is the need to create new processes rather than threads to do these parallel computations, since Python does not allow to run multiple threads at the same time in the same process. Thus, to achieve true parallelism separate processes are required which creates a bigger memory and process management footprint.

Parallelization is used in each step to calculate models for each AAindex. The number of separate processes can be specified as an argument. Based on the number of processes, the AAindices are divided between these processes and each process performs the PLS training and the leave-one-out validation for the assigned list of AAindices. The implementation is based on the futures and promises, where each process provides a future with the best result of all the assigned indices. The main process waits for the futures, combines all the best results together, and selects the indices with the highest score.

With the custom parallelization, a problem has been revealed when using the numpy library, which already utilizes its own multi-threading to some extent. However, the Python does not have true parallel multi-threading due to its interpreted nature: the interpreter can execute only one thread at a time. This is also the reason why multi-processing is used on the platform, and each process has its own Python interpreter. It is important to note that the numpy library uses external implementations of linear algebra and other algorithms from libraries implemented in C. These libraries internally use true thread parallelism for certain operations. When the internal parallelization of mathematical libraries took place, the physical processor cores were loaded to 100%, but almost 80% was used by the operating system for management. This caused unnecessarily long training times. To solve this problem the internal parallelization of the libraries needed to be turned off and only the custom parallelization is used. Since the numpy uses C libraries of different implementations that can differ from system to system, it is necessary to disable the library multi-threading by setting the option for maximum number of threads to one.

4.7 MetaCentrum

MetaCentrum is a project of the CESNET department that handles the operation and coordination of the distributed computing and data storage infrastructure in the Czech Republic. There are many academic and research institutions involved in MetaCentrum, and they are able to share computational power using this service. The services are provided free of charge for students of most universities in Czech Republic, however registration is required to be given access to the system.

In order to compute certain machine learning models in a reasonable amount of time, it is necessary to use more computational power and resources than the personal machine is able to provide. MetaCentrum is used in this project to perform testing of statistical significance and it also might be beneficial for the users to run more extensive trainings of the innov'SAR method using the MetaCentrum .

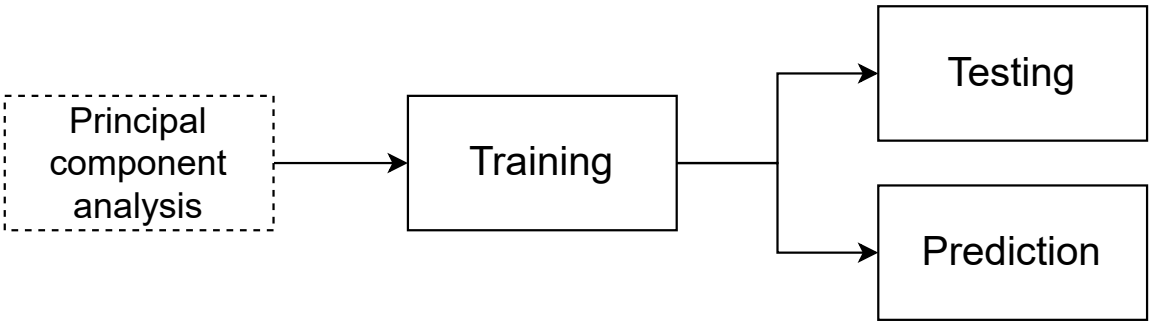
MetaCentrum is a distributed computational network with many nodes across Czech Republic, but from the user's point of view, it has uniform access [1]. The user first needs to log in using SSH to a frontend server, which does not serve as computational node, but rather for users to log in and schedule their computations.

The computations are scheduled using a bash script and the information needed for scheduling is the script that should be run at the destination node, number of CPU cores, memory and estimated time of running. The scheduler is a service within the system then

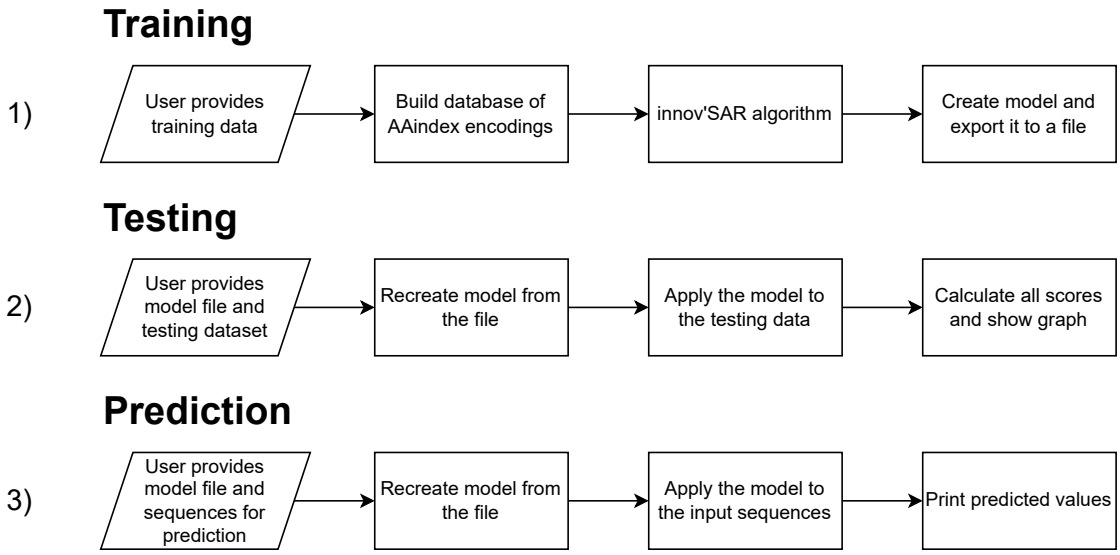
schedules and assigns different tasks to the nodes across the network to compute. For a better performance, the script that is invoked by the scheduler copies the necessary data to the current node and then runs the program itself. After the computation, it copies the obtained results to the user's home directory or another network storage because the persistence of the node memory is not guaranteed.

There are some time and power restrictions based on the user account level. For example, more publications acknowledging the usage of MetaCentrum can give the user access to more CPU time.

At the time of writing, MetaCentrum contains 30 894 CPUs and 531 GPU cards. In disk arrays there is 15 771 TiB of space in total and hierarchical storage contains different types of storage with total capacity of 17 843 TiB for long term storage with slower access times. MetaCentrum consists of 53 physical machines These are logically divided into 767 virtual machines. The whole infrastructure is geographically distributed around Czech Republic and divided between 14 institutions.



(a) General overview of the pipeline. The principal component analysis can be used to reduce the labels to one real value. The training process includes the innov'SAR method with parallelization. The testing provides means to validate the model by providing different scores and a graph of predicted values vs. measured values. Prediction step is the final one and allows predicting unlabeled data. (The dotted line represents an optional step.)



