Machine learning to understand and engineer the structural kinome

Ivan Reveguk

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAX151

Thèse de doctorat

Machine learning to understand and engineer the structural kinome

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'Ecole polytechnique

École doctorale n°626 Ecole Doctorale de l'Institut Polytechnique de Paris (ED IP PARIS)
Spécialité de doctorat: Biologie

Thèse présentée et soutenue à Palaiseau, le 08/11/2023, par

Ivan REVEGUK

Composition du Jury :

Florence D'ALCHE-BUC Professeure, Télécom Paris, Laboratoire Traitement et Communication de l'Information	Président
Vincent ZOETE Associate Professor, Université de Lausanne, Ludwig Institute for Cancer Research, Departement of oncology, Laboratory V. Zoete	Rapporteur
Roland L. DUNBRACK Professor, Fox Chase Cancer Institute	Rapporteur
Sergei GRUDININ Chargé de recherche, Laboratoire Jean Kuntzmann, CNRS Grenoble	Examinateur
Thomas SIMONSON Director, Ecole Polytechnique, Laboratoire de Biologie Structurale de la Cellule, UMR 7654	Directeur de thèse

Abstract

Protein kinases (PKs) comprise one of the most ancient and ubiquitous enzyme groups deeply embedded in a cell's molecular machinery. PKs transfer a γ-phosphate group from a nucleotide triphosphate (NTP) to a hydroxyl group of a specific substrate. They are key signaling networks' actors and important drug targets during cancer treatment. PKs cycle between active and inactive conformations, distinguished by a few elements within the catalytic domain. One is the activation loop (AL), whose conserved DFG motif can occupy DFG-in, DFG-out, and some rarer conformations. Annotation and classification of the structural kinome are essential, as different inhibitors and activators can target different conformations. Namely, DFG-out conformation typically confers greater inhibitor selectivity and efficiency, as it impedes substrate binding, and fewer kinases can achieve it. Despite being pivotal to drug design efforts, little is known regarding which features shape the PK conformational landscape, both on sequence and structure levels. This work constitutes a systematic attempt to elucidate them via careful data curation and mining.

Machine learning (ML) has become an increasingly popular tool in structural biology. A branch of ML known as Interpretable Machine Learning (IML) provides techniques for explaining ML models, however complex. This work utilized such techniques on PK sequences and structures to understand features distinguishing between active/inactive and various DFG states. Still, even state-of-the-art methodology will fall short if the data are inaccurate. Despite decades of research and development, we still lack reliable, accurate, general-purpose tools to characterize and relate protein sequences and structures. Here, we addressed this problem by creating an open-source Python library lXtractor.

Using lXtractor, we assembled the largest structural kinome collection to date, encompassing nearly ten thousand domains. Their sequences and struc-

tures were represented by an extensive series of numerical descriptors anchored to a single reference sequence profile covering the whole PK domain. We annotated DFG conformations of over 90% of the assembled domains using a semi-supervised clustering-based approach. Moreover, the approach enabled the assessment of the accuracy of the existing DFG labeling strategies, which revealed their frequent inadequacy in labeling minor DFG states, leading to internal inconsistency and conflicts with each other. As an improvement, we trained ensembles of decision trees predicting active/inactive and DFG states with near-perfect accuracy. The obtained models were then interpreted to reveal the most prominent structural features distinguishing these conformational states. They were associated with well-known structural regions and are readily interpretable by structural biologists. Furthermore, the ML models were applied to annotate structures published by a recent AlphaFold2 tool. This demonstrated that AlphaFold2 predictions are likely biased towards the DFG-in conformation.

With accurate structural annotations, we addressed the problem of sequence elements shaping DFG conformational preferences in the inactive state. Namely, we sought to identify a handful of residues harbored by a PK domain that could explain why a particular kinase has a propensity towards DFG-in or DFG-out states. In doing so, we first assessed how this propensity has evolved throughout a PK domain evolution. This revealed that the ability to perform the DFG-in→DFG-out transition is sporadically distributed and does not adhere to a discernable propagation pattern. Thus, we treated tyrosine and serine-threonine kinases (TKs and STKs) separately. We first confirmed that our ML pipeline can efficiently distinguish TK and STK sequences, obtaining perfect classifier for this "test problem." The models relied on sequence positions that could be structurally interpreted and relate to different properties of tyrosine and serine/threonine phosphoacceptor residues. Following this, the models were trained to discriminate between sequences with DFG-in and DFG-out propensities, where labels were derived using ligand-free structures and datasets were complemented by orthologs lacking structural data. The obtained models were highly accurate, making only a handful of errors across 1330 STK and 1000 TK sequences. For TKs, the most prominent positions the models relied on could be structurally rationalized as reinforcing the AL in the DFG-out conformation. This insight corroborated existing studies and

mutagenesis data. STK models were more challenging to interpret, with diverging position sets across different subfamilies being the most likely scenario. Thus, natural DFG-out propensity may be a convergent trait. Without selective pressure, kinases may have sequence determinants dependent on a broader evolutionary context.

Overall, this work provided an accurate assessment of the PK conformational landscape and improved our understanding of the structural kinome. The PK data collection, sequence, and structure-based ML models and their predictions were made publicly available. These can be used to develop more accurate PK inhibition strategies and annotate conformations and propensities of new experimental and predicted PK structures and sequences. The tools developed in this work were also published as open-source packages written in Python: (1) *KinActive*, encompassing ML models, a pipeline to train and evaluate them, and a protocol to assemble a PK data collection; (2) *lXtractor*, to assemble, manage, and characterize sequence/structure data collections; (3) *eBoruta*, a tool for model-agnostic feature selection using the Boruta algorithm. Cumulatively, we expect them to increase the transparency and reproducibility of data mining in structural biology.

Résumé

Les protéines kinases (PK) constituent l'un des groupes d'enzymes les plus anciens et les plus omniprésents, profondément ancrés dans la machinerie moléculaire d'une cellule. Les PK transfèrent un groupe γ-phosphate d'un nucléotide triphosphate (NTP) à un groupe hydroxyle d'un substrat spécifique. Ce sont des acteurs clés des réseaux de signalisation et des cibles médicamenteuses importantes lors du traitement du cancer. Les PK alternent entre les conformations actives et inactives, distinguées par quelques éléments dans le domaine catalytique. L'une est la boucle d'activation (AL), dont le motif DFG conservé peut occuper DFG-in, DFG-out et certaines conformations plus rares. L'annotation et la classification du kinome structurel sont essentielles, car différents inhibiteurs et activateurs peuvent cibler différentes conformations. À savoir, la conformation DFG-out confère généralement une plus grande sélectivité et efficacité aux inhibiteurs, car elle empêche la liaison au substrat, et moins de kinases peuvent y parvenir. Bien qu'elles soient essentielles aux efforts de conception de médicaments, on sait peu de choses sur les caractéristiques qui façonnent le paysage conformationnel PK, tant au niveau de la séquence que de la structure. Ce travail constitue une tentative systématique de les élucider via une conservation et une exploration minutieuses des données.

L'apprentissage automatique (ML) est devenu un outil de plus en plus populaire en biologie structurale. Une branche du ML connue sous le nom d'Interpretable Machine Learning (IML) fournit des techniques pour expliquer les modèles de ML, aussi complexes soient-ils. Ce travail a utilisé de telles techniques sur les séquences et les structures PK pour comprendre les caractéristiques distinguant les états actif/inactif et divers états DFG. Pourtant, même les méthodes les plus avancées ne suffiront pas si les données sont inexactes. Malgré des décennies de recherche et de développement, nous man-

quons toujours d'outils fiables, précis et polyvalents pour caractériser et relier les séquences et structures protéiques. Ici, nous avons résolu ce problème en créant une bibliothèque Python open source lXtractor.

Grâce à lXtractor, nous avons rassemblé la plus grande collection de kinomes structurels à ce jour, englobant près de dix mille domaines. Leurs séquences et structures étaient représentées par une vaste série de descripteurs numériques ancrés à un seul profil de séquence de référence couvrant l'ensemble du domaine PK. Nous avons annoté les conformations DFG de plus de 90% des domaines assemblés en utilisant une approche basée sur le clustering semi-supervisé. De plus, l'approche a permis d'évaluer l'exactitude des stratégies d'étiquetage DFG existantes, qui ont révélé leur insuffisance fréquente dans l'étiquetage des États mineurs DFG, conduisant à des incohérences internes et à des conflits les uns avec les autres. En guise d'amélioration, nous avons formé des ensembles d'arbres de décision prédisant les états actif/inactif et DFG avec une précision presque parfaite. Les modèles obtenus ont ensuite été interprétés pour révéler les caractéristiques structurelles les plus importantes distinguant ces états conformationnels. Ils étaient associés à des régions structurelles bien connues et sont facilement interprétables par les biologistes structuraux. De plus, les modèles ML ont été appliqués pour annoter des structures publiées par un outil AlphaFold2 récent. Cela démontre que les prédictions d'AlphaFold2 sont probablement biaisées en faveur de la conformation DFG-in.

Avec des annotations structurelles précises, nous avons abordé le problème des éléments de séquence façonnant les préférences conformationnelles du DFG à l'état inactif. À savoir, nous avons cherché à identifier une poignée de résidus hébergés par un domaine PK qui pourraient expliquer pourquoi une kinase particulière a une propension aux états DFG-in ou DFG-out. Ce faisant, nous avons d'abord évalué comment cette propension a évolué tout au long de l'évolution d'un domaine PK. Cela a révélé que la capacité à effectuer la transition DFG-in→DFG-out est distribuée de manière sporadique et n'adhère pas à un modèle de propagation discernable. Ainsi, nous avons traité séparément les tyrosine et sérine-thréonine kinases (TK et STK). Nous avons d'abord confirmé que notre pipeline ML peut distinguer efficacement les séquences TK et STK, obtenant ainsi un classificateur parfait pour ce "problème de test". Les modèles reposaient sur des positions de séquence qui pouvaient être interprétées structurellement et liées à différentes propriétés des

résidus phosphoaccepteurs tyrosine et sérine/thréonine. Suite à cela, les modèles ont été entraînés à distinguer les séquences avec des propensions DFG-in et DFG-out, où les étiquettes ont été dérivées à l'aide de structures sans ligand et les ensembles de données ont été complétés par des orthologues manquant de données structurelles. Les modèles obtenus étaient très précis, ne commettant que quelques erreurs sur 1330 séquences STK et 1000 séquences TK. Pour les savoirs traditionnels, les positions les plus importantes sur lesquelles s'appuient les modèles pourraient être structurellement rationalisées comme renforçant l'AL dans la conformation DFG-out. Cette idée a corroboré les études existantes et les données de mutagenèse. Les modèles STK étaient plus difficiles à interpréter, le scénario le plus probable étant des ensembles de positions divergentes entre différentes sous-familles. Ainsi, la propension naturelle à sortir du DFG peut être un trait convergent. Sans pression sélective, les kinases peuvent avoir des déterminants de séquence dépendant d'un contexte évolutif plus large.

Dans l'ensemble, ce travail a fourni une évaluation précise du paysage conformationnel PK et amélioré notre compréhension du kinome structurel. Les modèles ML de collecte de données PK, de séquence et de structure ainsi que leurs prédictions ont été rendus publics. Ceux-ci peuvent être utilisés pour développer des stratégies d'inhibition PK plus précises et annoter les conformations et les propensions de nouvelles structures et séquences PK expérimentales et prédites. Les outils développés dans ce travail ont également été publiés sous forme de packages open source écrits en Python: (1) *KinActive*, englobant des modèles ML, un pipeline pour les entraîner et les évaluer, et un protocole pour assembler une collection de données PK; (2) *lXtractor*, pour assembler, gérer et caractériser des collections de données de séquence/structure; (3) *eBoruta*, un outil de sélection de caractéristiques indépendantes du modèle utilisant l'algorithme de Boruta. Cumulativement, nous espérons qu'ils augmenteront la transparence et la reproductibilité de l'exploration de données en biologie structurale.

Acknowledgements

I am profoundly grateful to Prof. Thomas Simonson, who has steered the direction of this thesis over the course of three eventful years. His direct and indirect guidance has taught me to balance between innovation and best practices, cherish deep conceptual understanding, and approach my work in a structured and critical way. I value our intense discussions, his patience, and his willingness to delve into less familiar topics. I am thankful for the academic and personal freedom granted and, overall, for assisting me in taking this vital step toward scientific maturity.

I extend this appreciation to all the jury members who exhibited a genuine interest in my research and dedicated their time to its assessment. I am also grateful to Prof. Roland L. Dunbrack for his astute observations during the mid-defense.

Lastly, I wish to express my heartfelt thanks to my family, whose unwavering support and mere existence warrant immeasurable gratitude.

Contents

Chapter 1

Introduction to Machine Learning

Machine learning (ML) is a field that focuses on the development of algorithms learning from data to make predictions. It can be viewed as a modern extension of applied statistics, emphasizing predictive accuracy and computational efficiency over traditional statistical inference. The term was coined by Arthur Samuel in 1959 [1], reflecting the idea that computers could be taught to learn and adapt without being explicitly programmed for specific tasks. Pioneering work by researchers such as Frank Rosenblatt [2] laid the groundwork for subsequent advancements in artificial neural networks and other machine learning techniques.

In the following decades, the field of machine learning has grown exponentially, benefiting from technological innovation and vast amounts of data. Today, ML plays a vital role in various domains, ranging from healthcare and finance to natural language processing and computer vision, transforming the way we analyze and interpret data [3].

An important distinction comes up when comparing ML to Data Mining (DM). While ML models may be used as part of the data mining process, DM aims to uncover underlying structures and relationships within the data itself. In essence, ML is often about prediction, while DM focuses on description and understanding. Both fields play a crucial role in modern data science, each contributing unique perspectives and tools to the analysis and interpretation of complex data [4].

This chapter aims to describe general ML principles and provide back-

ground for ML techniques employed in subsequent chapters. We will begin by delving into foundational concepts, including generalization, performance estimation, and cross-validation. Next, we will introduce how ML approaches can be categorized and discuss several models in greater depth. Building on this foundation, we'll lay out techniques allowing to interpret trained models, bridging the gap between DM and ML. Special attention will be given to methods such as LIME and SHAP. Finally, we'll discuss general principles of feature selection and provide an in-depth description of the Boruta algorithm.

1.1 Basic concepts

1.1.1 The core ML approach

An ML model can be conceptualized as a sophisticated function f with both trainable parameters (θ) and non-trainable parameters (λ). The trainable parameters are central to the model, adapting and evolving during the training process. Governed by a specific algorithm, the model "learns" to adjust θ based on the input data X and, in supervised learning (see below), the corresponding target values y.

The training algorithm relies on particular criteria to assess the quality of predictions. In many models, this criterion is defined by a loss function $l(y, y')$ that quantifies the difference between the target values and predicted values y', with the goal of minimizing this difference. In contrast, models like decision trees utilize gain functions, such as reduction in entropy or variance, which they aim to maximize.

The non-trainable parameters, on the other hand, are predetermined settings that govern the structure and behavior of the model but do not change during training. These settings can be informed by a user's knowledge, iteratively adjusted through trial and error, or fine-tuned via a specific optimization algorithm.

1.1.2 Learning paradigms

Machine Learning (ML) models can be broadly categorized based on the nature of the output y and the availability of target values during training. This cat-

egorization leads to three main learning paradigms: supervised, unsupervised, and semi-supervised learning:

- **Supervised Learning:** In supervised learning, the target values y are known and provided during training. The model learns to map input data X to these target values, adjusting its parameters to minimize the difference between predicted and actual outputs. Depending on the nature of y, supervised learning can be further divided into:
  - **Classification:** If y represents discrete categories. The number of categories determines whether the problem is binary (two categories) or multiclass (more than two categories).
  - **Regression:** If y represents continuous values.
- **Unsupervised Learning:** In unsupervised learning, the target values y are unknown or not provided during training. The model focuses on discovering underlying patterns, structures, or relationships within the input data X without guidance from specific target values. Common tasks include clustering, dimensionality reduction, and anomaly detection.
- **Semi-Supervised Learning:** Semi-supervised learning falls between supervised and unsupervised learning. It utilizes both labeled data (with known y) and unlabeled data (without y). This approach leverages the information in the unlabeled data to enhance the learning process.

The nature of the learning problem informs the choice of algorithms, evaluation metrics, and methodologies, shaping the design and implementation of ML solutions. Other learning paradigms, such as reinforcement learning, also exist but fall outside the scope of this study, as they do not align as closely with the data mining-oriented framework that guides our exploration. Here, we focus more on supervised classification paradigm.

1.1.3 Generalization, Bias, and Variance

Generalization in ML refers to a model's ability to perform well on unseen data, capturing the underlying patterns without being influenced by noise or random fluctuations. A model that generalizes well is essentially the antithesis

of an overfitted model, which performs well on the training data but poorly on new, unseen data.

Generalization is closely related to the concepts of bias and variance [5], which provide insights into a model's potential underperformance or overfitting:

- **Bias:** Bias is the systematic error introduced by simplifying a real-world problem. A model with high bias makes strong assumptions about the underlying data and may oversimplify the problem, leading to underfitting.
- **Variance:** Variance measures a model's sensitivity to small fluctuations in the training data. A model with high variance pays too much attention to the noise or random variations in the training data, leading to overfitting.
- **Bias-Variance Tradeoff:** Balancing bias and variance is a fundamental challenge in ML. Reducing bias may increase variance, and vice versa (Fig. 1.1). A well-balanced model minimizes both to achieve good generalization.

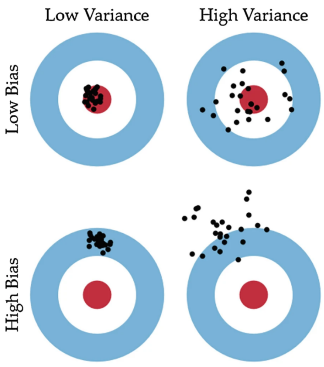

Figure 1.1: An illustration of the relationships between a model's bias and variance. The figure is taken from [6].

The bias-variance tradeoff is typically linked to the discussion on model complexities. Overparametrization, or the use of an excessive number of parameters, increases the model's complexity and may lead to a model that "memorizes" its input, exerting high variance and becoming prone to overfitting [7]. However, with the emergence of large deep learning architectures, some researchers are starting to revisit the traditional understanding, exploring how these complex models can sometimes defy conventional wisdom [8].

1.1.4 Performance Estimation and Cross-Validation

To evaluate a model's generalization capacity, one uses a performance estimate. In the context of supervised learning, this performance estimate evaluates the difference between predicted and actual values. Mathematically, this is represented as a function $\rho(y, y')$. Depending on the metric, one may seek to either maximize or minimize this function. For instance, Mean-Squared Error (MSE) is commonly minimized, while Accuracy is maximized.

An important consideration is the data used for evaluating performance. Using the same data for both training and testing fails to capture a model's ability to generalize to unseen data. To address this, one typically employs a separate test set or a procedure known as cross-validation (CV) [9]. In CV, the training instances are randomly partitioned into K non-overlapping folds. The model is trained on $K-1$ of these folds and tested on the remaining one. This process is repeated K times, with each fold serving as the test set exactly once, to produce an averaged ρ^{CV}. When K equals the size of the dataset, this strategy is specifically known as Leave-One-Out Cross-Validation (LOOCV).

Using these strategies mimics a real-world scenario, where a model is applied to unseen data, leading to conclusion regarding its generalization capacity. However, CV provides a more robust estimate of a model's performance than a single test set, reducing the risk of an overly optimistic or pessimistic assessment.

Apart from evaluating generalization, CV has broader applications, such as:

- **Hyperparameter Tuning**: The initial settings of an ML algorithm can significantly impact its performance. Searching for optimal parameters is often framed as an optimization problem [10], where one seeks hyperparameter combinations that maximize/minimize ρ^{CV}. We'll explore this in more detail below.
- **Model Selection**: CV is also used for comparing different models. Combining model selection with hyperparameter tuning poses an even greater challenge. Some methods like greedy K-fold CV can mitigate the computational burden [11].
- **Unsupervised Problems**: Surprisingly, CV can also be applied to unsupervised problems like clustering, where it can help determine the

number of clusters based on intra-cluster similarity or inter-cluster dissimilarity [12, 13].

In practice, applying CV can be nuanced and may lead to overfitting during model selection. Namely, it can sometimes lead to ”optimization bias” or ”selection bias,” especially when the dataset is small [14]. This risk can be mitigated through techniques like early stopping and regularization. Early stopping refers to halting the training when a chosen performance metrics deteriorates. Regularization, in turn, is penalizing a model's complexity, such as very large values of trainable parameters [15].

The nature of the underlying data can also influence the CV strategy. Namely, using random sampling of training instances when constructing CV folds may be inadequate in some scenarios, requiring to employ some external similarity measure [16]. In such cases, CV may operate on pre-computed clusters of samples, enclosing sufficiently similar instances. This is especially relevant when training instances are (potentially related) protein sequences [17].

Finally, if the data are plentiful, a common practice to avoid overfitting is splitting the data modelling and testing subsets [6]. The former is used for model selection, hyperparameter tuning, and fitting the final model, while the latter is used to produce a final performance assessment.

1.1.5 Hyperparameter Tuning

Above, we've introduced hyperparameter optimization (HPO) as a combinatorial problem of searching for an ML algorithm's settings most suitable for the learning objective [18]. Assuming that higher values of ρ indicate a better model, one can formulate HPO as an optimization problem $\underset{\lambda}{\operatorname{argmax}}\ \rho$ (or $\underset{\lambda}{\operatorname{argmax}}\ \rho^{CV}$), where λ is a space of possible hyperparameters. In practice, the “tunability” of different hyperparameters may vary [19]. Thus, informed by prior knowledge, λ is initially constrained to some sensible value ranges.

Various strategies exist for searches over λ [20], including:

- **Grid Search**: An exhaustive evaluation of all possible combinations. The guarantee of convergence is offset by the computational cost.
- **Random Grid Search**: To mitigate the computational burden, a random grid search can be employed. Instead of evaluating all combinations,

a random subset is chosen, offering a trade-off between computational cost and optimization quality.

- **Genetic Algorithms**: These are heuristic search algorithms inspired by the process of natural selection. They are more efficient than grid search but may require fine-tuning of their own parameters.
- **Bayesian Optimization**: This probabilistic model-based optimization technique is particularly effective for high-dimensional hyperparameter spaces. It builds a probabilistic model of the objective function and uses it to select the most promising hyperparameters to evaluate in the true objective function.

Among these, Bayesian Optimization (BO) techniques have gained prominence for their efficiency and effectiveness. The core idea behind BO is to build a probabilistic model $P(\rho|\lambda)$ that estimates the distribution of ρ given a particular set of hyperparameters λ. This model is iteratively updated based on newly evaluated points in λ throughout the optimization process [21].

Another key ingredient to BO is the acquisition function, which guides the sampling process. It takes $P(\rho|\lambda)$ as input and produces a utility value for each point in λ. The nature of the acquisition function can vary, but certain formulations like the Expected Improvement (see below) aim to balance between:

- **Exploration**: Exploring regions with higher uncertainty in $P(\rho|\lambda)$ to ensure that the algorithm doesn't miss any potentially optimal regions.
- **Exploitation**: Exploiting regions where the expected performance, as indicated by the mean of $P(\rho|\lambda)$, is high.

A common example of an acquisition function is the Expected Improvement (EI). For the best performance ρ^* observed so far, it is defined as

$$EI(\lambda) = \mathbb{E}[\max(\rho - \rho^*, 0)] \tag{1.1}$$

The EI function aims to maximize the expected improvement over the best-known point ρ^* by considering both the predicted mean and variance of $P(\rho|\lambda)$. Specifically, it produces a new point λ^* such that

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}}\ EI(\lambda) \tag{1.2}$$

The BO algorithm iteratively performs the following steps:

1. Use the probabilistic model to find the hyperparameter set λ^* that maximizes ρ.
2. Evaluate the true objective function $\rho(\lambda^*)$ to obtain a new data point.
3. Update the probabilistic model $P(\rho|\lambda)$ with the new data point.

Optuna is a popular framework that implements Bayesian Optimization. It extends the basic BO algorithm by incorporating features like pruning of unpromising trials and parallelization. Optuna is adaptable to various types of hyperparameter spaces, including those with categorical variables, and allows for easy customization of acquisition functions, making it a flexible tool for hyperparameter optimization [22].

1.2 A Brief Overview of ML Algorithms

Above, we've introduced briefly the foundational ML principles, laying out the general frameworks and methodologies that are broadly applicable across a variety of supervised learning problems. However, the theoretical underpinnings are often adapted and specialized to fit the particular models at hand. As of today, a plethora of models are available, each with its unique strengths, weaknesses, and assumptions. The choice of a model is rarely arbitrary; it is closely guided by the learning objective and the nature of the data. In the following sections, we will delve into some of these models, exploring their inner workings, applicability, and how they embody the general principles we've discussed.

1.2.1 The diversity of ML models

Machine learning algorithms can be broadly categorized based on several key characteristics:

1. **Linear vs. Non-linear:** Linear models, such as Logistic Regression, assume a linear relationship between the input features and the output. Non-linear models like Decision Trees and Neural Networks do not make this assumption and can capture complex relationships in the data.

2. **Parametric vs. Non-parametric:** Parametric models make certain assumptions about the data distribution and have a fixed number of parameters. Non-parametric models, like K-Nearest Neighbors, make fewer assumptions and can have an infinite number of parameters.
3. **Lazy vs. Eager:** Lazy learners, such as K-Nearest Neighbors, defer the decision-making until a new instance is encountered. Eager learners like Decision Trees and Neural Networks make decisions during training.
4. **Generative vs. Discriminative:** Generative models like Naive Bayes learn the joint probability distribution $P(X, Y)$ and make predictions by estimating $P(Y|X)$. Discriminative models like Logistic Regression directly learn the boundary between classes.
5. **Interpretable vs. Black-box:** Interpretable models like Decision Trees and Linear Regression are easy to understand and interpret. Black-box models like Neural Networks are more complex and harder to interpret.

Understanding these categorizations can guide the choice of algorithm for a given problem, ensuring that the chosen model is well-suited to the task at hand. Below, we'll elaborate on some of these approaches in greater detail.

1.2.2 Logistic Regression

Logistic Regression (LR) is perhaps the simplest example of a binary classifier. In binary classification, the response variable y is a Bernoulli random variable with parameter p. In linear regression, the expectation of y is modeled via a linear combination of data X and trainable parameters θ, i.e., $p = X^T\theta$. In logistic regression, a logistic link function is applied to p, resulting in

$$\log \frac{p}{1-p} = X^T\theta. \tag{1.3}$$

Solving this equation yields

$$p = \frac{1}{1 + e^{-X^T\theta}} \tag{1.4}$$

The log-likelihood function to be maximized with respect to θ is given by:

$$\sum_{i=1}^{N} (y_i \log(p) + (1 - y_i) \log(1 - p)) - Cr(\theta) \tag{1.5}$$

where $r(\theta)$ is the regularization term that penalizes model complexity to prevent overfitting, and C controls the regularization strength. For example, in the elastic net regularization method, a combination of l_2 and l_1 regularization is used, balanced by a parameter α:

$$r(\theta) = \frac{1 - \alpha}{2}\theta^T\theta + \alpha||\theta||_1 \tag{1.6}$$

where $||\theta||_1$ is the sum of the absolute values of the parameters.

1.2.3 Illustrating Core ML Principles Using Logistic Regression

LR serves as a suitable choice to illustrate the core principles introduced above. For this, we'll generate a synthetic dataset X with 100 instances, where each instance has two features x_1 and x_2. The dataset is divided into two classes, with 70 and 30 instances belonging to each class (0 and 1). The feature matrix X is defined as:

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ \vdots & \vdots \\ x_{100,1} & x_{100,2} \end{pmatrix}$$

For the sake of simplicity, we assume that x_1 and x_2 are normally distributed:

- For class 0: $x_1 \sim \mathcal{N}(2, 1)$ and $x_2 \sim \mathcal{N}(3, 1)$
- For class 1: $x_1 \sim \mathcal{N}(4, 1)$ and $x_2 \sim \mathcal{N}(5, 1)$

This setup provides a dataset where the two classes are somewhat separable, making it suitable for logistic regression (Fig. 1.2a).

As the dataset is imbalanced, using regular Accuracy may provide an inaccurate assessment of the predictive power. Thus, we opt for using Balanced

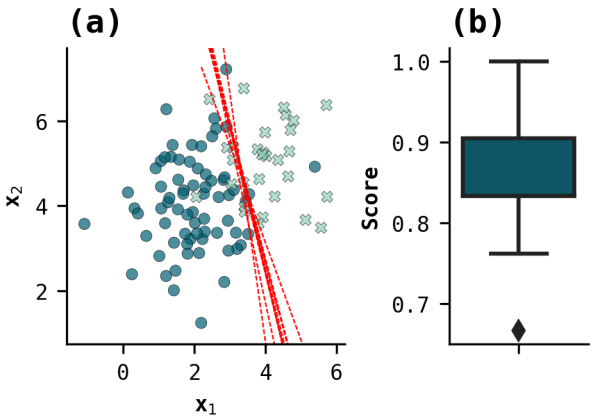

Figure 1.2: Sampled dataset, decision boundaries, and CV performance distribution. **(a)** The data points of X with classes denoted by color: blue circles are class 0 instances, whereas light cyan crosses depict class 1 instances. Red lines depict decision boundaries of LR trained on different CV folds. **(b)** The distribution of Balanced Accuracy across the CV folds. The algorithm was trained using the optimized hyperparameters as described in the main text.

Accuracy (BAC) as the performance metric. BAC is defined as the arithmetic mean of Sensitivity (True Positive Rate) and Specificity (True Negative Rate):

$$\text{BAC} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP}+\text{FN}} + \frac{\text{TN}}{\text{TN}+\text{FP}}\right) \tag{1.7}$$

Here, TP, TN, FP, and FN stand for True Positives, True Negatives, False Positives, and False Negatives, respectively. Balanced Accuracy effectively handles class imbalance by giving equal weight to both Sensitivity and Specificity, ensuring that both classes are equally important in the evaluation.

First, we'll tune two hyperparameters controlling regularization: $\frac{1}{C}$ and α, where the former is an inverse of C introduced above, with smaller values implying stronger regularization. With this, we'll set up the search space λ as a 10×10 grid with ten values for each parameter, amounting to 100 different combinations:

- $\frac{1}{C}$ ranges between 0.05 and 2.0, with values equally spaced in this range.
- α ranges between 0 and 1, with values equally spaced in this range.

Next, we'll set up an exhaustive grid search over λ. For this, we'll generate ten stratified CV folds, preserving the class ratios observed in X within each fold. For each $(\frac{1}{C}, \alpha)$ combination, we'll fit the LR algorithm on the training

fold and compute the BAC on the training fold.

As illustrated in Fig. 1.3a, the performance of the Logistic Regression model deteriorated as the regularization strength increased (i.e., as $\frac{1}{C}$ decreased), suggesting that the model became too biased. Conversely, as $\frac{1}{C}$ increased, indicating weaker regularization, the performance stabilized. This suggests that the model benefits from additional complexity, without showing signs of high variance or overfitting.

In turn, the α had marginal impact on the performance across the whole explored value range. The best performance (BAC=0.85) was achieved for a combination of $\frac{1}{C}$=1.13 and α=0.88 (Fig. 1.3b). The latter indicates that the model relied primarily on the l_1 penalty for regularization.

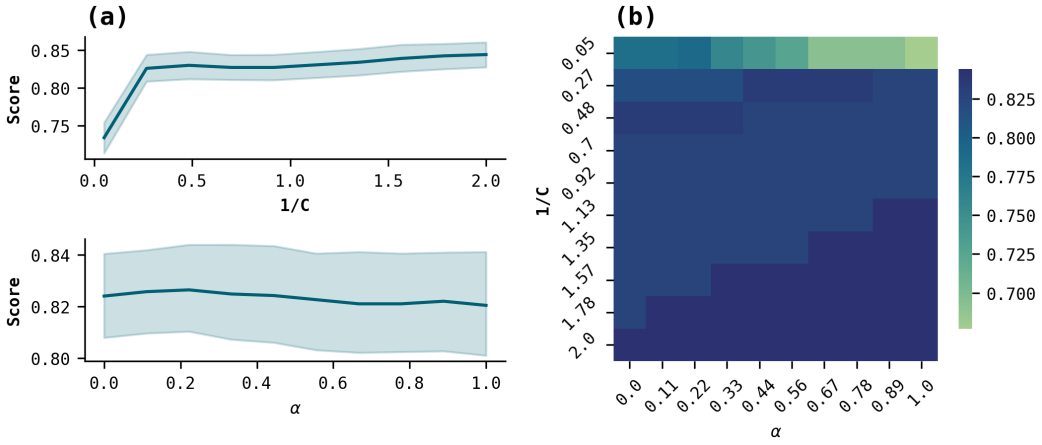

Figure 1.3: Grid search results. **(a)** X-axes depict two model settings sampled during HPO. Y-axes depict BAC scores with line depicting mean values and shaded ranges depicting BAC fluctuations across CV folds. **(b)** Hyperparameter combinations and their influence on BAC averaged across CV folds.

To gain further insights, we once again cross-validated the algorithm, this time using the attained hyperparameters. Firstly, as Fig. 1.2b indicates, the mean CV performance remained consistent. Secondly, (Fig. 1.2a) demonstrates the stability of the decision boundaries of models trained on different CV folds.

This practical illustration allowed us to explore some of the core ML principles described above. Namely, we've demonstrated: (1) the basics of hyperparameter optimization, (2) dealing with imbalanced datasets via stratified CV and an appropriately chosen metric, (3) the concepts of overfitting, bias, and variance, and (4) how regularization impacts the algorithm's performance.

1.2.4 Decision Trees

When the training data are not linearly separable, LR may suffer significant performance losses. In such cases, using non-linear ML models is justified. One of the simplest non-linear models is the Decision Tree (DT). DTs were popularized by Leo Breiman, who introduced an efficient algorithm to train them for both classification and regression tasks [23].

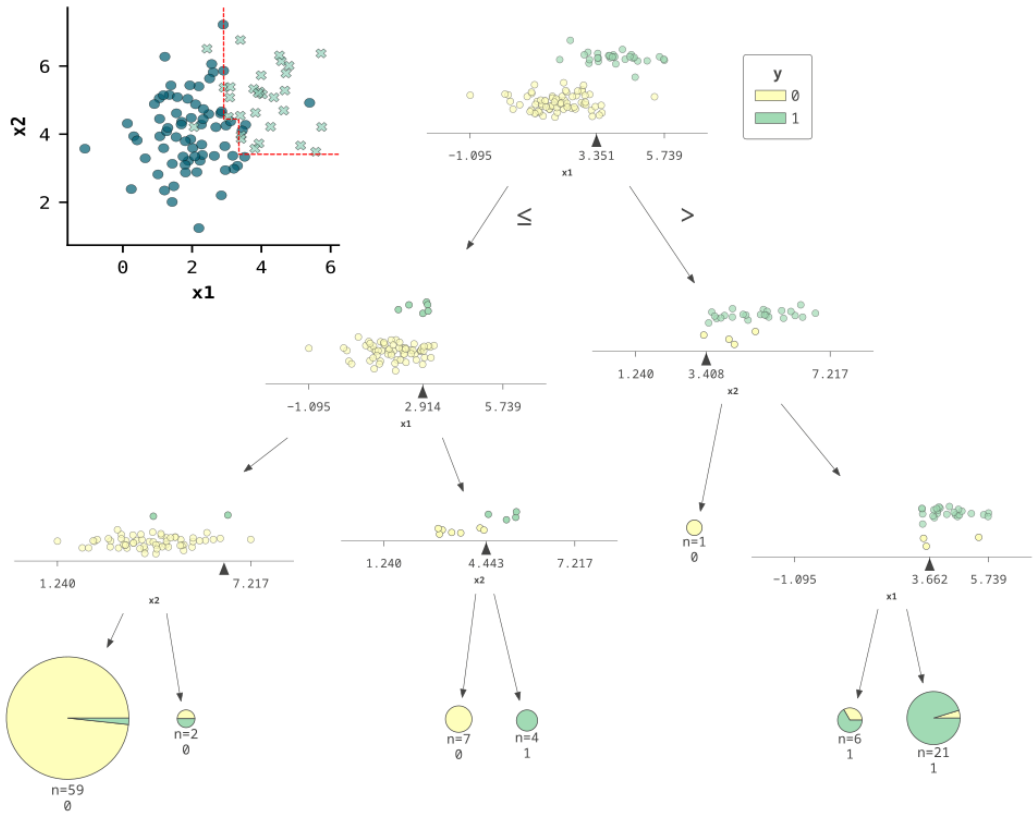

Figure 1.4: An example of a decision tree trained to partition the artificial data introduced in the main text.