(b) Details of the individual pipeline parts. 1) describes the training step where the user provides training data and the result is a model file; 2) the testing requires test set and the model file created in training and provides statistics and graph; 3) should be the final step where the user provides sequences with unknown property values that are estimated by the model and printed for the user.

Figure 4.2: Workflow graph. In (a) a general overview is depicted and in (b) more detailed steps for each level are explained.

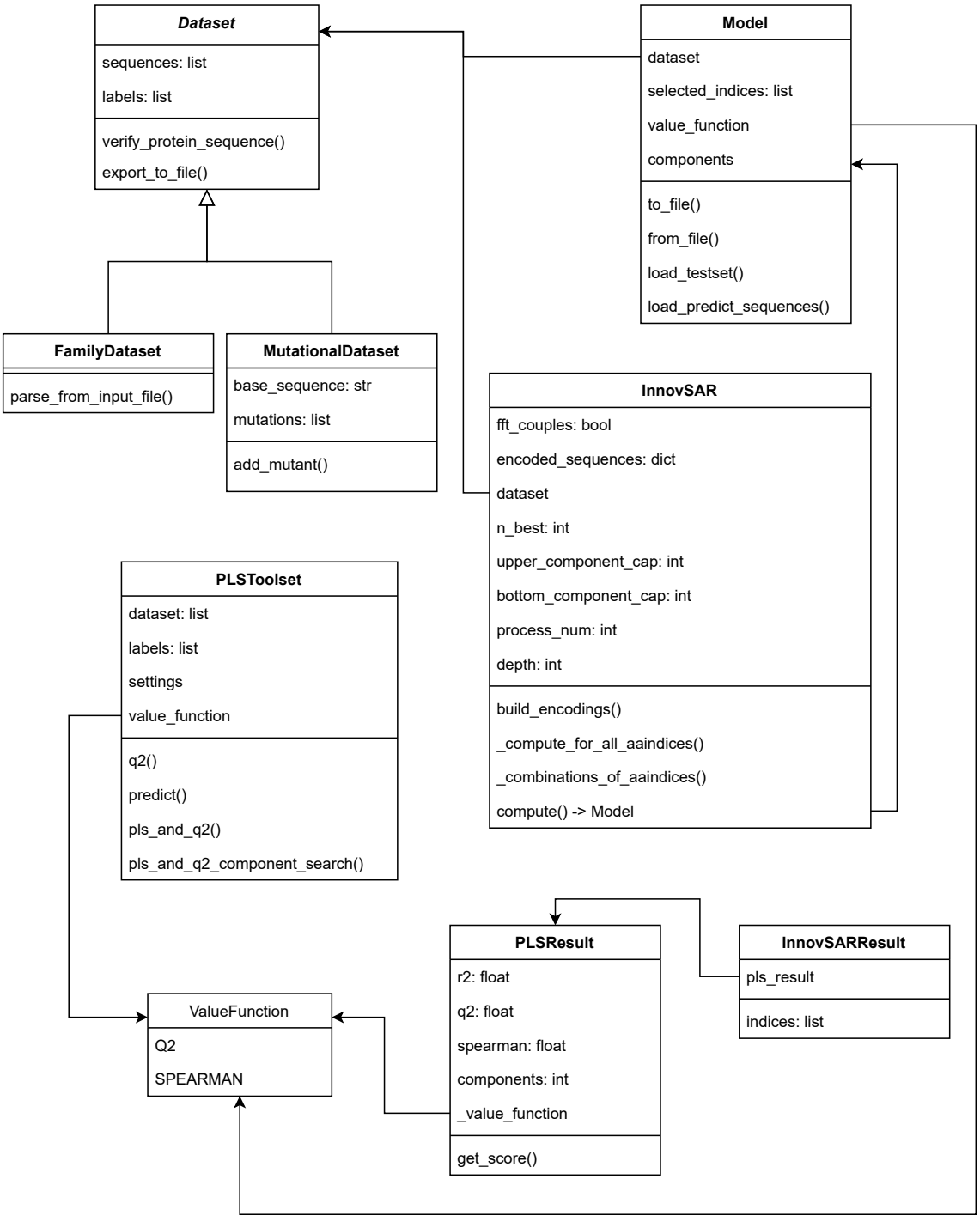


Figure 4.3: Simplified class diagram of the implementation. Most of the functionality is implemented in the `InnovSAR` class, where the innov'SAR method is implemented and in `PLSToolset` the Q^2 computation and PLS component optimization is implemented.

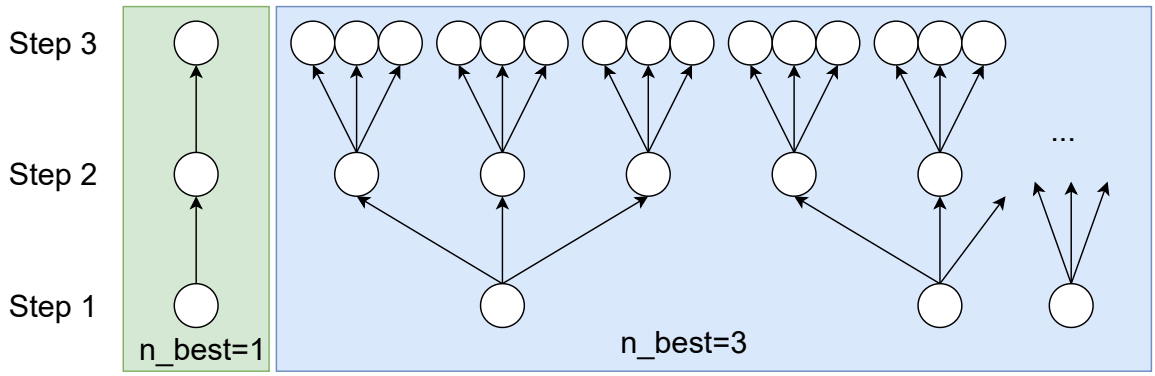


Figure 4.4: Effect of `n_best` property. The individual nodes symbolize different AAindex entries that are explored in depth-first manner. The number of combinations increases exponentially.

Chapter 5

Testing

The implemented platform has been used to create predictors for two different datasets. The datasets used for testing were selected based on the ongoing research in one of the protein engineering labs and consist of newly collected and recently published data for the haloalkane dehalogenase enzymatic family. The chapter is organised as follows. In section 5.1, the datasets are introduced and the challenges when using these datasets are explained. Section 5.2 presents the results of independent testing of the models using the test sets as well as the comparison with the baseline performance. Next, section 5.3 explains how the statistical testing was performed to evaluate the robustness of the method.

5.1 Benchmark datasets

The datasets on which the pipeline was tested were split into training and test sets to verify the strength of the predictors. The experimental data also contains a lot of noise, which decreases the precision of the final predictors. Both datasets were measured experimentally in the lab for the family of haloalkane dehalogenases.

Family dataset

This dataset contains 32 individual proteins with sequences of different haloalkane dehalogenases that differ in length and have less than 60% similarity. The size of this dataset is usually considered low for sophisticated machine learning methods, which is a common problem in protein engineering and is also the reason why the PLS algorithm might be useful. All of these proteins are haloalkane dehalogenases, and the property that was measured is their activity towards a set of halogenated compounds. The sequences were previously obtained using state-of-the-art enzyme mining tool EnzymeMiner[1] [11]. The activity was measured towards 24 different substrates using a recently published microfluidic platform. The PCA was applied to transform the multidimensional enzyme activity profile to a single value for each enzyme, representing its average activity towards the whole group of substrates (*total activity*).

Mutational dataset

This dataset consists of 130 different mutations of the haloalkane dehalogenase DhaA (Uniprot ID: P0A3G2). The format of the data is similar to that used in the original

[1] https://loschmidt.chemi.muni.cz/enzymeminer/

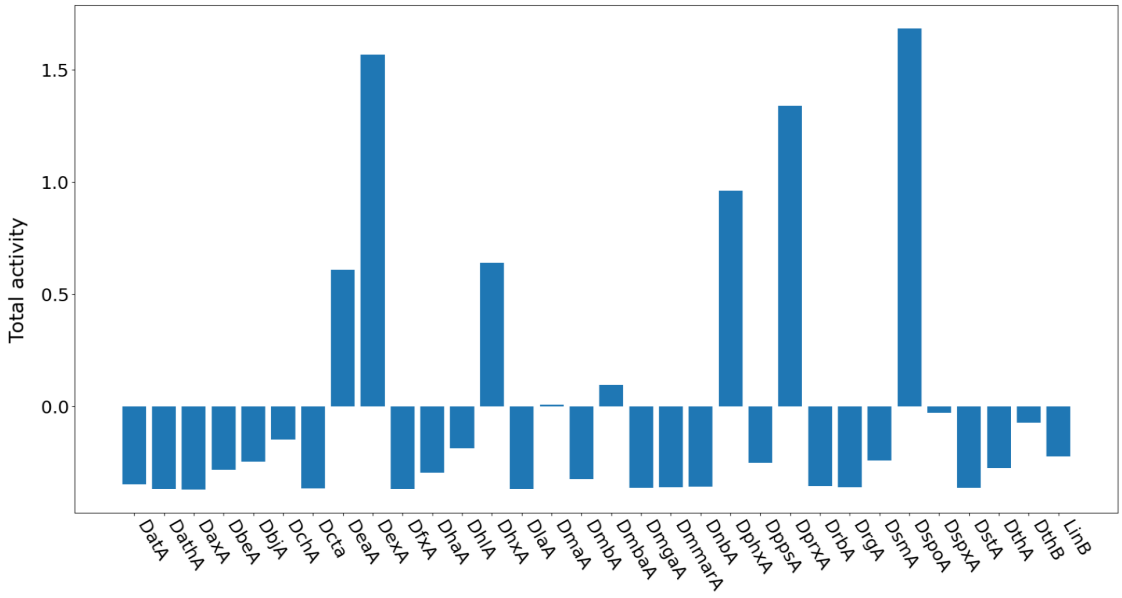


Figure 5.1: Family dataset - total activity values, which were obtained as first component in Principal Component Analysis. The original data was collected on a microfluidic platform for a set of halogenated substrates. A high positive value indicates high activity and negative values represent low-activity enzymes.

publication of the innov'SAR method. The property that was measured is the thermal stability in the form of the melting temperature. The data has been collected over the span of several years. Some of the mutants (3) had invalid mutations and had to be excluded from the dataset. Multiple measurements for the same mutant using the same or different methods were sometimes reported, and these needed to be unified to a single value. The unification was done by averaging the different temperatures, and using the average as the target label. However, this technique introduces inconsistent noise since some proteins were measured with up to three different methods and some only with one. Moreover, different methods might also introduce shifted measurements where some method might report higher melting temperatures than other. Thus, the analysis of the temperatures from different methods was performed to see if such inconsistencies exist. The average deviation between the methods was almost 0, so no conclusions or corrections of existing results could be made to minimize the error.

5.2 Independent testing

For an objective evaluation of the created model, another dataset, separate from the training dataset is required, called the test set. A typical workflow of machine learning is to train the predictor on a training set and then test the trained predictor on the data that the algorithm has not seen before to simulate real use. The test set is usually a portion of the available data that was not used during the training, and it serves for the verification of the predictive strength. For the test set, the real measured values are compared with the values predicted by the model, and the scoring function evaluates how well the obtained data fits the expected values. The R^2 score or Spearman coefficient (based on the training

parameters) of this dataset is the final score that evaluates the predictive power of the model.

5.2.1 Baseline

The goal of this project was also to verify the claims that the authors made about the performance and improvement of the innov'SAR method. To verify their claims, multiple baseline models were created to be compared to the models from the innov'SAR method.

All baseline models used the PLS algorithm. The first baseline model was simple regression on the feature vector consisting only of 20 values, each counting the frequency of a particular amino acid in the sequence. The second baseline model used the one-hot encoding of the sequence. Each position in the sequence was extended to a vector of 20. In this vector, each position contained either one or zero and represented a different amino acid. There is always a single one based on which type of amino acid is on the given position, and all other positions in the vector are zeros. The third baseline model is more complex. Every AAindex entry was used to encode a sequence, and all of these numerical sequences were joined together to form a feature vector. The final fourth model uses the Fourier transform similar to the innov'SAR method but does not perform any AAindex selection. It again uses the whole AAindex similar to the third baseline model, the difference being that each numerical sequence is first processed using the Fourier transform, and then they are all joined together to form a feature vector.

The baseline results can be seen in the Table 5.1. The first baseline model has the worst performance as the score was negative even for the training. All the remaining baseline models showed promising positive scores on the training data but performed poorly on the independent test data, with the negative scoring implying the performance worse that that of a constant predictor. This points to the conclusion that the models are not precise at predicting values included in their training sets, and cannot be trusted at all for the sequences not included in the training sets.

Baseline model performance	Training score	Test score
Frequency model	-0.589	-143.95
Simple one-hot encoding	0.287	-79.46
AAindex	0.332	-98.43
AAindex + Fourier Transform	0.334	-90.6

Table 5.1: Baseline model Q^2 scores for the family dataset.