A DT is a hierarchical tree model composed of internal and external nodes organized into "levels" (Fig. 1.4). Internal nodes are represented by a pair (f, t), where f is an input feature and t is a learned threshold. If $f > t$, a data instance is directed to one child node (typically the right); otherwise, it goes to the other child node. In this manner, data instances are guided by internal nodes towards the leaf nodes, which contain continuous values (for regression problems) or classes (for classification problems) that constitute the model's predictions.

To train a Decision Tree model, one typically employs a top-down, recursive, divide-and-conquer approach. The most commonly used algorithms for this purpose are ID3 (Iterative Dichotomiser 3) [24], C4.5 (the successor of ID3) [25], and CART (Classification and Regression Trees) [23].

The algorithm starts by selecting the feature and threshold that minimize the chosen criterion for the entire dataset. For classification tasks, this criterion is often the Gini impurity or the information gain. The Gini impurity for a node t is defined as:

$$\text{Gini}(t) = 1 - \sum_{i=1}^{C} p_i^2 \tag{1.8}$$

where p_i is the proportion of samples belonging to class i at node t, and C is the number of classes. Information gain based on Gini impurity is then calculated as:

$$\text{Gain}(t, f) = \text{Gini}(t) - \left(\frac{n_{\text{left}}}{n} \text{Gini}(t_{\text{left}}) + \frac{n_{\text{right}}}{n} \text{Gini}(t_{\text{right}}) \right) \tag{1.9}$$

where n is the total number of samples at node t, n_{left} and n_{right} are the number of samples in the left and right child nodes, respectively, created by splitting node t based on feature f.

This creates the root node of the tree, which splits the dataset into two subsets. The algorithm then recursively applies the same procedure to each subset, creating new internal nodes and partitions until one of the stopping conditions is met. These conditions could be a maximum tree depth, a minimum number of samples per leaf, or an acceptable level of impurity or error.

Once the tree is built, it can be pruned to remove branches that do not provide sufficient predictive power. This helps in reducing the complexity of the final model and mitigates the risk of overfitting.

Combining learned thresholds leads to non-linear decision boundaries. To illustrate this, we trained a single decision tree on the artificial dataset introduced above, restricting it to the maximum depth of three levels. Fig. 1.4 depicts this tree and its decision boundaries. As one can observe, the DT non-linearity allows it to better partition the data compared to LR.

1.2.5 Ensemble Algorithms

The motivation behind ensemble methods is rooted in the idea that while a single model may have limitations in terms of bias, variance, or generalization, a combination of models trained on the same objective can compensate for these weaknesses. This disctinction is made explicit by the commonly adopted terminology, where single models are typically denoted as **weak** learners, while their combination is referred to as **strong** learner [26].

The concept of ensemble learning was popularized by several pivotal works, including Leo Breiman's 1996 paper on "Bagging Predictors," which introduced the technique of bootstrap aggregating, or bagging, to improve classification and regression models [27]. The technique involves generating multiple bootstrap samples from the original training dataset and training a separate model on each sample. These models are typically identical in structure but differ in the data they are trained on. The final prediction is obtained by averaging the predictions (in the case of regression) or by taking a majority vote (in the case of classification) from all the individual models. Bagging is particularly effective for algorithms that are sensitive to the specific data they are trained on, as it provides a way to "average out" the noise and errors, leading to a more robust and stable model.

While bagging aims to reduce model variance by creating multiple bootstrap samples and averaging the predictions, AdaBoost, introduced by by Yoav Freund and Robert Schapire in 1997 [28], takes a different approach by focusing on instances that are hard to classify. In essence, AdaBoost adapts by giving more weight to training instances that are misclassified by previous models in the ensemble. Each subsequent model is then trained to correct the mistakes of its predecessors, thereby reducing the overall bias of the ensemble. This adaptive nature allows AdaBoost to create a strong learner from a series of weak learners, often achieving better performance than bagging, especially when the base models are simple and prone to underfitting.

Random Forests (RFs) comprise ensembles of decision trees. Introduced by Leo Breiman in 2001 [29] RF can be viewed as an extension of his earlier work on bagging. While bagging uses bootstrap samples to train each model independently, Random Forests add an additional layer of randomness by also selecting a random subset of features for each decision tree. This "decorrelates"

the trees, making the ensemble less prone to overfitting and often improving generalization. The method has become one of the most popular machine learning algorithms for both classification and regression tasks.

Gradient Boosting is another ensemble method that builds upon the principles introduced by both AdaBoost and Random Forests [30]. Like AdaBoost, it trains models in a sequential manner, focusing on instances that are difficult to predict. However, it often employs decision trees as base learners, similar to Random Forests, but optimizes them in a greedy manner. Each new model is fitted to the residuals of the combined ensemble of existing models, allowing the algorithm to adapt quickly and fit complex functions. Among the various implementations of gradient boosting, XGBoost stands out for its efficiency and effectiveness [31]. We'll provide a detailed explanation of this algorithm later on.

The versatility of Random Forest and gradient-boosted trees is evident in their wide range of applications. As we previously explored the role of Bayesian Optimization in hyperparameter tuning, it's worth noting that these ensemble methods often serve as surrogate probability models $P(\rho|\lambda)$, particularly useful for navigating large, high-dimensional hyperparameter spaces [32]. In the realm of anomaly detection, Random Forest algorithms, including specialized variants like "Isolation Forest," excel at identifying outliers based on their unique decision paths across the ensemble [33, 34]. The co-occurrence of features within the trees also allows Random Forest to be adapted for clustering tasks, generating a dissimilarity matrix that can be used for further analysis [35, 36]. Additionally, the algorithm's robustness makes it an effective tool for imputing missing values [37].

1.3 Interpretable ML

Machine learning approach has traditionally focused on predictive power. However, gradual increase of ML algorithms' complexity and their usage in various sensitive spheres like healthcare [38], necessitates gaining more insights into their decision-making process. Thus, Interpretable Machine Learning (IML[1]) has emerged and gained popularity in recent years [39–41].

IML focuses on conjuring meaningful explanations regarding an ML model's

[1]Some researches prefer the term "Explainable AI," abbreviated as XAI.

functioning. In this context, it's worth distinguishing global vs. local explanations. The former attempt to explain a model as a whole, while the latter focus on interpreting predictions for specific data instances. IML techniques differ in how they are applied to a model. Some of them are model-specific, while others are model-agnostic. Furthermore, as we'll elaborate below, some models have built-in features making them interpretable, while others (black-box models) are explained post-hoc [42].

1.3.1 Intrinsically Interpretable Models

Some ML models are interpretable by design. For instance, above we've introduced a Logistic Regression model and trained it on an artificial dataset. In LR, the log-odds of belonging to class 1 is modeled as a linear function of the features:

$$\log\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{1.10}$$

Here, p is the probability of belonging to class 1.

The LR model trained above had parameters $[\theta_0, \theta_1, \theta_2] = [-8.38, 2.00, 0.39]$, where θ_0 is an intercept coefficient. The positive θ_1 indicates that higher values of x_1 increase the log-odds of belonging to class 1, while the smaller θ_2 suggests a lesser influence of x_2 on the outcome. The intercept term $\theta_0 = 8.38$ represents the log-odds of the positive class when θ_1 and θ_2 are both zero, implying that, in this case, the model is biased towards the negative class.

DT model introduced above is also interpretable. For small decision trees, one can draw explanations from a visual representation. For instance, by observing the root node in Fig. 1.4, one can deduce that the x_1 variable alone separated most of the instances; hence, it's more useful for the objective than x_2.

For larger trees, visual inspection can be cumbersome. In such cases, the usefulness of a feature can be quantified via various metrics, such as feature importance scores. These scores are often calculated based on the average reduction in impurity or error that a feature brings when used in a tree split. Formally, the feature importance $\mathrm{FI}(f)$ for a feature f is computed as:

$$\mathrm{FI}(f) = \frac{\sum_{t \in \text{Nodes using } f} \text{Impurity Reduction}(t)}{\text{Total number of nodes}} \tag{1.11}$$

Here, Impurity Reduction(t) is the reduction in impurity or error at node t due to the split on feature f. This provides a quantitative measure of how much each feature contributes to the predictive power of the tree.

DTs can aid in local (instance-level) interpretations as well. By tracing the path an instance takes from the root to a leaf, one can understand the sequence of decisions made by the model for that particular instance that can be translated into a set of "if-then" rules. This set of rules provides a transparent way to understand the exact conditions under which a particular prediction is made.

We'll briefly mention some other interpretable models: (1) k-NN, (2) Naive Bayes, and (3) LASSO. The k-NN algorithm is a type of instance-based learning that classifies a new instance based on the majority class of its k nearest neighbors in the feature space. By examining the neighbors, one can understand why a particular instance was classified in a certain way.

Naive Bayes classifiers are probabilistic models based on Bayes' theorem, with the "naive" assumption that features are conditionally independent given the class label. The model is interpretable because it provides probabilities for each class, and one can look at the conditional probabilities to understand the contribution of each feature to these class probabilities.

LASSO (Least Absolute Shrinkage and Selection Operator) is a linear model trained with an l_1 penalty term on the coefficients, which has the effect of setting some coefficients to zero, effectively performing feature selection. This makes the model simpler and more interpretable, as only a subset of features contributes to the decision-making process.

1.3.2 Intepreting Ensemble Models

All the models introduced above have a small number of trainable parameters connected to their output in a straightforward manner. As the number of parameters and their interactions increase, a model essentially becomes a "black-box," requiring special techniques to interpret it.

Ensemble models illustrate well this interpretability-complexity trade-off [43]. While a single decision tree is interpretable, their large ensembles are not. Still, some of the techniques employed for a single tree may be generalized towards the ensemble as a whole. Thus, rule extraction procedure mentioned

above can be further improved by pruning techniques [44] and adapted for tree ensembles [45, 46]. However, a more intriguing line of work that generalizes beyond tree ensembles concerns feature importances.

Above, we've introduced a notion of feature importance. For extending it to a collection of trees, one can average the feature importance scores:

$$\mathrm{FI}_{\mathrm{ensemble}}(f) = \frac{1}{N}\sum_{i=1}^{N}\mathrm{FI}_{\mathrm{tree}_i}(f) \tag{1.12}$$

where N is the total number of trees in the ensemble [47]. This Mean Decrease in Impurity (MDI) gives a more comprehensive view of how each feature contributes to the predictive power of the ensemble model [48].

The Mean Decrease Accuracy (MDA) is another feature importance measure originally introduced for RF [29]. One of the advantages of using MDA is the availability of Out-of-Bag (OOB) scores [49]. In Random Forests, each tree is trained on a different bootstrap sample, and the samples not included in the bootstrap (OOB samples) can be used to test the tree. This provides a computationally efficient way to compute the MDA. Specifically, one can permute a feature and recompute the OOB score without having to retrain the model or set aside a separate validation set. The MDA for a feature f is computed as follows:

$$\mathrm{MDA}(f) = \frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{Acc}^{(i)}_{\mathrm{original}} - \mathrm{Acc}^{(i)}_{\mathrm{permuted}}\right) \tag{1.13}$$

where N is the number of trees in the ensemble, and $\mathrm{Acc}^{(i)}_{\mathrm{original}}$ and $\mathrm{Acc}^{(i)}_{\mathrm{permuted}}$ are the OOB accuracies of the i^{th} tree before and after permuting the feature values, respectively.

RF models, however, were empirically shown to be biased towards correlated and numerical or high-cardinality categorical features, which hinders their interpretability via MDI and MDA in these cases [50, 51]. Such features introduce many possible split scenarios, which increases their chance of yielding higher impurity reduction than features taking fewer values [52]. This bias can be reduced by employing alternative tree construction algorithms (such as cforest [53]) or debiased MDI formulations like MDI-OOB [54]. On the other hand, other feature importance formulations like permutation importance are

more general (i.e., model-agnostic) and are thought to mitigate the MDI biases.

The term "permutation importance" can refer to two different approaches, both of which involve decoupling X from y through random permutation or scrambling. One of these approaches, known as "PIMP," aims to correct biases in the initially computed feature importances and is applicable to any model that provides a method for assessing these importances [55].

PIMP starts by scrambling y multiple times to compute a "null importance" for each feature. These null importances are then fit to a chosen probability distribution (e.g., Random Normal). Using this distribution, PIMP calculates the likelihood of observing a feature importance value equal to or greater than the one initially computed (without scrambling). This likelihood serves as a new, statistically corrected measure of feature importance, assessing whether initial importance values could have occured by chance given the distriution of the null importances.

The other permutation importance strategy is to scramble values of X instead of y. This repeated scrambling is followed by computing the $\rho_{\text{original}} - \frac{1}{\#\text{permutations}}\rho_{\text{permuted}}$ difference. Opinions diverge on the exact permutation protocol (conditional [56] vs. independent) and whether one should use the original model to compute ρ_{permuted} (permute-and-predict strategy) or refit the algorithm following the permutation. Recent research advocates permuting feature values conditionally on other features and refitting the model [57]. While potentially more accurate, this approach suffers from high computational demands and is therefore not universally applicable.

All the feature importance measures introduced above are global: they give no insight into why a certain prediction was made for a certain instance. Although some efforts exist to "localize" measures like MDI [58], they are not model-agnostic and less widely adopted. On the other hand, methods that are both local and model-agnostic exists and will be discussed below.

1.3.3 LIME

LIME stands for Local Interpretable Model-agnostic Explanations [59]. The central premise of LIME is that the decision boundaries of a complex, black-box model can be locally approximated by simpler, interpretable models such as logistic regression or a decision tree. To explain a specific instance x, LIME

generates a dataset for training a surrogate model by perturbing the feature values of x (e.g., injecting Gaussian noise) and evaluating the corresponding predictions from the black-box model. To ensure that the perturbations are relevant, LIME assigns weights to these perturbed samples based on their proximity to the original instance x.

More formally, for a black-box model f, LIME aims to find a surrogate model g within a class of models G (e.g., logistic regression models), such that the following objective is minimized:

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{1.14}$$

Here, $L(f, g, \pi_x)$ is a loss function that measures how well g approximates f in the neighborhood of x, defined by the proximity measure π_x. The term $\Omega(g)$ represents the complexity of the model g, with the aim to keep it as simple as possible for interpretability.

For instance, consider a text classification problem where the black-box model f predicts the sentiment of a given text. In this case, the proximity measure π_x could be the cosine similarity between the vector representations of the text samples. The loss function L could be the mean squared error (MSE) between the predictions of f and g for the perturbed samples, weighted by π_x.

$$L(f, g, \pi_x) = \sum_{i=1}^{N} \pi_x(x, x_i) \left(f(x_i) - g(x_i)\right)^2 \tag{1.15}$$

Here, x is the original instance, x_i are the perturbed samples, and N is the number of such samples. The term $\pi_x(x, x_i)$ weights the contribution of each perturbed sample based on its similarity to x, ensuring that the surrogate model g is a good local approximation of f.

In practice, LIME simplifies the optimization problem by limiting the feature space and using weighted least squares for linear models or greedy algorithms for decision trees. Regularization techniques and user-defined constraints (specified by $\Omega(g)$) further guide the search for an optimal g, making the problem more tractable.

Despite its advantages, such as the ability to exert full control over the input data, LIME has notable limitations. For instance, LIME employs an exponential smoothing kernel to define a local neighborhood around a data point,

which exact parameters often require manual tuning for different problems [60]. This kernel-based approach also restricts LIME's effectiveness in high-dimensional spaces, as the exponential kernel tends to smooth distances, making it challenging to differentiate between proximate and distant data points. Additionally, LIME suffers from stability issues due to the randomness in generating perturbations; different runs can yield inconsistent explanations [61]. These shortcomings have led to the development of various adaptations aimed at mitigating these issues [62–64].

1.3.4 SHAP

Above, we've introduced model-agnostic methods that enable either global (e.g., permutation importance) or local (LIME) explanations. The SHAP method, on the other hand, bridges the gap between local and global explanations and has solid theoretical foundations and desirable mathematical properties.

SHAP stands for "SHapley Additive exPlanations" and is inspired by Shapley values, a concept from cooperative game theory [65]. In this framework, the "game" is the prediction task, and the "players" are the individual features [60]. Let ϕ_j be the SHAP value for feature j, N be the set of all features, and $f(S) = E[f(x)|x \in S]$ be the expected value of the model's prediction conditioned on a given subset of features S.

$$\phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left(f(S \cup \{j\}) - f(S)\right) \tag{1.16}$$

Thus, the SHAP value ϕ_j for a feature j is calculated as a weighted average of the differences between the expected model's predictions when feature j is included versus when it is not, across all possible subsets S of features. This weighted average ensures that the contribution of each feature is fairly allocated, satisfying properties like local accuracy, missingness, and consistency.

The robustness of SHAP values is rooted in the axioms of Shapley values from cooperative game theory, further extended by Lundberg and Lee to suit machine learning applications [66]. The original Shapley axioms are:

- **Efficiency**: The sum of the SHAP values ϕ_j for all features j plus the base value (the prediction that would be made without any features) is

equal to the actual prediction $f(x)$.

$$f(x) = \phi_0 + \sum_{j=1}^{F} \phi_j$$

- **Symmetry**: If two features i and j contribute equally to all possible combinations of features, their SHAP values should be the same.