5.2.2 Model performance on the mutational dataset

Independent testing for this dataset is relatively straightforward since the mutations from the test set are applied to the wild-type sequence, and then the model is used to predict the thermal stability values. The important discovery is that by redistributing the sequences between the training set and the test set, the model scores vary greatly. Multiple such redistributions were made as can be seen in the Table 5.2. The first redistribution named M1 is based on time when the data was obtained. In the training set, there are mutants that were explored from the beginning of the experiment and in the test set there are mutants that were created at the end, taking into account information gathered throughout the

experiment. The second redistribution M2 is based on the number of mutations, where the mutants with smaller number of mutations are in the training set and mutants with more mutations are in the test set, such that the ratio of dataset sizes is 7:3. Redistribution M3 follows a different idea, where the test set contains only mutants that have mutations already observed in the training set but in different combinations. The ratio in this case is not traditional with approximately 6:4 split between training set and test set because only that many variants were required in the trainig set to cover the mutations in the test set. The scores vary greatly ranging from negative to positive numbers close to 1.0. There might be multiple reasons for this behaviour, the first one being that the information required for the label estimation is better included in the well performing redistributions. The second reason might be that the data in the training set does not represent well the data in the test set. M3 redistribution was suppose to show similar use as in the original innov'SAR paper where the test set contained only mutations from the training set, however in the case of M3 the average number of mutations per variant is 4, whereas in the original paper it was 1.

The properties that were selected from the AAindex are listed in Table 5.3 for the Q^2 model and in Table 5.7 for Spearman correlation model. For the Q^2 model the property that contributed the most information contains hydrophobicity coefficient for M1, preference for parallel beta-strands for M2 and normalised frequency of turn for M3. How the model trained on M2 performed on the test set can be seen in Figure 5.2, where a clear linear trend can be seen that suggest that the model captured the correlations well. In the case of Spearman coefficient model, for M1 the property represented normalised frequency of chain reversal, for M2 the normalised frequency of extended structure and for M3 surface composition of amino acids in nuclear proteins.

Redistribution	Training set Q^2	Test set R^2	Training set ρ	Test set ρ
M1	0.769	-5.820	0.859	0.699
M2	0.536	0.004	0.696	0.592
M3	0.762	-0.08	0.895	0.393

Table 5.2: Scores of different dataset redistributions for both scoring functions. The redistribution M1 is based on the time the data was collected. The latest data was used for testing set and the rest for training. M2 is based on the number of mutations. The variants with fewer mutations are in the training set, and the ones with more mutations are in the test set.

Redistribution	AAindex entries selected based on Q^2 scoring function
M1	WILM950101, MAXF760102, NAKH900110
M2	LIFS790102, QIAN880102, RACS770101
M3	PALJ810105, BUNA790103, GEIM800103

Table 5.3: Selected AAindex entries for Q^2 as the scoring function.

Redistribution	AAindex entries selected based on ρ scoring function
M1	TANS770104, FINA910103, CHOP780213
M2	TANS770103, ROBB760105, CHAM830102
M3	FUKS010104, KUMS000103, AURR980120

Table 5.4: Selected AAindex entries for ρ – Spearman correlation as the scoring function.

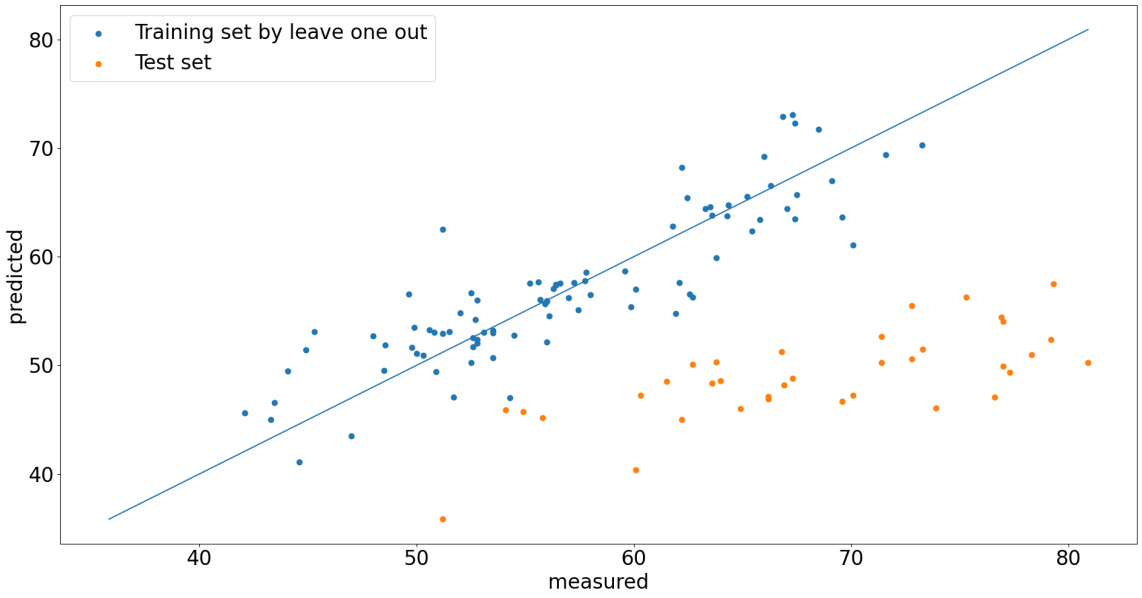


Figure 5.2: Graph produced by the tool while testing the Q^2 model of mutational dataset M1.

5.2.3 Model performance on the family dataset

In this case, the process is more complicated since the input to the predictor is an aligned sequence. For the training data, the alignment was done using the multiple sequence alignment of the whole training set. However, to align the test set and any other sequences to use with the model, the co-alignment of the new sequence to the original sequences from the training set is necessary. The co-alignment process, however, introduces some constraints on the data that can be used with this type of model. First, the maximum length of the sequence is constrained to the length of the aligned training sequences, but for a good fit, it should be even shorter to account for the gaps in the alignment. Second, the co-alignment process might result in some amino acids being excluded form the data, and then it is necessary for the user to consider whether the remaining subsequence is still representative of the initial protein sequence.

Multiple data splits were made with different properties in mind. The first one (F1) is a random redistribution. F2 was created by evenly distributing the total activity between training set and test set, such that the test set has 6 datapoints and F3 was very similar with test set having 8 datapoints.

The results can be seen in Table 5.5. All the redistributions perform poorly for the Q^2 scoring function, with the M1 even having negative test score. But the Spearman correlation

Redistribution	Training set Q^2	Test set R^2	Training set ρ	Test set ρ
F1	0.59	-158.87	0.94	0.76
F2	0.37	0.16	0.61	0.77
F3	0.65	0.51	0.88	0.64

Table 5.5: Model scores for the family dataset redistributions for models trained using both scoring functions. F1 dataset is randomly redistributed and F2 and F3 are based on even representation of label values between training set and test set with different ratios (26:6 and 24:8).