  If $f(S \cup \{i\}) - f(S) = f(S \cup \{j\}) - f(S)$ for all $S \subseteq N \backslash \{i, j\}$, then $\phi_i = \phi_j$

- **Additivity**: The SHAP value for a feature is additive across the contributions of all other features. This property is inherent in the formula for ϕ_j, as it is a weighted sum of the feature's contributions across all possible subsets S.

Lundberg and Lee introduced additional properties tailored for machine learning [66, 67]:

- **Local Accuracy**: This is equivalent to the Efficiency axiom but emphasizes that the sum of the SHAP values equals the difference between the model output and the expected value for the instance being explained.
- **Missingness**: Features that are already missing (i.e., have a zero value) are attributed no importance.
- **Consistency**: Changing a model so that a feature has a larger impact will never decrease the attribution assigned to that feature.

These properties collectively make SHAP values a robust and consistent method for explaining both individual predictions and the global behavior of a model.

However, the SHAP formulation in Eq. 1.16 presents two challenges: (1) the efficient estimation of $f(S)$, and (2) the exponential complexity involved in considering 2^F combinations for F features. As a result, SHAP values are often estimated rather than computed exactly in practice. Lundberg et al. proposed various heuristic approaches tailored for different types of models. One such approach is the `TreeSHAP` algorithm [67, 68], which estimates SHAP contributions specifically for tree-based models like Random Forests or Gradient Boosting Machines in polynomial time.

In addition to providing local explanations, SHAP values can also be used to derive global feature importances and interaction values. To obtain global feature importances, consider an input dataset X consisting of N instances and M features. After computing the SHAP values for all instances in X, we obtain an $N \times M$ matrix Ψ, where Ψ_{ij} represents the SHAP value of the j-th feature for the i-th instance. The global feature importances can then be calculated as:

$$F = \sum_{i=1}^{N} |\Psi_{ij}| \tag{1.17}$$

where F is an M-dimensional vector. Each element F_j quantifies the overall impact of the j-th feature on the model's output, irrespective of the direction (positive or negative) of its effect.

SHAP interaction values extend the concept of SHAP values to capture not just the individual contributions of features but also their interactions [68]. This provides a more nuanced understanding of the model's behavior by allowing for the separate consideration of interaction effects for individual model predictions.

The key term for understanding SHAP interaction values is $\nabla_{ij}(f, x, S)$, defined as:

$$\nabla_{ij}(f, x, S) = f_x(S \cup \{i, j\}) - f_x(S \cup \{i\}) - f_x(S \cup \{j\}) + f_x(S) \tag{1.18}$$

This term quantifies the change in the model's prediction when both features i and j are included in the subset S, compared to when they are considered individually.

One considers all possible subsets S and attributes weigths that accounts for the number of ways a particular subset S, yielding

$$\Phi_{i,j}(f, x) = \sum_{S \subseteq M \setminus \{i,j\}} \frac{|S|! \cdot (M - |S| - 2)!}{(M - 1)!} \nabla_{ij}(f, x, S) \tag{1.19}$$

The interaction value is symmetric, i.e., $\Phi_{i,j}(f, x) = \Phi_{j,i}(f, x)$. SHAP value for i-th feature ϕ_i can be decomposed into main (diagonal) and interaction (off-

diagonal) effects:

$$\phi_i(f, x) = \Phi_{i,i}(f, x) + \sum_{j \neq i} \Phi_{i,j}(f, x) \tag{1.20}$$

This ensures that the sum of all interaction values and individual contributions equals the model's output, providing a complete and fairly allocated explanation of the model's prediction.

1.4 Feature Selection

As previously discussed in this chapter, supervised learning aims to establish a relationship between input variables and a target variable by fine-tuning model parameters. However, not all input variables contribute equally to predictive performance. This is particularly true in bioinformatics, where datasets are often plagued by noise [69]. To address this, a preprocessing step is essential to filter out irrelevant or redundant variables – a process known as Feature Selection (FS) [70].

Deleting irrelevant or redundant input data may lead to the following outcomes:

- **Performance Enhancement**: Many machine learning algorithms are sensitive to noisy data. Eliminating irrelevant features can lead to improved model accuracy.
- **Enhanced Transparency and Interpretability**: FS can aid in a model's interpretation. A reduced set of variables simplifies reasoning about the model. Furthermore, the remaining features after filtering are inherently valuable for the objective, providing additional insights.
- **Model Simplification**:
  - **Overfitting**: Feature selection can act as an implicit form of regularization, reducing the model's susceptibility to overfitting. However, improper application of FS techniques can inadvertently lead to overfitting, especially in scenarios where FS serves as a form of model selection [71, 72].
  - **Reduced Computation**: A less complex model eases the computational load, making it more efficient for time-sensitive tasks like

hyperparameter tuning.

Due to these properties, FS techniques became an indespesible tool in many areas, including bioinformatics [73], genomics [74], chemoinformatics [75], and healthare [76], where they are adapted to fit field-specific requirements and learning objectives.

1.4.1 Feature Categories

Above, we've used terms such as "relevant" or "redundant." Here, we'll introduce their precise meaning following the definitions from Li et al. [77]. Let $P(C|S)$ denote the conditional probability of a class C given a set of features S that some classifier model outputs. With this, features can be categorized into:

- **Strongly Relevant**: A feature f_j is strongly relevant if

$$P(C|f_j, S_j) \neq P(C|S_j)$$

- **Weakly Relevant**: A feature f_j is weakly relevant if

$$P(C|f_j, S_j) = P(C|S_j)$$

and

$$\exists S'_j \subset S_j \text{such that} P(C|f_j, S'_j) \neq P(C|S'_j)$$

- **Irrelevant**: A feature f_j is irrelevant if

$$\forall S'_j \subset S_j, P(C|f_j, S'_j) = P(C|S'_j)$$

- **Redundant**: A feature f_j is redundant if it is weakly relevant and has a Markov blanket M_j within f. M_j is a Markov blanket for f_j if

$$P(F - M_j - \{f_j\}, C|f_j, M_j) = P(F - M_j - \{f_j\}, C|M_j)$$

To elaborate, a feature is *strongly relevant* if it alone has an impact on the prediction of the class C. A feature is *weakly relevant* if it doesn't affect the prediction when considered with all other features but does make a difference

with some subset of features. A feature is *irrelevant* if it doesn't help in predicting C better for any subset of features. Lastly, a feature is *redundant* if it is weakly relevant but there exists a set of features, known as the Markov blanket, that makes the feature unnecessary for predicting C.

A feature selection algorithm must discriminate between relevant and irrelevant features. Some contexts may also require removing redundant or weakly relevant features.

1.4.2 Feature Selection Strategies

From the learning perspective, FS follows a broad categorization according to the learning paradigms [78, 79]:

- **Supervised**: In supervised learning, the target values are known, and features are selected based on their relevance to these target values. Methods such as Recursive Feature Elimination (RFE) and LASSO are commonly used in this category.
- **Unsupervised**: In unsupervised learning, the target values are not known. Feature selection is often based on the structure and distribution of the data. Techniques like Principal Component Analysis (PCA) and clustering-based feature selection are examples in this category.
- **Semi-Supervised**: Semi-supervised feature selection methods like Laplacian Score [80] leverage both labeled and unlabeled data to perform feature selection.

Some machine learning models inherently perform feature selection during the training process; these models simply ignore features that are irrelevant to the objective [81]. Examples include interpretable models like decision trees and LASSO, as previously discussed. Such intrinsic feature selection techniques are commonly referred to as *embedded* methods. However, in many instances, an external search algorithm is necessary for feature selection. These external algorithms are generally categorized into *filter* and *wrapper* methods.

Filter methods operate independently of any learning algorithm and evaluate the relevance of each feature based on statistical properties of the data. The primary advantage of filter methods is their computational efficiency, as they do not necessitate model training. Examples of filter methods include Chi-Squared Test, Information Gain, and Correlation Coefficient methods. These

techniques often rank features based on a particular metric and allow for the selection of the top-ranked features for model training.

Wrapper methods, in contrast, utilize the trained model itself to evaluate different feature subsets. In supervised learning scenarios, the typical workflow involves retraining the model for each feature subset and assessing its performance. Formally, for each subset S within the feature space $\mathcal{F}$, wrapper methods aim to solve the following optimization problem:

$$\underset{S \subseteq \mathcal{F}}{\operatorname{argmax}} \ \rho(Y, f(S)) \tag{1.21}$$

Here, $f(S)$ represents the predictions made by a model trained using only the features in subset S. In this context, wrapper methods essentially serve as a form of model selection and are susceptible to overfitting if not applied cautiously [14, 72].

Due to the combinatorial nature of the problem, finding the global optimum would require exploring $2^{\mathcal{F}}$ combinations, which is computationally infeasible in most cases. This necessitates the use of heuristic approximations. Techniques akin to those used in hyperparameter optimization, such as evolutionary algorithms [82], can be employed for global searches across $\mathcal{F}$. While greedy strategies like recursive feature elimination (RFE) are less commonly used in practice, they still hold educational value. Moreover, certain adaptations of RFE can effectively filter out correlated features [83].

1.4.3 The Boruta Algorithm

Boruta (a good of forest in Polish folklore) is a supervised all-relevant feature selection wrapper method initially developed by Kursa and Rudnicki for Random Forests [84]. The key idea behind Boruta is that, for an ML model like RF, irrelevant features would be indistinguishable from noise in their importance values, while relevant features would yield consistently higher feature importances than noise variables.

Here, we'll describe a model-agnostic version of Boruta (Algorithm 1), suitable for any model, as long as it can be queried for feature importances. Another deviation from the original formulation is an imporoved correction for multiple statistical testing (see below).

Algorithm 1 Boruta Algorithm for Feature Selection

Input: Model f, Feature evaluation function g, data matrix X of N instances and M features, response variables Y, total number of steps K, percentile k, p-value α.
Output: Vector $\mathbf{r}$ of feature relevance
procedure BORUTA(f, g, X, Y, K, k, α)
    Initialize: $\mathbf{c} \leftarrow \mathbf{0}^M$, $\mathbf{r} \leftarrow \mathbf{0}^M$
    $i \leftarrow 1$
    while $i < K$ **and** $\exists r_j : r_j = 0$ **do**
        $M' \leftarrow \text{size}(X, 2)$ ▷ Current number of features in X
        Create shadow features X^s by permuting X
        Augment dataset $[X\, X^s]$
        Train f on $[X\, X^s], Y$
        Compute feature importances $\mathbf{I} = g(f([X\, X^s]))$
        Compute $Q_k = \text{percentile}(\{I_l : l > M'\})$
        $\mathbf{h} \leftarrow \mathbf{I}_{1:M'} > Q_k$ ▷ Vectorized Hit/Miss
        $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{h}$
        $\mathbf{r} \leftarrow \mathit{UPDATE}(\mathbf{r}, \mathbf{c}, i, \alpha)$
        Update X to include only features where $r_j = 0$
        $i \leftarrow i + 1$
    end while
    return r
end procedure
procedure UPDATE($\mathbf{r}, \mathbf{c}, i, \alpha$)
    Initialize: $\mathbf{a}, \mathbf{b} \leftarrow \mathbf{0}^M$
    $I \leftarrow \{j : r_j = 0\}$ ▷ Indices where $\mathbf{r} = 0$
    $\mathbf{t} \leftarrow c_I$ ▷ Subset of $\mathbf{c}$ for tentative features
    function TESTHITS($\mathbf{t}, I$, alternative)
        $\mathbf{p} \leftarrow \mathbf{0}^M$
        for j in I **do**
            $p_j \leftarrow \text{BinomTest}(t_j, i, \text{alternative})$
        end for
        $\mathbf{d} \leftarrow \text{FDR}(\mathbf{p}_I)$ ▷ False Discovery Rate correction
        $\mathbf{e} \leftarrow \mathbf{p}_I \leq \frac{\alpha}{i}$ ▷ Bonferroni correction
        return $\mathbf{d} \odot \mathbf{e}$
    end function
    $\mathbf{a}_I \leftarrow \text{TestHits}(\mathbf{t}, I, \text{"greater"})$
    $\mathbf{b}_I \leftarrow \text{TestHits}(\mathbf{t}, I, \text{"less"})$
    $r_j \leftarrow \begin{cases} 1 & \text{if } a_j = 1, j \in I \\ -1 & \text{if } b_j = 1, j \in I \\ 0 & \text{otherwise} \end{cases}$
    return r
end procedure

The Boruta algorithm aims to identify all relevant features in a given dataset by comparing the importance of each feature against random noise. The algorithm takes as input a machine learning model f, a feature evaluation function g, a data matrix X with M features, response variables Y, a total number of iterations K, a percentile k, and a p-value α. Here, k is the percentile used for importance thresholding, and α is the significance level for statistical tests (explained below). The output is a vector $\mathbf{r}$, where each element corresponds to the relevance of a feature: -1 for irrelevant, 0 for tentative, and 1 for relevant.

The algorithm uses dynamic data structures that are updated throughout its iterations. Specifically, $\mathbf{c}$ is a vector that keeps track of the "hit counts" for each feature, and $\mathbf{r}$ is a vector that stores the final relevance decision for each feature. The dataset X itself is also dynamic; it is pruned to only include tentative features as the algorithm progresses.

The algorithm iterates through a series of steps that involve creating shadow features, training the model, calculating feature importances, and performing statistical tests to update $\mathbf{r}$. This iterative process continues until either a maximum number of iterations K is reached or all features have been classified as either relevant or irrelevant.

Namely, each iteration of the algorithm comprises the following steps:

- **Creating an Augmented Dataset:** The algorithm starts by creating "shadow" features, which are randomized versions of the original features in X. These shadow features are then concatenated to the original dataset, forming an augmented dataset $[X\, X^s]$.
- **Fitting the Model:** The machine learning model f is trained on this augmented dataset $[X\, X^s]$ and the target variable Y.
- **Calculating Feature Importance Values:** Once the model is trained, feature importances are calculated using the function g. This results in a vector $\mathbf{I}$, where each element represents the importance of a corresponding feature in the augmented dataset.
- **Hit/Miss Determination:** A percentile Q_k is computed based on the importances of the shadow features. Each real feature is then compared to this threshold. If its importance is greater, it's considered a "hit," and the corresponding element in the hit count vector $\mathbf{c}$ is incremented.
- **Statistical Tests:** The *UPDATE* procedure is called, which performs

statistical tests on the hit counts. These tests are corrected for multiple comparisons using both False Discovery Rate (FDR) and Bonferroni methods. Based on these tests, the relevance vector $\mathbf{r}$ is updated.

- **The Dataset Update:** After each iteration, the dataset X is pruned to only include features that are still considered "tentative," i.e., those for which $r_j = 0$.

The *UPDATE* procedure serves as the statistical backbone of the Boruta algorithm, determining the relevance or irrelevance of each feature based on accumulated "hit" counts. Its core part is the binomial test, evaluating whether the number of hits for a feature could have occurred by random chance, given i number of trials.

However, the p-values produced by the binomial test are susceptible to high false positive rates due to two distinct types of multiple comparisons: (1) testing multiple features simultaneously in each iteration, and (2) testing the same features across multiple iterations. To control for these, the algorithm employs two types of corrections:

1. **False Discovery Rate (FDR)**: This correction is applied to account for the multiple features being tested in each iteration. FDR is less conservative than other methods like Bonferroni and is often used in high-dimensional settings.
2. **Bonferroni Correction**: This is used to adjust for the repeated testing of the same features across multiple iterations. It is a more conservative method designed to control the family-wise error rate, thereby reducing the chance of any false positives.

By combining these two corrections, the algorithm aims to provide a more accurate and conservative assessment of feature relevance, effectively controlling for both types of multiple comparisons.

The presented algorithm offers a highly flexible and adaptable approach to feature selection, making it a versatile tool for various machine learning applications. By abstracting the evaluation of feature importance, it can be applied to any supervised machine learning model. This flexibility extends to the choice of feature importance metrics, allowing for the incorporation of state-of-the-art interpretability methods like SHAP values.

Despite being categorized as a *wrapper* method, Boruta diverges from traditional wrapper approaches by focusing on feature importance values rather than optimizing for the best-performing subset of features. On the one hand, it mitigates the risk of overfitting, a common pitfall associated with model selection methods. On the other hand, as the algorithm progresses and the most relevant features are eliminated from the system, the model becomes less accurate. Assuming the SHAP importance is used, a "relevant" feature may have high contribution but to the wrong output.

In terms of computational efficiency, one might initially perceive Boruta as resource-intensive due to its iterative nature and the requirement to fit the model multiple times on an augmented dataset. However, in practice, the algorithm is quite efficient. Most irrelevant features are typically eliminated in the early iterations, making it particularly well-suited for high-dimensional and noisy data, a common scenario in bioinformatics.

Another noteworthy aspect of Boruta is its "all-relevant" feature selection approach, implying that the algorithm may retain features that are correlated or redundant. Depending on the specific application, this can be either an advantage or a drawback.

Although Boruta doesn't rank the resulting features, the initial importance evaluator g can be applied post-selection to the features deemed relevant by Boruta.

1.5 Concluding Remarks

This overview serves as a foundational guide for the concepts and methodological choices explored in subsequent chapters regarding ML. Here, we explored the general principles behind training, evaluating, and interpreting supervised models. Discussing the neighborhood of closely related techniques with their strengths and weaknesses meant to clarify our methodological standpoint. We paid special attention to methods like SHAP and the Boruta algorithm and laid out their theoretical basis. The field of interpretable machine learning is blooming with new approaches and theoretical studies behind the existing ones. Thus, the generalized interpretable ML pipeline presented in subsequent chapters will likely age fast and require improvements. Yet, its core ingredients are backed up by solid theoretical foundations and many successful applica-

tions in bioinformatics and beyond.

References

1. Samuel, A. L. Some Studies in Machine Learning Using the Game of Checkers. *IBM J. Res. & Dev.* **3,** 210–229. ISSN: 0018-8646 (July 1959).
2. Rosenblatt, F. Perceptron Simulation Experiments. *Proceedings of the IRE* (Mar. 1960).
3. Pugliese, R., Regondi, S. & Marini, R. Machine learning-based approach: global trends, research directions, and regulatory standpoints. *Data Sci. & Manag.* **4,** 19–29. ISSN: 2666-7649 (Dec. 2021).
4. Witten, I. H., Frank, E. & Hall, M. A. *Chapter 1 - What's It All About?* Third Edition, 3–38. ISBN: 978-0-12-374856-0 (Morgan Kaufmann, Boston, 2011).
5. Breiman, L. *Bias, variance, and arcing classifiers* tech. rep. 460 (Statistics Department, University of California at Berkeley, 1996).
6. Montesinos López, O. A., Montesinos López, A. & Crossa, J. *Overfitting, Model Tuning, and Evaluation of Prediction Performance* 109–139 (Springer International Publishing, Cham, 2022).
7. Rocks, J. W. & Mehta, P. Memorizing without overfitting: Bias, variance, and interpolation in overparameterized models. *Phys. Rev. Res.* **4,** 013201 (1 Mar. 2022).
8. Neal, B. On the Bias-Variance Tradeoff: Textbooks Need an Update. *arXiv.* eprint: 1912.08286 (Dec. 2019).
9. Stone, M. Cross-Validatory Choice and Assessment of Statistical Predictions. *J. Roy. Stat. Soc. Ser. B* **36,** 111–133. ISSN: 0035-9246 (Jan. 1974).
10. Kohavi, R. & John, G. H. in *Machine Learning Proceedings 1995* (eds Prieditis, A. & Russell, S.) 304–312 (Morgan Kaufmann, San Francisco (CA), 1995). ISBN: 978-1-55860-377-6.
11. Soper, D. S. Greed Is Good: Rapid Hyperparameter Optimization and Model Selection Using Greedy k-Fold Cross Validation. *Electronics* **10.** ISSN: 2079-9292 (2021).
12. Kawamoto, T. & Kabashima, Y. Cross-validation estimate of the number of clusters in a network. *Scient. Rep.* **7,** 3327. eprint: 1605.07915 (June 2017).
13. Fu, W. & Perry, P. O. Estimating the Number of Clusters Using Cross-Validation. *J. Comp. & Graphic. Stat.* **29,** 162–173. ISSN: 1061-8600 (Jan. 2020).

14. Cawley, G. C. & Talbot, N. L. C. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *J. Mach. Learn. Res.* **11,** 2079–2107 (2010).

15. Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)* ISBN: 0387310738 (Springer-Verlag, Berlin, Heidelberg, 2006).

16. Tabe-Bordbar, S., Emad, A., Zhao, S. D. & Sinha, S. A closer look at cross-validation for assessing the accuracy of gene regulatory networks and models. *Scient. Rep.* **8,** 6620 (Apr. 2018).

17. Kramer, C. & Gedeck, P. Leave-Cluster-Out Cross-Validation Is Appropriate for Scoring Functions Derived from Diverse Protein Data Sets. *J. Chem. Inform. Model.* **50,** 1961–1969. ISSN: 1549-9596 (Nov. 2010).

18. Yang, L. & Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **415,** 295–316. ISSN: 0925-2312 (2020).

19. Probst, P., Bischl, B. & Boulesteix, A.-L. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *arXiv.* eprint: 1802.09596 (Feb. 2018).

20. Yu, T. & Zhu, H. Hyper-Parameter Optimization: A Review of Algorithms and Applications. *arXiv.* eprint: 2003.05689 (Mar. 2020).

21. Feurer, M. & Hutter, F. Automated Machine Learning. *The Springer Series on Challenges in Machine Learning,* 3–33. ISSN: 2520-131X (Jan. 2019).

22. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. *Optuna: a next-generation hyperparameter optimization framework* in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019).

23. Gordon, A. D., Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. Classification and Regression Trees. *Biometrics* **40,** 874. ISSN: 0006-341X (Sept. 1984).

24. Quinlan, J. R. Induction of Decision Trees. *Mach. Learn.* **1,** 81–106 (1986).

25. Quinlan, J. R. *C4.5: Programs for Machine Learning* (Morgan Kaufmann, San Mateo, California, 1993).

26. Zhou, Z.-H. in *Ensemble Methods: Foundations and algorithms* 15–17 (Chapman & Hall/CRC, 2012).

27. Breiman, L. Bagging predictors. *Mach. Learn.* **24,** 123–140. ISSN: 0885-6125 (Aug. 1996).

28. Freund, Y. & Schapire, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Computer & Syst. Sci.* **55,** 119–139. ISSN: 0022-0000 (Aug. 1997).

29. Breiman, L. Random forests. *Mach. Learn.* **45,** 5–32. ISSN: 0885-6125 (2001).

30. Friedman, J. H. Greedy function approximation: a gradient boosting machine. *The Annals Stat.* **29.** ISSN: 0090-5364 (2001).

31. Chen, T. & Guestrin, C. XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 785–794 (2016).

32. Hoof, J. v. & Vanschoren, J. Hyperboost: Hyperparameter Optimization by Gradient Boosting surrogate models. *arXiv.* eprint: 2101.02289 (Jan. 2021).

33. Kopp, M., Pevný, T. & Holeňa, M. Anomaly explanation with random forests. *Exp. Syst. Appl.* **149,** 113187. ISSN: 0957-4174 (July 2020).

34. Liu, F. T., Ting, K. M. & Zhou, Z.-H. Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining,* 413–422 (Dec. 2008).

35. Shi, T. & Horvath, S. Unsupervised Learning With Random Forest Predictors. *J. Comp. & Graphic. Stat.* **15,** 118–138. ISSN: 1061-8600 (Mar. 2006).

36. Bicego, M. DisRFC: a dissimilarity-based Random Forest Clustering approach. *Patt. Recogn.* **133,** 109036. ISSN: 0031-3203 (Jan. 2023).

37. Tang, F. & Ishwaran, H. Random Forest Missing Data Algorithms. *Stat. Anal. & Data Min.* **10,** 363–377. ISSN: 1932-1864 (June 2017).

38. Rubinger, L., Gazendam, A., Ekhtiari, S. & Bhandari, M. Machine learning and artificial intelligence in research and healthcare. *Injury* **54,** S69–S73. ISSN: 0020-1383 (May 2023).

39. Arrieta, A. B. *et al.* Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Infor. Fusion* **58,** 82–115. ISSN: 1566-2535 (2020).

40. Molnar, C., Casalicchio, G. & Bischl, B. Interpretable Machine Learning – A Brief History, State-of-the-Art and Challenges. *arXiv.* eprint: 2010.09337 (Oct. 2020).

41. Linardatos, P., Papastefanopoulos, V. & Kotsiantis, S. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy* **23,** 18 (Dec. 2020).

42. Kamath, U. & Liu, J. *Introduction to Interpretability and Explainability* 1st, 1–18 (Springer Cham, 2021).

43. Angelov, P. P., Soares, E. A., Jiang, R., Arnold, N. I. & Atkinson, P. M. Explainable artificial intelligence: an analytical review. *WIREs Data Min. & Knowl. Disc.* **11.** ISSN: 1942-4787 (July 2021).

44. Pal, N. R. & Chakraborty, S. Fuzzy Rule Extraction From ID3-Type Decision Trees for Real Data. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics* **31,** 745–754. ISSN: 1083-4419 (Oct. 2001).

45. Bénard, C., Biau, G., Veiga, S. d. & Scornet, E. Interpretable Random Forests via Rule Extraction. *arXiv.* eprint: 2004.14841 (Apr. 2020).

46. Obregon, J. & Jung, J.-Y. RuleCOSI+: Rule extraction for interpreting classification tree ensembles. *Inform. Fusion* **89,** 355–381. ISSN: 1566-2535 (Jan. 2023).

47. Han, H., Guo, X. & Yu, H. Variable Selection Using Mean Decrease Accuracy and Mean Decrease Gini Based on Random Forest. *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS),* 219–224 (Aug. 2016).

48. Scornet, E. Trees, forests, and impurity-based variable importance in regression. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques* **59.** ISSN: 0246-0203 (Feb. 2023).

49. Archer, K. J. & Kimes, R. V. Empirical characterization of random forest variable importance measures. *Comp. Stat. & Data An.* **52,** 2249–2260. ISSN: 0167-9473 (Jan. 2008).

50. Strobl, C., Boulesteix, A.-L., Zeileis, A. & Hothorn, T. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinf.* **8,** 25 (Jan. 2007).

51. Nembrini, S., König, I. R. & Wright, M. N. The revival of the Gini importance? *Bioinformatics* **34,** 3711–3718. ISSN: 1367-4803 (Nov. 2018).

52. Wright, M. N., Dankowski, T. & Ziegler, A. Unbiased split variable selection for random survival forests using maximally selected rank statistics. *Stat. Medic.* **36,** 1272–1284. ISSN: 0277-6715. eprint: 1605.03391 (Apr. 2017).

53. Hothorn, T., Hornik, K. & Zeileis, A. Unbiased Recursive Partitioning: A Conditional Inference Framework. *J. Comp. & Graphic. Stat.* **15,** 651–674. ISSN: 1061-8600 (Sept. 2006).

54. Li, X., Wang, Y., Basu, S., Kumbier, K. & Yu, B. A Debiased MDI Feature Importance Measure for Random Forests. *arXiv.* eprint: 1906.10845 (June 2019).

55. Altmann, A., Toloşi, L., Sander, O. & Lengauer, T. Permutation importance: a corrected feature importance measure. *Bioinformatics* **26,** 1340–1347. ISSN: 1367-4803 (May 2010).

56. Fisher, A., Rudin, C. & Dominici, F. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *J. Mach. Learn. Res.* **20.** ISSN: 1532-4435 (Jan. 2019).

57. Hooker, G., Mentch, L. & Zhou, S. Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance. *Stat. & Comp.* **31,** 82. ISSN: 0960-3174 (Nov. 2021).

58. Sutera, A., Louppe, G., Huynh-Thu, V. A., Wehenkel, L. & Geurts, P. From global to local MDI variable importances for random forests and when they are Shapley values. *arXiv.* eprint: 2111.02218 (Nov. 2021).

59. Krishnapuram, B. *et al.* "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 1135–1144 (Aug. 2016).

60. Molnar, C. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable* 2nd ed. https://christophm.github.io/interpretable-ml-book (2022).

61. Visani, G., Bagli, E., Chesani, F., Poluzzi, A. & Capuzzo, D. Statistical stability indices for LIME: Obtaining reliable explanations for machine learning models. *J. Oper. Res. Soc.* **73,** 91–101. ISSN: 0160-5682. eprint: 2001.11757 (Jan. 2022).

62. Zafar, M. R. & Khan, N. M. DLIME: A Deterministic Local Interpretable Model-Agnostic Explanations Approach for Computer-Aided Diagnosis Systems. *arXiv.* eprint: 1906.10263 (June 2019).

63. Zhou, Z., Hooker, G. & Wang, F. S-LIME: Stabilized-LIME for Model Explanation. *arXiv.* eprint: 2106.07875 (June 2021).

64. Li, X. *et al.* G -LIME: Statistical learning for local interpretations of deep neural networks using global priors. *Artif. Intel.* **314,** 103823. ISSN: 0004-3702 (Jan. 2023).

65. Shapley, L. S. in *Contributions to the Theory of Games II* (eds Kuhn, H. W. & Tucker, A. W.) 307–317 (Princeton University Press, Princeton, 1953).

66. Lundberg, S. & Lee, S.-I. A Unified Approach to Interpreting Model Predictions. *arXiv.* eprint: 1705.07874 (May 2017).

67. Lundberg, S. M., Erion, G. G. & Lee, S.-I. Consistent individualized feature attribution for tree ensembles. *arXiv.* eprint: 1802.03888 (2018).

68. Lundberg, S. M. *et al.* From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.* **2,** 56–67 (Jan. 2020).

69. Liu, S. *et al.* Feature selection of gene expression data for Cancer classification using double RBF-kernels. *BMC Bioinf.* **19,** 396 (Oct. 2018).

70. Liu, H. & Motoda, H. *Feature Selection for Knowledge Discovery and Data Mining* 1st. ISBN: 978-1-4613-7604-0 (Springer Science+Business Media, 1998).

71. Kohavi, R. & Sommerfield, D. *Feature Subset Selection Using the Wrapper Method: Overfitting and Dynamic Search Space Topology* in *Proceedings of the First International Conference on Knowledge Discovery and Data Mining* (AAAI Press, Montréal, Québec, Canada, 1995), 192–197.

72. Smialowski, P., Frishman, D. & Kramer, S. Pitfalls of supervised feature selection. *Bioinformatics* **26,** 440–443. ISSN: 1367-4803 (Feb. 2010).

73. Saeys, Y., Inza, I. & Larrañaga, P. A review of feature selection techniques in bioinformatics. *Bioinformatics* **23,** 2507–2517. ISSN: 1367-4803 (Oct. 2007).

74. Tadist, K., Najah, S., Nikolov, N. S., Mrabti, F. & Zahi, A. Feature selection methods and genomic big data: a systematic review. *J. Big Data* **6,** 79 (Aug. 2019).

75. Goodarzi, M., Dejaegher, B. & Heyden, Y. V. Feature Selection Methods in QSAR Studies. *J. AOAC Intern.* **95,** 636–651. ISSN: 1060-3271 (June 2012).

76. Pudjihartono, N., Fadason, T., Kempa-Liehr, A. W. & O'Sullivan, J. M. A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction. *Front. Bioinf.* **2,** 927312 (June 2022).

77. Li, Y., Li, T. & Liu, H. Recent advances in feature selection and its applications. *Knowl. & Inform. Syst.* **53,** 551–577. ISSN: 0219-1377 (2017).

78. Li, J. *et al.* Feature Selection: A Data Perspective. *ACM Comp. Surv.* **50,** 1–45. ISSN: 0360-0300. eprint: 1601.07996 (2017).

79. Solorio-Fernández, S., Carrasco-Ochoa, J. A. & Martínez-Trinidad, J. F. A review of unsupervised feature selection methods. *Artif. Intel. Rev.* **53,** 907–948. ISSN: 0269-2821 (2020).

80. He, X., Cai, D. & Niyogi, P. *Laplacian Score for Feature Selection* in *Advances in Neural Information Processing Systems* (eds Weiss, Y., Schölkopf, B. & Platt, J.) **18** (MIT Press, 2005).

81. Kuhn, M. & Johnson, K. *Feature Selection Overview* 227–240. ISBN: 978-1-138-07922-9 (Chapman & Hall, July 2019).

82. Goldberg, D. E. & Holland, J. H. Genetic Algorithms and Machine Learning. *Mach. Learn.* **3,** 95–99. ISSN: 0885-6125 (1988).

83. Darst, B. F., Malecki, K. C. & Engelman, C. D. Using recursive feature elimination in random forest to account for correlated variables in high dimensional data. *BMC Genet.* **19,** 65 (Sept. 2018).

84. Kursa, M. B. & Rudnicki, W. R. Feature selection with the Boruta package. *J. Stat. Soft.* **36,** 1–13 (2010).

Chapter 2

Introduction to Protein Kinases

Kinases are enzymes that catalyze the phosphorylation reaction, transferring a γ-phosphate group from a nucleotide triphosphate (NTP) to a hydroxyl group of a specific substrate (Fig. 2.1). Although ATP is the most commonly used phosphate donor, other NTPs can also serve this role. Enzymatic classification of kinases is based on their substrates, such as Protein Kinases (PKs), Lipid Kinases, and Carbohydrate Kinases, each with distinct roles and regulatory mechanisms.

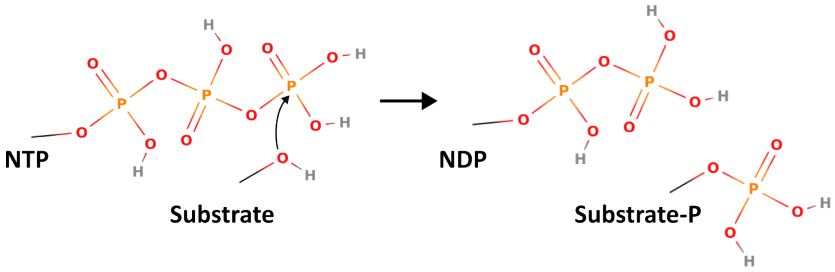

Figure 2.1: The phosphorylation reaction. A phosphoryl moiety is transferred from an NTP, typically ATP, to a substrate.

Protein phosphorylation, primarily executed by PKs, is the most commonly observed post-translational modification [1]. The dynamic balance between protein kinases and phosphatases, which reverse phosphorylation, forms cellular signaling networks [2, 3]. PKs are key players in these networks, often chaining their interactions to form signaling cascades [4]. Remarkably, interactions among PKs are predominantly intra-family, indicating that kinases from the same family are more likely to form functional clusters than to interact

with kinases from different families [5].

PKs regulate a wide array of cellular processes, from cell division and metabolism to transcription and apoptosis [6]. Beyond phosphorylation, they also function in protein complex scaffolding, allosteric regulation, subcellular targeting, and DNA binding [7]. Due to this functional diversity, dysfunctional PKs are often implicated in a range of diseases, including cancer [8], neurodegenerative [9], immunological [10], and hematological [11] disorders.

This chapter is structured as follows. First, we'll provide an overview of the PKs phylogenetic diversity. Next, we'll analyze the structure of a PK domain and describe its dynamic nature. Finally, we'll delve into how their dynamic architecture can be exploited for developing small molecules inhibiting their function.

2.1 Diversity of Protein Kinases

PKs that phosphorylate Ser, Thr, and Tyr residues are the most common and abundant. These PKs are found in eukaryotic, bacterial, and archeal cells and likely share a common evolutionary origin [12–14]. In prokaryotes, however, PKs exhibit a wider range of targets, and "non-canonical" phosphate acceptors like histidine are common [15].

Eukaryotic PKs (ePKs) comprise a large superfamily [16]. Traditionally, ePKs phylogeny was based on sequences of a catalytic domain, with early efforts distinguishing between PKs specific to Ser/Thr and Tyr [17]. The pivotal work of Manning [18] refined our understanding of the kinome by providing a comprehensive classification of PKs into groups, families, and subfamilies based on sequence homology and functional motifs (Fig. 2.2). This classification not only expanded the known kinome, constituting about 2% of human genes (to date, over 500 PKs were described), but also facilitated deeper insights into the functional and evolutionary relationships between different kinases [19].