Redistribution	AAindex entries selected based on Q^2 scoring function
F1	GEOR030103, RICJ880109, LEVM780102
F2	FUKS010108, WILM950102, ARGP820103
F3	VINM940104

Table 5.6: AAindex entries selected for the redistributions trained using Q^2 as the scoring function for the family dataset.

performs well in all the cases for the training set and for the test set and it shows great promise in ranking other sequences or mutations.

The indices that were selected for the Q^2 (see Table 5.6) and have the most contribution to the prediction describe Linker propensity in the case of F1 and interior composition of amino acids in nuclear proteins for F2. For F3 dataset the result is more interesting because even though up to three AAindices were combined, model created with only one of them performed better than the three together. The property that it represents is normalized flexibility parameters for each residue surrounded by two rigid neighbours. Indices selected for Spearman coefficient (see Table 5.7) describe normalized frequency of beta sheets, transfer free energy, and flexibility parameter for one rigid neighbour for F1, F2 and F3 respectively.

5.3 Statistical significance testing

The innov'SAR method uses a combinatorial approach to maximize the scoring function. However, this might bring an unintentional bias, since thousands of models are being trained in rather random environment, and the best ones are selected. Thus, we performed the testing of statistical significance to verify that the results obtained using this combinatorial approach did not appear randomly, and there was an actual relationship between the labels and the features.

There are two groups of statistical testing methods with their own strengths and weaknesses. The first group is based on resampling techniques such as permutation or bootstrapping. Here, the goal of testing is to check whether the model, in fact, captures the codependency of labels and features, or whether random reshuffling can yield similar results. However, it assumes that the dataset at hand approximates the underlying distribution of the data, which might not be the case with small datasets. The second group of meth-

Redistribution	AAindex entries selected based on ρ scoring function
F1	PALJ810111, CHOP780211, BUNA790103
F2	SIMZ760101, CHOP780205, WILM950104
F3	KARP850102, OOBM770104, RACS820111

Table 5.7: Selected AAindex entries for ρ – Spearman correlation as the scoring function for the family dataset.

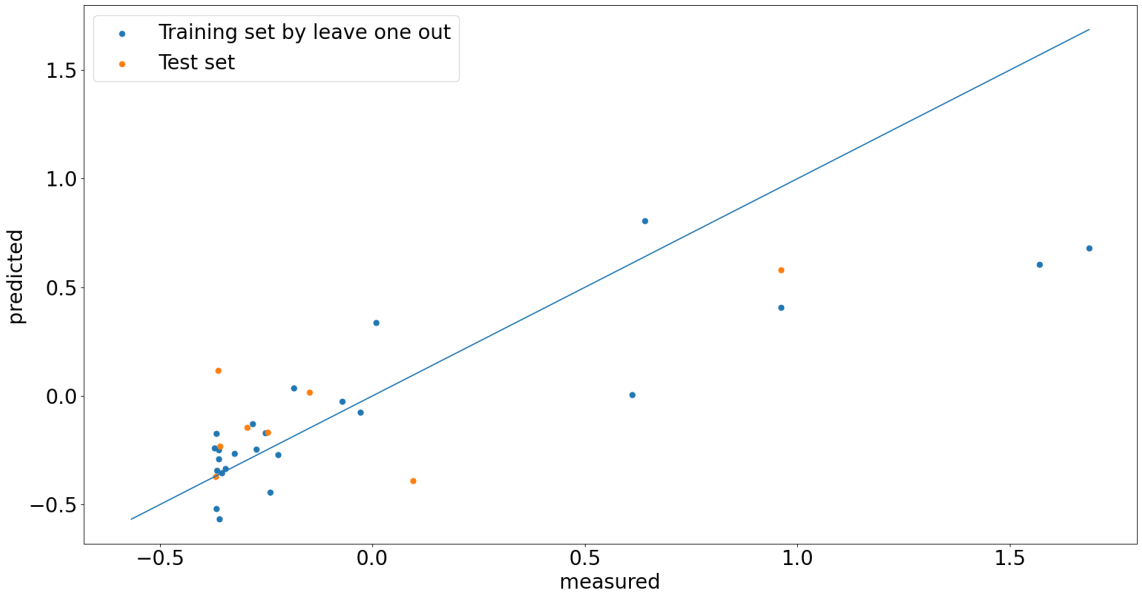


Figure 5.3: Graph produced by the tool while testing the Q^2 model for family dataset F3.

ods is based on Monte Carlo simulations, where the underlying distribution of the data is approximated, and then synthetic datasets are sampled from the approximate distribution.

To compute the validation within a reasonable amount of time the services of Metacentrum were used. The job for computation was submitted for 16 cores and 10GB of memory, from which only 500MB were used. The wall time for running of 500 models was from 14 to 18 hours. Multiple of these simulation runs were performed, each of roughly 500 models.

5.3.1 Permutation testing

In permutation testing, the goal is to test the statistical significance of the score that was obtained for the training data. This is done by shuffling the labels between the entries and retraining using the innov'SAR method many times. The process is illustrated in the Figure 5.4. The goal is to check whether the performance of the model is compromised by the random reshuffling of the labels. The results then show whether the particular value of Q^2 or any other scoring function was due to correctly identified relationship between labels and corresponding sequence features or it was simply by chance.

The permutation testing was run at least 500 times. The results can be seen in the Figure 5.5, and they show good statistical significance of the method, due to most of the points lying left and under the red lines, which mark the model trained on the original data.

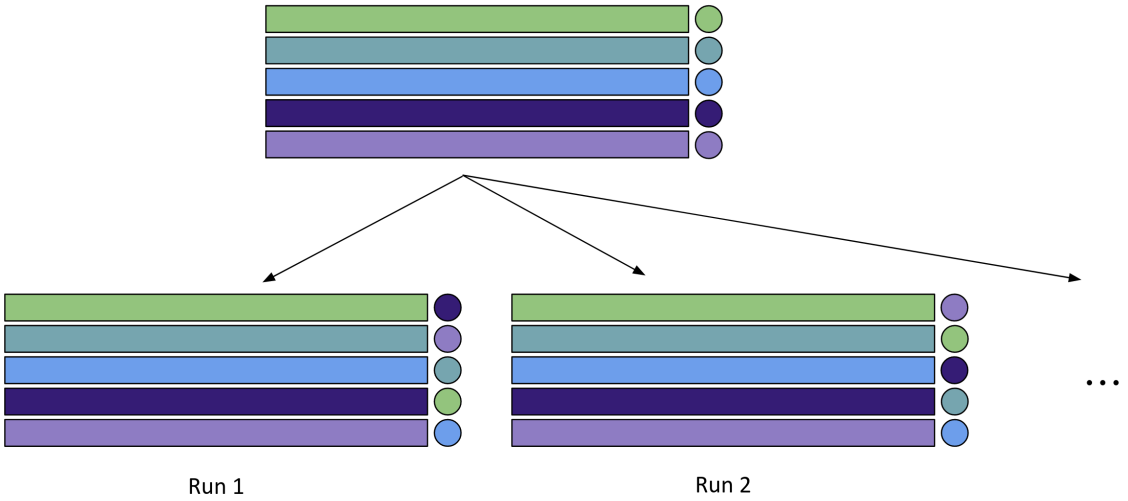


Figure 5.4: Graph showing the process of permutation testing.