Namely, Manning separated kinases into the following groups:

- **CAMK (Calcium/Calmodulin-Dependent Protein Kinase)**: These kinases are primarily involved in the transduction of calcium signals. They play a crucial role in various cellular processes including memory formation in neurons, cell cycle progression, and muscle contraction [21].

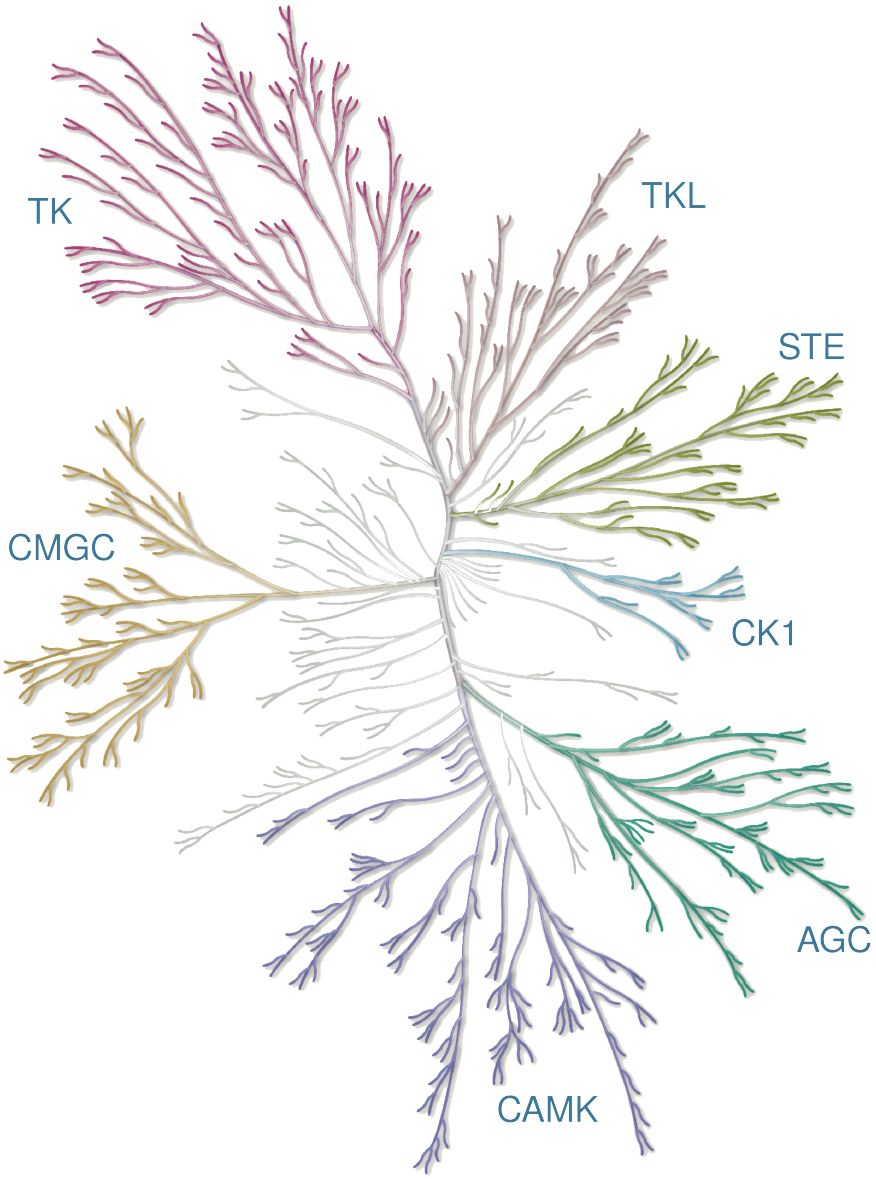

Figure 2.2: PK groups depicted on a phylogenetic tree constructed using the KinMap service [20].

- **AGC (Protein Kinase A, G, and C families)**: This group contains kinases that are regulated by second messengers, such as cyclic AMP (cAMP) and cyclic GMP (cGMP). They are involved in a wide range of processes including cell growth, proliferation, and metabolism [22].
- **CMGC (Cyclin-Dependent Kinases, Mitogen-Activated Pro-**

tein Kinases, Glycogen Synthase Kinases, and CDK-Like kinases): These kinases are central to cell cycle control and signal transduction pathways that respond to various extracellular stimuli. They are involved in cellular growth, division, differentiation, and stress responses [23].

- **CK1 (Casein Kinase 1)**: CK1 family members are involved in various cellular processes including DNA repair, cell cycle progression, and circadian rhythm regulation. They are known for their role in Wnt signaling pathway and RNA metabolism [24].
- **STE (Sterile Kinases)**: These kinases are a part of the MAP kinase signaling cascade, which is involved in processes such as cell growth and stress response [25].
- **TKL (Tyrosine Kinase-Like)**: This group includes a diverse set of kinases that share structural similarity with tyrosine kinases. For instance, it includes Raf family, whose members are often implicated in cancer [26], and LRRK family, which members' mutations are the most frequent cause of the Parkinson's disease [27].
- **TK (Tyrosine Kinase)**: Tyrosine kinases are critical in cell signaling pathways and are involved in the regulation of cell growth, differentiation, and survival [28]. They play a significant role in cancer progression as mutations in these kinases can lead to uncontrolled cell proliferation [29]. TKs are typically divided into receptor and non-receptor subgroups (RTKs and nRTKs) [30].

Alongside these groups, which share significant sequence similarity, Manning identified a small out-group of "atypical" kinases (aPKs). Although they maintain a tertiary structure akin to ePKs, aPKs exhibit limited sequence homology with them and possess distinct variations in conserved PK elements, such as the HRD motif (which appears as DRH in aPKs), the APE motif, and the G-rich loop (discussed further below) [31]. For certain aPKs, such as the aminoglycoside and choline kinase families, a common evolutionary origin with ePKs is suggested. However, other groups, like the phosphatidylinositol phosphate kinases, may have have evolved independently [32].

The methodology underlying the phylogenetic analysis of PKs continues to be refined and expanded [33]. However, as highlighted by Martin et al., relying solely on domain-based phylogeny might not always provide a comprehensive

view [34]. Given that PKs are multi-domain proteins with the catalytic domain often constituting only a small portion of the entire protein sequence, this approach can overlook a significant amount of sequence data. While it serves as a useful tool for grouping PKs based on the similarity of their catalytic domains, it may not fully capture the nuances of their evolutionary divergence and branching.

2.2 Protein Kinase Domain

2.2.1 PKD Structure

The Protein Kinase Domain (PKD) is generally composed of 250–350 residues and forms the core structural and functional unit of protein kinases [16, 33, 34]. This domain is characterized by a bilobal structure consisting of a smaller N-terminal lobe (N-lobe) and a larger C-terminal lobe (C-lobe), connected by a region commonly referred to as the "hinge" (Fig. 2.3a). The N-lobe is structured with a sequence of five β-sheets and a prominent α-helix, denoted as αC, arranged in the order: β_1–β_2–β_3–αC–β_4–β_5. Some PKs like Aurora kinase A have a small αB-helix between the β_3 and αC regions. In contrast, the C-lobe is predominantly composed of α-helices, labeled from D to I. The activation loop (AL), situated between the N and C lobes, collaborates with the hinge region, N-lobe's interior, and the AL to delineate the ATP binding pocket.

The PKD exhibits dynamic behavior, transitioning between various states during the enzymatic cycle, a process modulated by external stimuli such as AL phosphorylation, dimerization, and substrate binding [35–37]. This intricate process encompasses numerous semi-stable transition states that are interconnected, facilitating the complex regulation and signaling pathways in which kinases are involved [38, 39].

The active conformation of the PKD is a highly orchestrated state, necessitating the precise alignment of specific residues to facilitate effective catalysis [40]. This alignment is often elucidated through the formation of two integral structural elements: the catalytic (C) spine and the regulatory (R) spine, which are essential for optimal substrate positioning and catalytic activity [41, 42]:

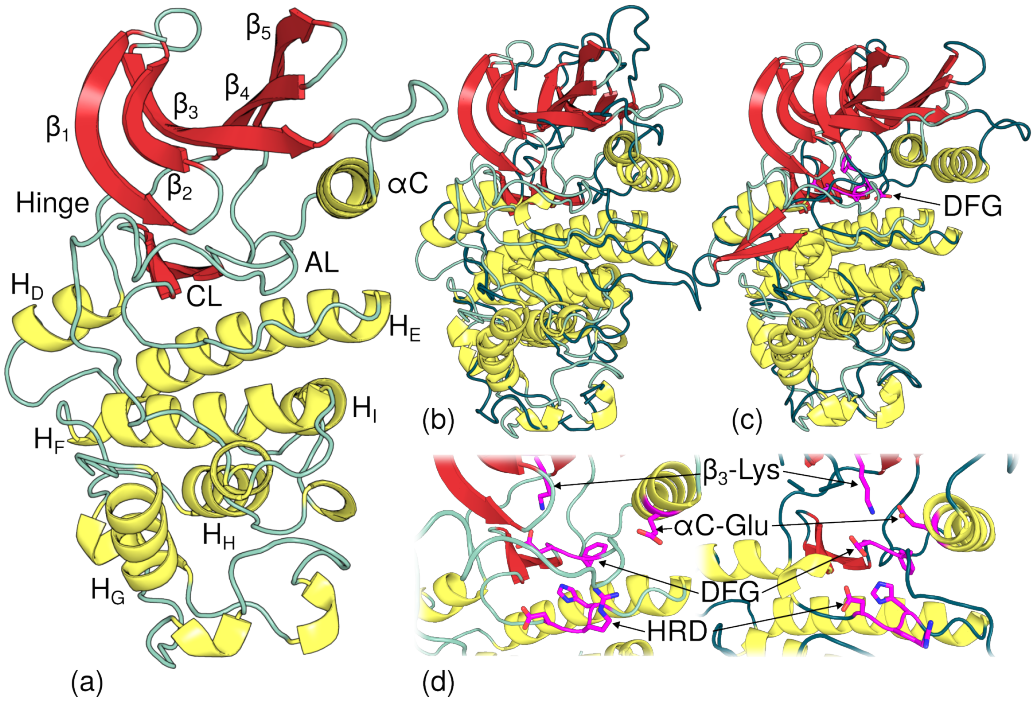

Figure 2.3: Protein kinase domains of CDK6 human protein (UniProt: Q00534; PDB codes: 1G3N:E, 1JOW:B, and 1BLX:A). **(a)** An inactive CDK6 structure (1BLX:A) with annotated structural regions. **(b)** The same CDK6 structure contrasted against an active structure (1JOW:B). **(c)** The same CDK6 structure contrasted against an inactive CDK6 structure (1G3N:E) adopting the DFG-out conformation. **(d)** Comparison of key conserved structural elements using structures depicted in (b).

- The **Catalytic Spine (C-spine)** is formed by the assembly of residues from both the N and C lobes, creating a hydrophobic spine that stabilizes the adenine ring of ATP. The key residues involved in the formation of the C-spine include those from the HRD motif (HRD-Arg), the HD-helix, the β3 strand, and the HF-helix.
- The **Regulatory Spine (R-spine)** is a dynamic structure comprising residues that undergo conformational changes during activation, aligning to form a continuous spine that stabilizes the active conformation. The R-spine includes residues from the DFG motif, the HRD motif (HRD-His), the αC-helix, and the β4-β5 loop.

AL plays a critical role in the regulation of kinase activity [43]. Positioned between the DFG and APE motifs, the AL is a dynamic and versatile segment

that undergoes substantial conformational changes, facilitating the transition between active and inactive states of the kinase [44, 45]. Phosphorylation of specific residues within the AL is a common regulatory mechanism, often serving as a switch that modulates kinase activity [46, 47].

In addition to its role in kinase regulation, the AL is also implicated in substrate recognition [48]. This region is characterized by its flexibility, adopting distinct conformations in different kinases and even exhibiting varied structures in the active and inactive states of the same kinase (Fig. 2.3). This dynamic nature is reflected in X-ray crystallography studies, where the AL often appears disordered, indicating its inherent flexibility and diverse range of motion.

The DFG-motif, demarcating the start of the activation loop, plays a crucial role in the activation dynamics [49, 50]. In the active state, the DFG-Asp points towards the active site, facilitating the binding of ATP and the transfer of the phosphate group. This orientation, termed the DFG-in conformation, is critical for proper substrate positioning. Some PKs can access an alternative, DFG-out conformation, where DFG-Asp and DFG-Phe swap places (Fig. 2.3c-d).

The dynamic nature of the DFG motif, and its ability to adopt different conformations, is central to the regulatory complexity of protein kinases, making it a focal point for understanding kinase activation and a potential target for therapeutic interventions. We'll discuss the DFG motif in great detail in the subsequent chapters.

The αC-helix, another pivotal structural element in shaping the conformational landscape of PKs, is frequently found to be disordered in X-ray structures [51]. In the active state, PKs adopt the αC-in conformation, where a conserved Glu residue on the αC-helix forms a salt bridge with a Lys residue on the β_3 strand, a critical interaction depicted in Fig. 2.3d. This salt bridge, a hallmark of kinase activation, facilitates the proper alignment of the catalytic and regulatory spines, thereby fostering a competent active site ready for substrate binding and phosphorylation. In contrast, the inactive state sees PKs adopting the αC-out conformation, during which the disruption of this salt bridge triggers a rearrangement in the kinase domain, hindering ATP and substrate binding and consequently inhibiting kinase activity [52].

The "gatekeeper" residue, situated just before the hinge region, serves a

critical role in modulating the accessibility of the ATP binding pocket in PKs [49, 53]. This residue essentially acts as a molecular gate, controlling the entry and exit of molecules at the ATP binding site. In many kinases, the size and nature of the gatekeeper residue can significantly influence the affinity and selectivity for ATP and small molecule inhibitors. Moreover, mutations at the gatekeeper position have been linked to altered kinase activity and are often implicated in drug resistance (elaborated below)[54, 55].

2.2.2 PKD Substrate Specificity

Another group of functionally important residues, besides those implicated in catalysis, are the ones determining substrate specificity, also known as specificity-determining residues (SDRs). Given the high structural similarity in the active state of PKD, it is logical to anticipate the emergence of sequence patterns that allow PKs to discriminate between substrates. This problem can be approached from two angles: (1) at the level of the phosphate acceptor residue, and (2) at the level of other substrate residues forming specific contacts with PKD. Both aspects have been studied extensively for Ser/Thr/Tyr kinases.

The ability to discriminate between different phosphoacceptor residues can be attributed to variations in their physical properties. The distinction is particularly pronounced between Ser/Thr and Tyr kinases; Tyr is a large aromatic residue, whereas Ser and Thr are smaller and differ by a single methyl group. The residues facilitating the discrimination between Ser/Thr and Tyr kinases can be identified from multiple sequence alignments (MSAs) by quantifying the conservation differences between the Ser/Thr and Tyr subsets [56]. For instance, residues following the HRD motif are small and non-polar in TKs (primarily Ala), accommodating Tyr. In turn, a Pro residue preceding the APE motif forms hydrophobic interactions with the phosphoacceptor Tyr, whereas Ser/Thr kinases contain Thr [57].

Furthermore, despite the declared specificity towards both Ser and Thr residues, PKs often exhibit different propensities towards these residues. As demonstrated by Chen et al., a single residue following the DFG motif (DFG+1) influences this propensity. Specifically, Phe at this position can accommodate Ser, but the additional methyl group of Thr makes this sterically challenging.

Conversely, DFG+1-Val favors Thr as a phosphoacceptor, and mutations between Phe and Val can effectively manipulate substrate specificity in PKs like STK20 [58].

However, these mechanisms are not universal. Some kinases, such as MEK1, MEK2, and DYRK (along with its prokaryotic relative [59]), are capable of phosphorylating both Ser/Thr and Tyr, thus being known as dual-specificity kinases [60]. For instance, DYRK (Down Syndrome Kinase) can phosphorylate its own Tyr in the activation loop[1], but can only phosphorylate Ser and Thr on the substrate protein [61, 62]. The structural mechanism underlying dual specificity remains to be elucidated.

From an evolutionary standpoint, Tyr specificity likely evolved from Ser/Thr specificity, possibly independently in eukaryotes and prokaryotes [63]. In eukaryotes, Tyr kinases probably emerged in a unicellular ancestor, while a significant expansion in RTKs is observed exclusively in multicellular organisms, attributed to the pivotal role these proteins play in intercellular communication [64].

The issue of PKs discriminating between different substrates proves to be considerably complex. While substrate patterns correlated with specific PKs can be elucidated due to the abundant phosphoproteomic data available, identifying substrate-specific residues in PK remains a formidable challenge. These SDRs seem to be dispersed throughout the PKD, forming a sparse network, and generally correlate with phylogenetic proximity [65–67]. Considering the diverse range of substrates, it is plausible that SDRs have evolved in tandem with them, adapting to facilitate specific interactions and functions.

2.3 PK inhibition strategies

As delineated above, many protein kinases, when misregulated, can give rise to severe diseases. For instance, gain-of-function mutations in RTKs lead to enhanced signaling and subsequent overexpression of the receptors, further amplifying the transmitted signal. Consequently, the cellular kinase domain becomes constitutively active, causing the signaling pathways they orchestrate, such as those controlling cell proliferation, to function at an elevated rate [8].

In a similar vein, the ABL1 kinase, a pivotal player in cellular signaling,

[1] A process known as autophosphorylation, a common mechanism of PK activation.

governs various cellular processes including cell differentiation, division, and adhesion [68]. Under normal circumstances, the activity of ABL1 is tightly regulated. However, chromosomal aberrations can lead to the formation of a BCR-ABL fusion protein, a constitutively active tyrosine kinase that drives uncontrolled cell proliferation, a hallmark of chronic myeloid leukemia (CML) [69, 70]. This aberrant kinase activity of ABL1 necessitated the development of targeted inhibitors to curb its activity.