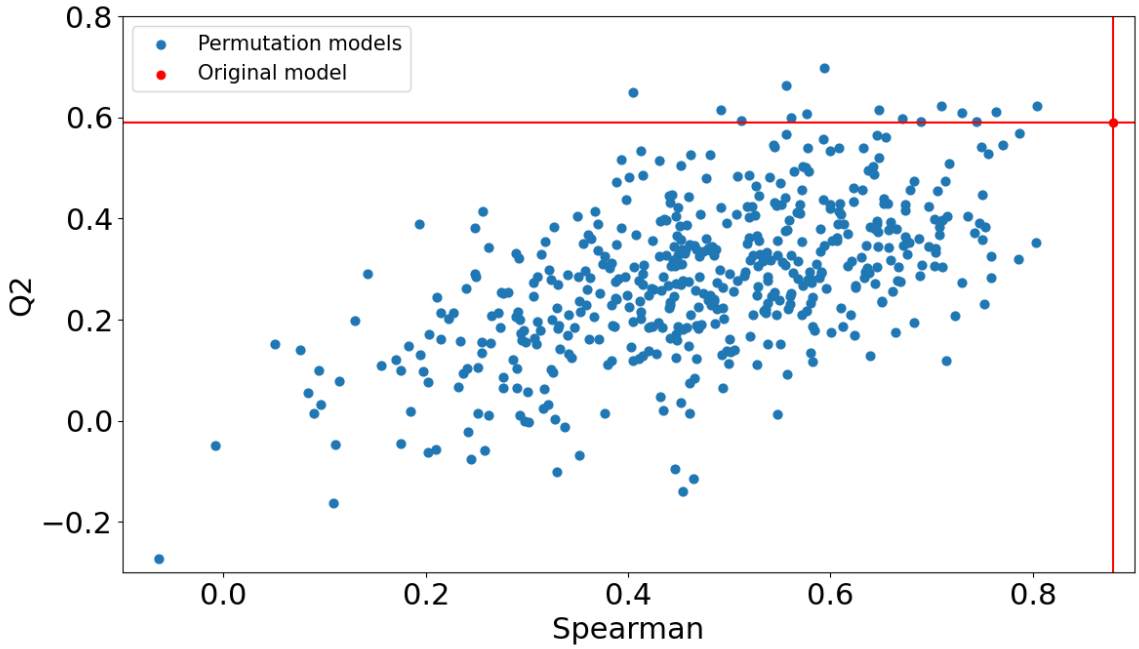


Figure 5.5: The dependency between two different scoring functions for the permutation testing. The red lines indicate the scores of the model for the original data. To verify the statistical significance of the model, most or ideally all other scores should be lower than those of the original model. In this case there are some points (2.6% of all points) above the red line, which means that some models have better scores than the original one, but most of them have lower scores, which means that the original model is statistically significant.

5.3.2 Monte Carlo simulations – Random sampling

The random sampling method is more complicated than the permutation testing. Here, a simple statistical model for each variable and label in the training set is build, and

a sample from the resulting independent distributions is drawn multiple times. The method, therefore, generates its own sequences and corresponding labels. An independent discreet distribution model was chosen for each position in the sequence based on the frequency of each amino acid. These models were then used to sample synthetic sequences that statistically represent the training set as is illustrated in Figure 5.6. For the process of generating labels, the normal distribution model is fitted to the training set labels and used to generate the new artificial labels.

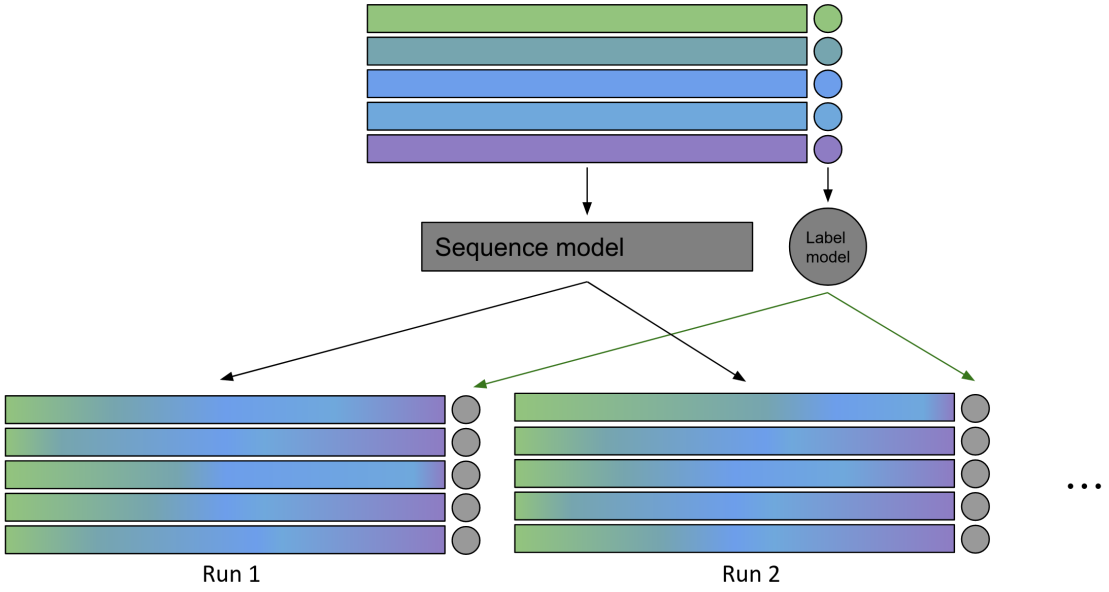


Figure 5.6: Graph showing the process of random sampling. First, for each position in the sequence an even distribution model is fitted and then for the labels an exponential distribution model is fitted. Finally, artificial sequences and labels are build using these models and the innov'SAR method is run repeatedly, while measuring scores.

The results can be seen in Figure 5.7, where, again, most of the results are located below and on the left of the original model, suggesting that the method has good statistical significance.