2.3.1 PK Inhibitor Types

The discovery of Imatinib, a selective inhibitor of ABL1, marked a significant milestone in targeted cancer therapy [71, 72]. This drug binds to the kinase domain of ABL1, stabilizing it in an inactive conformation and thus halting its oncogenic signaling [73]. Since then, Imatinib was found to efficiently inhibit other kinases like c-KIT, DDR1, and CSF1R [68, 74]. More importantly, this discovery paved the way for the exploration of various inhibitor inhbition strategies [75, 76] and systematization of the discovered inhibitors into different types [77].

The efficacy of Imatinib stems from its ability to target the DFG-out conformation, a state not readily accessible by other kinases, thereby conferring high selectivity [78]. For instance, Imatinib exhibits diminished potency against the closely related Src protein kinase, which less readily adopts the DFG-out conformation [79–81]. Small molecules like Imatinib, which demonstrate selectivity towards the DFG-out conformation, have been classified as "Type-2" inhibitors. These inhibitors exploit a specific binding pocket that becomes available exclusively in the DFG-out state, as illustrated in Fig. 2.4b [82].

Contrastingly, drugs such as Dasatinib, which exhibits a higher affinity for Src compared to ABL1 and is employed in cases where ABL1 develops mutations conferring resistance to Imatinib [83][2], operate through a distinct mechanism. Termed as "Type-1" inhibitors, these compounds competitively obstruct ATP from accessing the binding pocket, thereby binding exclusively in the DFG-in conformation [42]. Given that this conformational state is akin to the active state conserved across PKs and is distinct from the DFG-out state, Type-1 inhibitors generally exhibit reduced selectivity, potentially leading to

[2] Although it may induce paradoxical activation of Src; refer to [84].

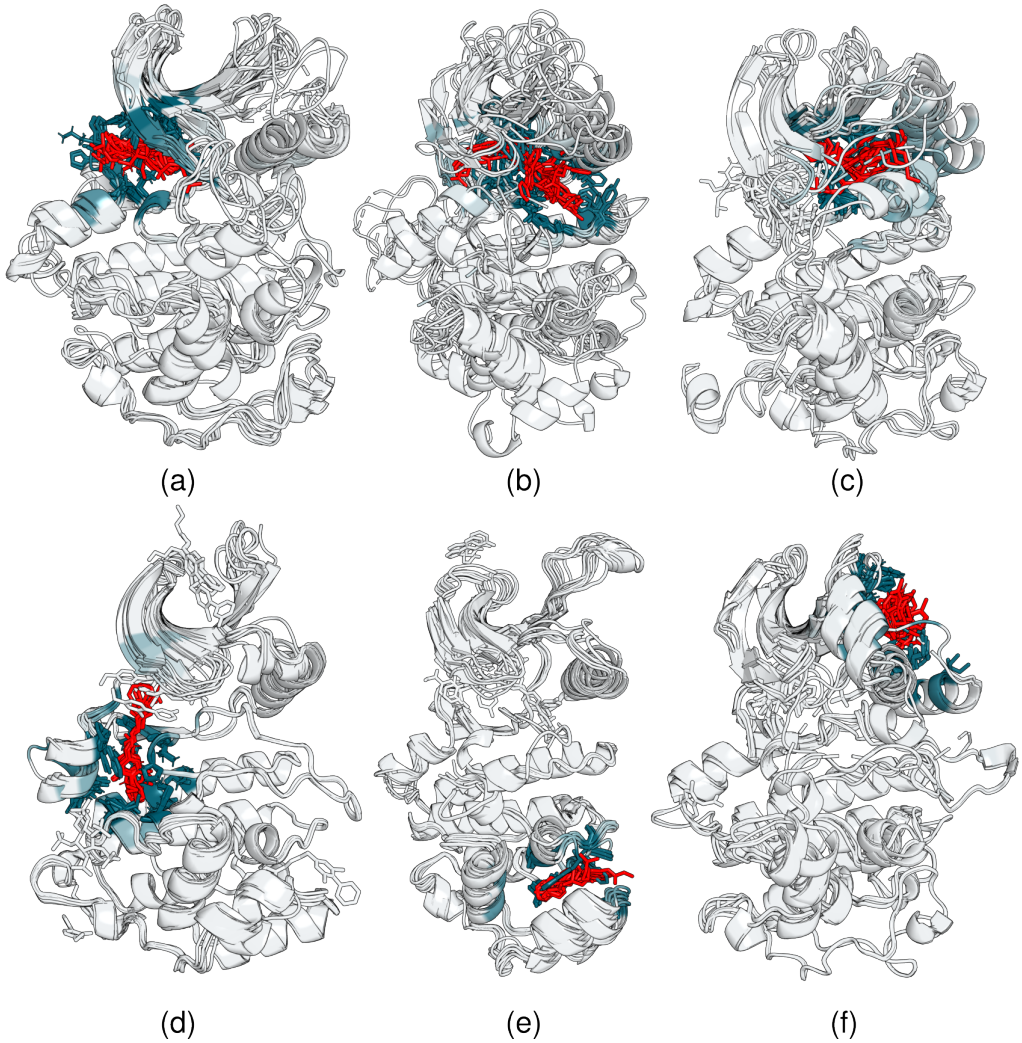

Figure 2.4: Examples of ligand binding pockets. **(a)** A Type-1 pocket, where ligands occupy the ATP binding site; some Type-1 ligands can extend deep into the pocket in the αC-out conforamtion. **(b)** Type-2 pocket, where the DFG-out conformation opens up a region between the AL and the αC-helix. **(c)** Type-1/2 pocket, where a combination DFG-in–αC-out allows small molecules to occupy a region between β_3, β_4, αC, and the AL N-proximal end. **(d)** A unique allosteric pocket opens up below the hinge, observed primarily in CSK kinases. **(e)** A distant allosteric pocket occupied primarily by small lipid molecules in; likely unique to MAPK14 proteins. **(f)** An allosteric pocket mostly observed in Aurora kinase A structures that can accomodate a ligand between αC and $\beta_4-\beta_5$ due to an additional αB helix.

adverse side effects [85].

Type-1/2 inhibitors, on the other hand, target a unique subpocket that emerges in the DFG-in–αC-out state. In this state, the disruption of the

αC-Glu–β_3-Lys salt bridge creates sufficient space to accommodate a small molecule, as depicted in Fig. 2.4c [53]. This pocket can be targeted by relatively large inhibitors extending towards the hinge region or by drugs that can synergize with small Type-1 ligands.

While Types 1, 2, and 1/2 encompass the majority of ligands that target the ATP binding pocket in various conformational states, there exist other categories of inhibitors that bind to minor, allosteric pockets, predominantly specific to groups of closely related kinases. Fig. 2.4d-e illustrate examples of such allosteric pockets. These inhibitors are typically grouped into categories such as Type-3, Type-4, and beyond, offering alternative avenues for kinase inhibition [77].

Inhibitor types and PK conformational variants were documented in numerous efforts. Databases such as KLIFS [86, 87], KinCore [88], and KinaMetrix [89] aggregate this information, facilitating the efforts of exploring the structural kinome and developing novel inhibtion strategies.

2.3.2 Resistance to PK Inhibitors

In the face of inherent predispositions or the high adaptability of diseases such as cancer, certain individuals may exhibit resistance to treatment with PK inhibitors. Primary resistance manifests in patients who do not respond to initial treatment, while secondary resistance develops over time, occurring after an initially successful treatment phase [90]. For instance, up to 25% of CML patients exhibit primary resistance to Imatinib, and 7–15% develop secondary resistance as the treatment progresses [91].

Given the complexity of the signaling pathways orchestrated by PKs and the substantial selective pressure to maintain these processes, resistance can emerge through various mechanisms. These mechanisms might involve alterations in the targeted protein or shifts in other components of the signaling networks and gene regulation [92, 93]. In the context of the target protein, resistance is typically acquired through specific mutations that render the binding of the initial inhibitor unfavorable.

For example, mutations dispersed across the P-loop, activation loop, hinge, and the C-lobe can confer resistance to Imatinib treatment in ABL1 [94]. A notable instance is the T315I gatekeeper mutation found in ABL1, as well

as in c-KIT, PDGFRA, and EGFR. This mutation engenders resistance to a spectrum of inhibitors, including Imatinib and Dasatinib [95]. The underlying mechanism of this resistance is attributed to the disruption of a vital hydrogen bond that the gatekeeper residue forms with Imatinib and analogous molecules, a bond that is absent in mutants such as T315I [96]. Nevertheless, this resistance can potentially be circumvented with alternative inhibitors, such as VX-680 [97]. This scenario underscores the necessity of adopting multifaceted inhibition strategies, which hinge on a profound understanding of the binding mechanisms of these drugs and the conformational plasticity inherent to PKs.

2.4 Concluding Remarks

Protein kinases, with their structural complexity and diversity, govern pivotal cellular processes. Their functional versatility is deeply rooted in their conformational plasticity. Despite over four decades of rigorous research, the full extent of their phylogenetic diversity and functional mechanisms remains elusive. The information available on PKs is abundant, and this review merely scratches the surface of this vast field. Nonetheless, it serves to acquaint readers with this vital segment of our proteome, emphasizing the pressing need to further refine our comprehension of the dynamic PK conformational landscape.

References

1. Khoury, G. A., Baliban, R. C. & Floudas, C. A. Proteome-wide post-translational modification statistics: frequency analysis and curation of the swiss-prot database. *Scient. Rep.* **1,** 90 (Sept. 2011).
2. Sacco, F., Perfetto, L., Castagnoli, L. & Cesareni, G. The human phosphatase interactome: An intricate family portrait. *FEBS Lett.* **586,** 2732–2739. ISSN: 0014-5793 (Aug. 2012).
3. Ardito, F., Giuliani, M., Perrone, D., Troiano, G. & Muzio, L. L. The crucial role of protein phosphorylation in cell signaling and its use as targeted therapy (Review). *Int. J. Molec. Medic.* **40,** 271–280. ISSN: 1107-3756 (2017).
4. Cuevas, B. D., Abell, A. N. & Johnson, G. L. Role of mitogen-activated protein kinase kinase kinases in signal integration. *Oncogene* **26,** 3159–3171. ISSN: 0950-9232 (May 2007).
5. Buljan, M. *et al.* Kinase Interaction Network Expands Functional and Disease Roles of Human Kinases. *Molec. Cell* **79,** 504–520.e9. ISSN: 1097-2765 (2020).
6. Cipak, L. Protein Kinases: Function, Substrates, and Implication in Diseases. *Int. J. Molec. Sci.* **23,** 3560 (Mar. 2022).
7. Rauch, J., Volinsky, N., Romano, D. & Kolch, W. The secret life of kinases: functions beyond catalysis. *Cell Comm. & Sign.* **9,** 23. ISSN: 1478-811X (Oct. 2011).
8. Du, Z. & Lovly, C. M. Mechanisms of receptor tyrosine kinase activation in cancer. *Molec. Canc.* **17,** 58 (2018).
9. Benn, C. L. & Dawson, L. A. Clinically Precedented Protein Kinases: Rationale for Their Use in Neurodegenerative Disease. *Front. Aging Neurosci.* **12,** 242. ISSN: 1663-4365 (Sept. 2020).
10. Castelo-Soccio, L. *et al.* Protein kinases: drug targets for immunological disorders. *Nat. Rev. Immun.,* 1–20. ISSN: 1474-1733 (May 2023).
11. Okay, M. & Haznedaroglu, I. C. Protein Kinases in Hematological Disorders. *Adv. Exper. Medic. & Biol.* **1275,** 383–393. ISSN: 0065-2598 (Jan. 2021).
12. Leonard, C. J., Aravind, L. & Koonin, E. V. Novel Families of Putative Protein Kinases in Bacteria and Archaea: Evolution of the "Eukaryotic" Protein Kinase Superfamily. *Genome Res.* **8,** 1038–1047. ISSN: 1088-9051 (1998).

13. Cousin, C. *et al.* Protein-serine/threonine/tyrosine kinases in bacterial signaling and regulation. *FEMS Microb. Lett.* **346,** 11–19. ISSN: 0378-1097 (Sept. 2013).

14. Stancik, I. A. *et al.* Serine/Threonine protein kinases from bacteria, archaea and eukarya share a common evolutionary origin deeply rooted in the tree of life. en. *J. Mol. Biol.* **430,** 27–32. ISSN: 00222836 (2018).

15. Stewart, R. C. Protein histidine kinases: assembly of active sites and their regulation in signaling pathways. *Curr. Opin. Microb.* **13,** 133–141. ISSN: 1369-5274 (Apr. 2010).

16. Hanks, S. K. & Hunter, T. The eukaryotic protein kinase superfamily: kinase (catalytic) domain structure and classification. *The FASEB J.* **9,** 576–596. ISSN: 0892-6638 (May 1995).

17. Hanks, S. K., Quinn, A. M. & Hunter, T. The Protein Kinase Family: Conserved Features and Deduced Phylogeny of the Catalytic Domains. *Science* **241,** 42–52. ISSN: 0036-8075 (1988).

18. Manning, G. The protein kinase complement of the human genome. en. *Science* **298,** 1912–1934. ISSN: 00368075, 10959203 (2002).

19. Manning, G., Plowman, G. D., Hunter, T. & Sudarsanam, S. Evolution of protein kinase signaling from yeast to man. *Trends Biochem. Sci.* **27,** 514–520. ISSN: 0968-0004 (Oct. 2002).

20. Eid, S., Turk, S., Volkamer, A., Rippmann, F. & Fulle, S. KinMap: a web-based tool for interactive navigation through human kinome data. *BMC Bioinf.* **18,** 16 (Jan. 2017).

21. Swulius, M. T. & Waxham, M. N. Ca2+/Calmodulin-dependent Protein Kinases. *Cellul. & Molec. Life Sci.* **65,** 2637. ISSN: 1420-682X (2008).

22. Pearce, L. R., Komander, D. & Alessi, D. R. The nuts and bolts of AGC protein kinases. *Nat. Rev. Molec. Cell Biol.* **11,** 9–22. ISSN: 1471-0072 (Jan. 2010).

23. Varjosalo, M. *et al.* The Protein Interaction Landscape of the Human CMGC Kinase Group. *Cell Rep.* **3,** 1306–1320. ISSN: 2211-1247 (Apr. 2013).

24. Schittek, B. & Sinnberg, T. Biological functions of casein kinase 1 isoforms and putative roles in tumorigenesis. *Molec. Canc.* **13,** 231. ISSN: 1476-4598 (Oct. 2014).

25. Dan, I., Watanabe, N. M. & Kusumi, A. The Ste20 group kinases as regulators of MAP kinase cascades. *Trends Cell Biol.* **11,** 220–230. ISSN: 0962-8924 (May 2001).

26. Matallanas, D. *et al.* Raf Family Kinases: Old Dogs Have Learned New Tricks. *Genes & Canc.* **2,** 232–260. ISSN: 1947-6019 (2011).

27. Gilsbach, B. K. & Kortholt, A. Structural biology of the LRRK2 GTPase and kinase domains: implications for regulation. *Front. Molec. Neurosci.* **7,** 32. ISSN: 1662-5099 (May 2014).

28. Bagnato, G., Leopizzi, M., Urciuoli, E. & Peruzzi, B. Nuclear Functions of the Tyrosine Kinase Src. *Int. J. Molec. Sci.* **21,** 2675 (Apr. 2020).

29. Paul, M. K. & Mukhopadhyay, A. K. Tyrosine kinase – Role and significance in Cancer. *Int. J. Molec. Sci.* **1,** 101–115. ISSN: 1449-1907 (June 2004).

30. Dev, S. S., Abidin, S. A. Z., Farghadani, R., Othman, I. & Naidu, R. Receptor Tyrosine Kinases and Their Signaling Pathways as Therapeutic Targets of Curcumin in Cancer. *Front. Pharmoc.* **12,** 772510. ISSN: 1663-9812 (Nov. 2021).

31. Kanev, G. K. *et al.* The Landscape of Atypical and Eukaryotic Protein Kinases. *Trends Pharm. Sci.* **40,** 818–832. ISSN: 0165-6147 (2019).

32. Scheeff, E. D. & Bourne, P. E. Structural Evolution of the Protein Kinase–Like Superfamily. *PLoS Comp. Biol.* **1,** e49. ISSN: 1553-734X (2005).

33. Modi, V. & Dunbrack, R. L. A structurally-validated multiple sequence alignment of 497 human protein kinase domains. *Scient. Rep.* **9,** 19790. ISSN: 2045-2322 (2019).

34. Martin, J., Anamika, K. & Srinivasan, N. Classification of Protein Kinases on the Basis of Both Kinase and Non-Kinase Regions. *PLOS One* **5,** e12460 (Sept. 2010).

35. Huse, M. & Kuriyan, J. The conformational plasticity of protein kinases. *Cell* **109,** 275–282. ISSN: 0092-8674 (2002).

36. Xie, T., Saleh, T., Rossi, P. & Kalodimos, C. G. Conformational states dynamically populated by a kinase determine its function. *Science* **370,** eabc2754. ISSN: 0036-8075 (2020).

37. Bae, J. H. & Schlessinger, J. Asymmetric tyrosine kinase arrangements in activation or autophosphorylation of receptor tyrosine kinases. *Molec. & Cells* **29,** 443–448. ISSN: 1016-8478 (2010).

38. McClendon, C. L., Kornev, A. P., Gilson, M. K. & Taylor, S. S. Dynamic architecture of a protein kinase. en. *Proc. Natl. Acad. Sci. USA* **111,** E4623–E4631. ISSN: 0027-8424, 1091-6490 (Oct. 2014).

39. Kim, J. *et al.* A dynamic hydrophobic core orchestrates allostery in protein kinases. *Sci. Advanc.* **3,** e1600663 (Apr. 2017).

40. Modi, V. & Dunbrack, R. L. Defining a new nomenclature for the structures of active and inactive kinases. *Proc. Natl. Acad. Sci. USA* **116,** 6818 (2019).

41. James, K. A. & Verkhivker, G. M. Structure-Based Network Analysis of Activation Mechanisms in the ErbB Family of Receptor Tyrosine Kinases: The Regulatory Spine Residues Are Global Mediators of Structural Stability and Allosteric Interactions. *PLOS One* **9,** e113488 (Nov. 2014).

42. Arter, C., Trask, L., Ward, S., Yeoh, S. & Bayliss, R. Structural features of the protein kinase domain and targeted binding by small-molecule inhibitors. *J. Biol. Chem.* **298,** 102247. ISSN: 0021-9258 (2022).