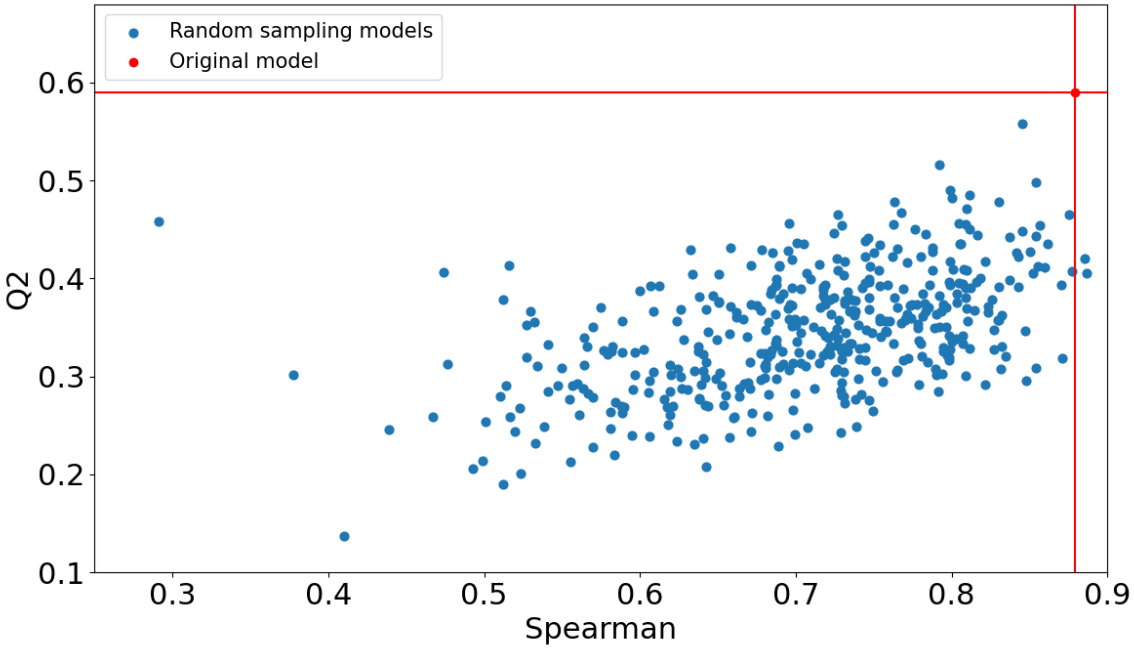


Figure 5.7: The dependency between two different scoring functions for the random sampling method. The red lines indicate the scores of the model for the original data. To verify the statistical significance of the model, most or ideally all other scores should be lower than those of the original model. In this case, there are only three points (0.6% of all points) to the right of the red line that represents models with better Spearman scores, but all other models have lower scores, which means that the performance on the original dataset is unlikely to be by chance.

Chapter 6

Conclusion

The innov'SAR algorithm was implemented and included in an easy-to-use platform suitable for creating a variety of models of protein properties. The models are based solely on protein sequences, not requiring any structural information, which is sometimes hard or even impossible to obtain. This allows the platform to be applied to almost all known protein sequences and protein families. The platform is not bound to any particular protein property, and the user is able to create models for any real-valued features, such as the enzyme activity, thermal stability, solubility, etc.

Apart from the implementation of the innov'SAR method, the possible application was extended to protein families or proteins that have similar primary structures. The testing and prediction is also provided as a part of the platform to easily verify and score the created models. Performance improvement of the algorithm was also introduced in terms of caching and parallelization. The model created by the platform is portable and contains all the information required for testing and prediction.

It is important to keep in mind that the PLS algorithm creates a linear model, which can lead to severe data underfitting in cases where linear dependencies cannot be found. This is the price for hard-to-overfit algorithm, only sequence-based input data, and the possibility to train on small datasets.

To further improve the platform in the future, a graphical interface can be created for easier interaction and data analysis. Improvement in the algorithm can be done by using not only protein spectra, but also general protein embeddings provided by the neural networks pre-trained on large sequence databases, which have already proven to capture a great amount of important information from sequences. In terms of real-world testing, the models that were produced during testing can be used on new sequence libraries to select a few best performing candidates for experimental validation. These will be synthesized and measured in the lab to see if the models have really captured the dependencies and can be applied to unknown proteins. The future work can also include the integration with other tools to improve existing pipelines and support for multiple standardized input file formats, such as alignments and FASTA files.

An interesting discovery was that the independent testing scores might be negative even if the Q^2 scores are relatively high, which points to a conclusion that such models cannot be trusted. However, in the original innov'SAR publication the only scores that are provided are the Q^2 scores without any independent testing scores.

Bibliography

[1] *Beginners guide – MetaCentrum*. February 2022. [Online; accessed 25. Apr. 2022]. Available at: https://wiki.metacentrum.cz/wiki/Beginners_guide.

[2] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. et al. *Molecular Biology of the Cell*. New York, NY, USA: Garland Science, 2002. ISBN 978-0-8153-3218. Available at: https://www.ncbi.nlm.nih.gov/books/NBK21054.

[3] Anderson, P. Pesticide Exposure Linked to Parkinson's, Alzheimer's Disease. *Medscape*. Medscape. january 2014. Available at: https://www.medscape.com/viewarticle/706374.

[4] Brems, M. A One-Stop Shop for Principal Component Analysis - Towards Data Science. *Medium*. Towards Data Science. Dec 2019. Available at: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c.

[5] Cadet, F., Fontaine, N., Li, G., Sanchis, J., Ng Fuk Chong, M. et al. A machine learning approach for reliable prediction of amino acid interactions and its application in the directed evolution of enantioselective enzymes. *Sci. Rep.* Nature Publishing Group. Nov 2018, vol. 8, no. 16757, p. 1–15. DOI: 10.1038/s41598-018-35033-y. ISSN 2045-2322.

[6] Cadet, F., Fontaine, N., Vetrivel, I., Chong, M. N. F., Savriama, O. et al. Application of fourier transform and proteochemometrics principles to protein engineering. *BMC Bioinf.* BioMed Central. Dec 2018, vol. 19, no. 1, p. 1–11. DOI: 10.1186/s12859-018-2407-8. ISSN 1471-2105.

[7] Ferruz, N. and Höcker, B. Towards Controllable Protein design with Conditional Transformers. *ArXiv*. january 2022. DOI: 10.48550/arXiv.2201.07338.

[8] Hasic, H., Buza, E. and Akagic, A. A hybrid method for prediction of protein secondary structure based on multiple artificial neural networks. In: *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2017, p. 1195–1200. DOI: 10.23919/MIPRO.2017.7973605.

[9] Heckmann, C. M. and Paradisi, F. Looking Back: A Short History of the Discovery of Enzymes and How They Became Powerful Chemical Tools. *ChemCatChem*. John Wiley & Sons, Ltd. december 2020, vol. 12, no. 24, p. 6082–6102. DOI: 10.1002/cctc.202001107. ISSN 1867-3880.

[10] Hon, J., Borko, S., Stourac, J., Prokop, Z., Zendulka, J. et al. EnzymeMiner: automated mining of soluble enzymes with diverse structures, catalytic properties and stabilities. *Nucleic Acids Res.* Oxford Academic. july 2020, vol. 48, W1, p. W104–W109. DOI: 10.1093/nar/gkaa372. ISSN 0305-1048.

[11] Hon, J., Borko, S., Stourac, J., Prokop, Z., Zendulka, J. et al. EnzymeMiner: automated mining of soluble enzymes with diverse structures, catalytic properties and stabilities. *Nucleic Acids Res.* Oxford Academic. Jul 2020, vol. 48, W1, p. W104–W109. DOI: 10.1093/nar/gkaa372. ISSN 0305-1048.

[12] Inan, T. Y., Al Hajji, A. and Koseoglu, O. R. Chemometrics-Based Analytical Method Using FTIR Spectroscopic Data To Predict Diesel and Diesel/Diesel Blend Properties. *Energy Fuels.* American Chemical Society. july 2016, vol. 30, no. 7, p. 5525–5536. DOI: 10.1021/acs.energyfuels.6b00731. ISSN 0887-0624.

[13] Jones, C. Another Nobel Prize for Catalysis: Frances Arnold in 2018. *ACS Catal.* American Chemical Society. november 2018, vol. 8, no. 11, p. 10913. DOI: 10.1021/acscatal.8b04266.

[14] Jones, R. W., Meglen, R. R., Hames, B. R. and McClelland, J. F. Chemical Analysis of Wood Chips in Motion Using Thermal-Emission Mid-Infrared Spectroscopy with Projection to Latent Structures Regression. *Anal. Chem.* American Chemical Society. january 2002, vol. 74, no. 2, p. 453–457. DOI: 10.1021/ac0106445. ISSN 0003-2700.

[15] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M. et al. Highly accurate protein structure prediction with AlphaFold. *Nature.* Nature Publishing Group. august 2021, vol. 596, no. 7873, p. 583–589. DOI: 10.1038/s41586-021-03819-2. ISSN 1476-4687.

[16] Lever, J., Krzywinski, M. and Altman, N. Principal component analysis. *Nat. Methods.* Nature Publishing Group. Jun 2017, vol. 14, no. 7, p. 641–642. DOI: 10.1038/nmeth.4346. ISSN 1548-7105.

[17] Lu, H., Diaz, D. J., Czarnecki, N. J., Zhu, C., Kim, W. et al. Machine learning-aided engineering of hydrolases for PET depolymerization. *Nature.* Nature Publishing Group. april 2022, vol. 604, no. 7907, p. 662–667. DOI: 10.1038/s41586-022-04599-z. ISSN 1476-4687.

[18] Mazurenko, S., Prokop, Z. and Damborsky, J. Machine Learning in Enzyme Engineering. *ACS Catal.* American Chemical Society. january 2020, vol. 10, no. 2, p. 1210–1223. DOI: 10.1021/acscatal.9b04321.

[19] Myers, J. and Well, A. *Research Design and Statistical Analysis.* Lawrence Erlbaum Associates, 2003. 503 p. Research Design and Statistical Analysis. ISBN 9780805840377.

[20] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research.* 2011, vol. 12, p. 2825–2830.

[21] Planas Iglesias, J., Marques, S. M., Pinto, G. P., Musil, M., Stourac, J. et al. Computational design of enzymes for biotechnological applications. *Biotechnol. Adv.* Elsevier. march 2021, vol. 47, p. 107696. DOI: 10.1016/j.biotechadv.2021.107696. ISSN 0734-9750.

[22] Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, X. et al. Evaluating Protein Transfer Learning with TAPE. *ArXiv.* Jun 2019.

[23] Robinson, S. L., Smith, M. D., Richman, J. E., Aukema, K. G. and Wackett, L. P. Machine learning-based prediction of activity and substrate specificity for OleA enzymes in the thiolase superfamily. *Synth. Biol.* Oxford Academic. Jan 2020, vol. 5, no. 1. DOI: 10.1093/synbio/ysaa004. ISSN 2397-7000.

[24] Samuel, A. L. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development.* 1959, vol. 3, no. 3, p. 210–229. DOI: 10.1147/rd.33.0210.

[25] The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Res.* Oxford Academic. january 2017, vol. 45, D1, p. D158–D169. DOI: 10.1093/nar/gkw1099. ISSN 0305-1048.

[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention Is All You Need. *ArXiv.* june 2017. DOI: 10.48550/arXiv.1706.03762.

[27] Weng, L. and Spoonamore, J. E. Droplet Microfluidics-Enabled High-Throughput Screening for Protein Engineering. *Micromachines.* 2019, vol. 10, no. 11. DOI: 10.3390/mi10110734. ISSN 2072-666X.

[28] Wold, S., Sjöström, M. and Eriksson, L. PLS-regression: a basic tool of chemometrics. *Chemom. Intell. Lab. Syst.* Elsevier. Oct 2001, vol. 58, no. 2, p. 109–130. DOI: 10.1016/S0169-7439(01)00155-1. ISSN 0169-7439.

[29] Yang, M., Fehl, C., Lees, K. V., Lim, E.-K., Offen, W. A. et al. Functional and informatics analysis enables glycosyltransferase activity prediction. *Nat. Chem. Biol.* Nature Publishing Group. Dec 2018, vol. 14, no. 12, p. 1109–1117. DOI: 10.1038/s41589-018-0154-9. ISSN 1552-4469.

Appendix A

Contents of the included storage media

- `/`
  - `tool/` — Platform implementation
    - `src/` — Source code files
      - `tool.py` — Main script for executing the platform
    - `requirements.txt` — List of python dependencies that can be used with pip tool
    - `README.md` — User documentation for installing dependencies and tool use
  - `data/` — Used datasets and redistributions in subfolders
  - `models/` — Model files created from the datasets
  - `thesis_latex/` — LaTeX source files for this thesis
  - `thesis.pdf` — This thesis in PDF format

Appendix B

File formats

All the files are in the csv format where the columns are separated by a colon.

Sequence name	Protein sequence	Label value
DthA	———MATDRGLEISSAFPFE...	-0.2735666958450363
DthB	———MAYDSSQLISAEFPFK...	-0.07102124804808232
DchA	————MAVFDEISSDFPFE...	-0.1482152816186778
...	...	...

Table B.1: File format of the family input file.

Mutant name	List of mutations	Label value	Method
100	I136L;V184E;V197E	51.5	CD
100	I136L;V184E;V197E	48.5	DSC
101	E20S;F80R;A155P	56.9	CD
...	...	...	...

Table B.2: File format of the mutants input file. The last column **Method** is optional.