43. Nolen, B., Taylor, S. & Ghosh, G. Regulation of Protein Kinases Controlling Activity through Activation Segment Conformation. *Molec. Cell* **15,** 661–675. ISSN: 1097-2765 (Sept. 2004).

44. Pucheta-Martínez, E. *et al.* An Allosteric Cross-Talk Between the Activation Loop and the ATP Binding Site Regulates the Activation of Src Kinase. *Scient. Rep.* **6,** 24235 (Apr. 2016).

45. Iverson, D. B., Xiao, Y., Jones, D. N., Eisenmesser, E. Z. & Ahn, N. G. Activation Loop Dynamics Are Coupled to Core Motions in Extracellular Signal-Regulated Kinase-2. *Biochemistry* **59,** 2698–2706. ISSN: 0006-2960 (July 2020).

46. Steichen, J. M. *et al.* Global Consequences of Activation Loop Phosphorylation on Protein Kinase A*. *J. Biol. Chem.* **285,** 3825–3832. ISSN: 0021-9258 (Feb. 2010).

47. Köhler, M. *et al.* Activation loop phosphorylation regulates B-Raf in vivo and transformation by B-Raf mutants. *EMBO J.* **35,** 143–161. ISSN: 0261-4189 (Jan. 2016).

48. Joshi, M. K. *et al.* Substrate binding to Src: A new perspective on tyrosine kinase substrate recognition from NMR and molecular dynamics. *Prot. Sci.* **29,** 350–359. ISSN: 0961-8368 (Feb. 2020).

49. Treiber, D. K. & Shah, N. P. Ins and Outs of Kinase DFG Motifs. *Chem. & Biol.* **20,** 745–746. ISSN: 1074-5521 (June 2013).

50. Fabbro, D., Cowan-Jacob, S. W. & Moebitz, H. Ten things you should know about protein kinases. *Brit. J. Pharm.* **172,** 2675–2700. ISSN: 0007-1188 (2015).

51. Gógl, G., Kornev, A. P., Reményi, A. & Taylor, S. S. Disordered Protein Kinase Regions in Regulation of Kinase Domain Cores. *Trends Biochem. Sci.* **44,** 300–311. ISSN: 0968-0004 (Apr. 2019).

52. Huang, H., Zhao, R., Dickson, B. M., Skeel, R. D. & Post, C. B. αC Helix as a Switch in the Conformational Transition of Src/CDK-like Kinase Domains. *J. Phys. Chem. B* **116,** 4465–4475. ISSN: 1520-6106 (Apr. 2012).

53. Zuccotto, F., Ardini, E., Casale, E. & Angiolini, M. Through the "Gatekeeper Door": Exploiting the Active Kinase Conformation. *J. Medicin. Chem.* **53,** 2681–2694. ISSN: 0022-2623 (Apr. 2010).

54. Azam, M., Seeliger, M. A., Gray, N. S., Kuriyan, J. & Daley, G. Q. Activation of tyrosine kinases by mutation of the gatekeeper threonine. *Nat. Struct. Mol. Biol.* **15,** 1109–1118. ISSN: 1545-9993 (2008).

55. Zhou, Y., Xiang, S., Yang, F. & Lu, X. Targeting Gatekeeper Mutations for Kinase Drug Discovery. *J. Medicin. Chem.* **65,** 15540–15558. ISSN: 0022-2623 (Dec. 2022).

56. Singh, J. Comparison of conservation within and between the Ser/Thr and Tyr protein kinase family: proposed model for the catalytic domain of the epidermal growth factor receptor. *Prot. Engin. Des. Sel.* **7,** 849–858. ISSN: 1741-0126 (1994).

57. Taylor, S. S., Radzio-Andzelm, E. & Hunter, T. How do protein kinases discriminate between serine/threonine and tyrosine? Structural insights from the insulin receptor protein-tyrosine kinase. *The FASEB J.* **9,** 1255–1266. ISSN: 0892-6638 (1995).

58. Chen, C. *et al.* Identification of a Major Determinant for Serine-Threonine Kinase Phosphoacceptor Specificity. *Molec. Cell* **53,** 140–147. ISSN: 1097-2765 (2014).

59. Arora, G. *et al.* Unveiling the Novel Dual Specificity Protein Kinases in Bacillus anthracis. *J. Biol. Chem.* **287,** 26749–26763. ISSN: 0021-9258 (Aug. 2012).

60. Dhanasekaran, N. & Reddy, E. P. Signaling by dual specificity kinases. *Oncogene* **17,** 1447–1455. ISSN: 0950-9232 (Sept. 1998).

61. Walte, A. *et al.* Mechanism of dual specificity kinase activity of DYRK1A. *FEBS Lett.* **280,** 4495–4511. ISSN: 1742-464X (Sept. 2013).

62. Soundararajan, M. *et al.* Structures of Down Syndrome Kinases, DYRKs, Reveal Mechanisms of Kinase Activation and Substrate Recognition. *Structure* **21,** 986–996. ISSN: 0969-2126 (June 2013).

63. Hunter, T. The Genesis of Tyrosine Phosphorylation. *Cold Springs Harbor Persp. Biol.* **6,** a020644 (May 2014).

64. Srivastava, M. *et al.* The Amphimedon queenslandica genome and the evolution of animal complexity. *Nature* **466,** 720–726. ISSN: 0028-0836 (Aug. 2010).

65. Creixell, P. *et al.* Unmasking Determinants of Specificity in the Human Kinome. *Cell* **163,** 187–201. ISSN: 0092-8674 (Sept. 2015).

66. Bradley, D. & Beltrao, P. Evolution of protein kinase substrate recognition at the active site. *PLoS Biol.* **17,** e3000341. ISSN: 1544-9173 (June 2019).

67. Bradley, D. *et al.* Sequence and Structure-Based Analysis of Specificity Determinants in Eukaryotic Protein Kinases. *Cell Rep.* **34,** 108602. ISSN: 2211-1247 (Jan. 2021).

68. Wang, J. Y. J. The Capable ABL: What Is Its Biological Function? *Molec. & Cellul. Biol.* **34,** 1188–1197 (2014).

69. Greuber, E. K., Smith-Pearson, P., Wang, J. & Pendergast, A. M. Role of ABL family kinases in cancer: from leukaemia to solid tumours. *Nat. Rev. Canc.* **13,** 559–571. ISSN: 1474-175X (2013).

70. Soverini, S., Mancini, M., Bavaro, L., Cavo, M. & Martinelli, G. Chronic myeloid leukemia: the paradigm of targeting oncogenic tyrosine kinase signaling and counteracting resistance for successful cancer therapy. *Molec. Canc.* **17,** 49 (2018).

71. Capdeville, R., Buchdunger, E., Zimmermann, J. & Matter, A. Glivec (STI571, imatinib), a rationally developed, targeted anticancer drug. *Nat. Rev. Drug Disc.* **1,** 493–502. ISSN: 1474-1776 (2002).

72. Iqbal, N. & Iqbal, N. Imatinib: a breakthrough of targeted therapy in cancer. en. *Chemother. Res. Pract.* **2014,** 1–9. ISSN: 2090-2107, 2090-2115 (2014).

73. Schindler, T. *et al.* Structural Mechanism for STI-571 Inhibition of Abelson Tyrosine Kinase. *Science* **289,** 1938–1942 (2000).

74. Mol, C. D. *et al.* Structural Basis for the Autoinhibition and STI-571 Inhibition of c-Kit Tyrosine Kinase. *J. Biol. Chem.* **279,** 31655–31663. ISSN: 0021-9258 (2004).

75. Roskoski, R. A historical overview of protein kinases and their targeted small molecule inhibitors. *Pharm. Res.* **100,** 1–23. ISSN: 1043-6618 (2015).

76. Cohen, P., Cross, D. & Jänne, P. A. Kinase drug discovery 20 years after imatinib: progress and future directions. *Nat. Rev. Drug Disc.* **20,** 551–569. ISSN: 1474-1776 (2021).

77. Roskoski, R. Classification of small molecule protein kinase inhibitors based upon the structures of their drug-enzyme complexes. *Pharm. Res.* **103,** 26–48. ISSN: 1043-6618 (2016).

78. Ayaz, P. *et al.* Structural mechanism of a drug-binding process involving a large conformational change of the protein target. *Nat. Commun.* **14,** 1885 (2023).

79. Aleksandrov, A. & Simonson, T. Molecular dynamics simulations show that conformational selection governs the binding preferences of imatinib for several tyrosine kinases. en. *J. Biol. Chem.* **285,** 13807–13815. ISSN: 0021-9258, 1083-351X. (2020) (2010).

80. Agafonov, R. V., Wilson, C., Otten, R., Buosi, V. & Kern, D. Energetic dissection of Gleevec's selectivity toward human tyrosine kinases. *Nat. Struct. Mol. Biol.* **21,** 848–853. ISSN: 1545-9993 (2014).

81. Morando, M. A. *et al.* Conformational selection and induced fit mechanisms in the binding of an anticancer drug to the c-Src kinase. *Scient. Rep.* **6,** 24439 (2016).

82. Zhao, Z. *et al.* Exploration of Type II Binding Mode: A Privileged Approach for Kinase Inhibitor Focused Drug Discovery? *ACS Chemic. Biol.* **9,** 1230–1241. ISSN: 1554-8929 (2014).

83. Aguilera, D. G. & Tsimberidou, A. M. Dasatinib in chronic myeloid leukemia: a review. *Therap. & Clinic. Risk. Manag.* **5,** 281–289. ISSN: 1176-6336 (Jan. 2009).

84. Higuchi, M. *et al.* Paradoxical activation of c-Src as a drug-resistant mechanism. *Cell Rep.* **34,** 108876. ISSN: 2211-1247 (Mar. 2021).

85. Nekoukar, Z., Moghimi, M. & Salehifar, E. A narrative review on adverse effects of dasatinib with a focus on pharmacotherapy of dasatinib-induced pulmonary toxicities. *Blood Res.* **56,** 229–242. ISSN: 2287-979X (Dec. 2021).

86. Kooistra, A. J. *et al.* KLIFS: a structural kinase-ligand interaction database. *Nucl. Acids Res.* **44,** D365–D371. ISSN: 0305-1048 (2016).

87. Kanev, G. K., de Graaf, C., Westerman, B. A., de Esch, I. J. P. & Kooistra, A. J. KLIFS: an overhaul after the first 5 years of supporting kinase research. *Nucl. Acids Res.* **49,** gkaa895–. ISSN: 0305-1048 (2020).

88. Modi, V. & Dunbrack, R. L. Kincore: a web resource for structural classification of protein kinases and their inhibitors. *Nucl. Acids Res.* **50,** gkab920–. ISSN: 0305-1048 (Oct. 2021).

89. Rahman, R., Ung, P. M.-U. & Schlessinger, A. KinaMetrix: a web resource to investigate kinase conformations and inhibitor space. *Nucl. Acids Res.* **47,** D361–D366. ISSN: 0305-1048 (2019).

90. Simasi, J., Schubert, A., Oelkrug, C., Gillissen, A. & Nieber, K. Primary and secondary resistance to tyrosine kinase inhibitors in lung cancer. *Anticanc. Res.* **34,** 2841–50 (June 2014).

91. Barouch-Bentov, R. & Sauer, K. Mechanisms of drug resistance in kinases. *Exp. Opin. Invest. Drugs* **20,** 153–208. ISSN: 1354-3784 (2011).

92. Sierra, J. R., Cepero, V. & Giordano, S. Molecular mechanisms of acquired resistance to tyrosine kinase targeted therapy. *Molec. Canc.* **9,** 75 (2010).

93. Yang, Y., Li, S., Wang, Y., Zhao, Y. & Li, Q. Protein tyrosine kinase inhibitor resistance in malignant tumors: molecular mechanisms and future perspective. *Sign. Transduc. Targ. Ther.* **7,** 329. ISSN: 2095-9907 (Sept. 2022).

94. Balzano, D., Santaguida, S., Musacchio, A. & Villa, F. A General Framework for Inhibitor Resistance in Protein Kinases. *Chem. & Biol.* **18,** 966–975. ISSN: 1074-5521 (2011).

95. Mondal, J., Tiwary, P. & Berne, B. J. How a Kinase Inhibitor Withstands Gatekeeper Residue Mutations. *J. Am. Chem. Soc.* **138,** 4608–4615. ISSN: 0002-7863 (Apr. 2016).

96. Krishnamurty, R. & Maly, D. J. Biochemical Mechanisms of Resistance to Small-Molecule Protein Kinase Inhibitors. *ACS Chemic. Biol.* **5,** 121–138. ISSN: 1554-8929 (2010).

97. Carter, T. A. *et al.* Inhibition of drug-resistant mutants of ABL, KIT, and EGF receptor kinases. *Proc. Natl. Acad. Sci. USA* **102,** 11011–11016. ISSN: 0027-8424 (Aug. 2005).

Chapter 3

Classifying protein kinase conformations with machine learning

Abstract

Protein kinases are key actors of signaling networks and important drug targets. They cycle between active and inactive conformations, distinguished by a few elements within the catalytic domain. One is the activation loop, whose conserved DFG motif can occupy DFG-in, DFG-out, and some rarer conformations. Annotation and classification of the structural kinome is important, as different conformations can be targeted by different inhibitors and activators. Valuable resources exist; however, large-scale applications will benefit from increased automation and interpretability of structural annotation. Interpretable machine learning models are described for this purpose, based on ensembles of decision trees. To train them, a set of catalytic domain sequences and structures was collected, somewhat larger and more diverse than existing resources. The structures were clustered based on the DFG conformation and manually annotated. They were then used as training input. Two main models were constructed, which distinguished active/inactive and in/out/other DFG conformations. They considered initially thousands of structural variables, spanning the whole catalytic domain, then identified ("learned") a small subset that sufficed for accurate classification. The first model correctly labeled

all but 3 of 3284 structures as active or inactive, while the second assigned the correct DFG label to all but 17 of 8826 structures. The most potent classifying variables were all related to well-known structural elements in or near the activation loop and their ranking gives insights into the conformational preferences. The models were used to automatically annotate 3850 kinase structures predicted recently with the Alphafold2 tool, showing that Alphafold2 reproduced the active/inactive but not the DFG-in proportions seen in the Protein Data Bank. We expect the models will be useful for understanding and engineering kinases.

3.1 Introduction

Protein kinases (PK) are one of the largest and oldest groups of enzymes [1, 2]. They catalyze the transfer of a phosphate from ATP onto specific side chains of target proteins, modulating the targets' activity. As key elements of signaling networks, they are highly regulated, and their dysfunction can lead to various diseases, including cancers [3, 4]. Thus, considerable efforts are made to develop molecules that inhibit them.

Like other ATPases, kinases cycle between an active and an inactive conformation, depending on their own phosphorylation state [1, 5–8]. Kinase active conformations are all similar, despite substantial sequence diversity [5, 9]. The inactive conformation has fewer functional constraints and is more variable. The active and inactive conformations differ in a few key elements. One is the so-called activation loop, or A-loop, which includes a conserved DFG motif. Another is the C helix (Fig. 3.1a). In the active state, the DFG motif adopts an inward-facing, "DFG-in" conformation, where its Asp side chain can coordinate an active site Mg^{2+} and help orient the ATP substrate (Fig. 3.1c-d). In the inactive state of many PKs, the DFG motif is flipped, with its Asp and Phe residues rotated by 180°, a conformation known as "DFG-out". The bulky Phe then occupies the ATP binding pocket, precluding ATP binding [10, 11]. In other PKs, like Src, the inactive state has a DFG-in conformation, as detailed below. One strategy to achieve inhibitor specificity has been to target the inactive conformation [7, 12]; for example, inhibitors that lock PKs in the DFG-out state have proven effective against chronic myeloid leukemia [13–15].

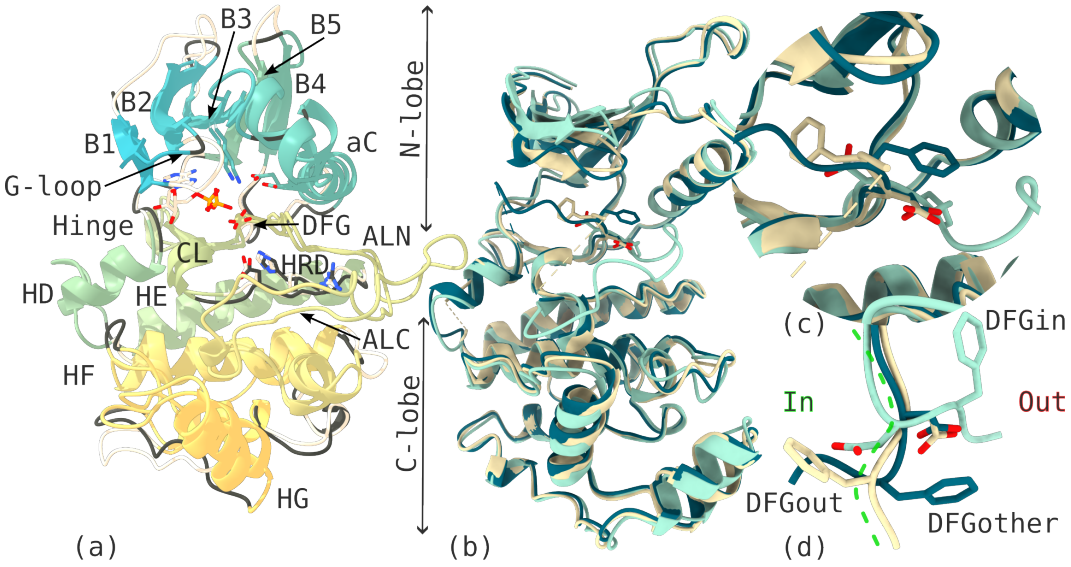

Figure 3.1: The protein kinase catalytic domain. **(a)** Main structural regions, shown on Src (PDB code 1YI6) and ADP-bound Aurora-A (PDB code 1OL7), both in the active conformation. N-lobe and C-lobe are labeled, with the ATP-binding pocket and activation loop (subdivided into ALN and ALC) sandwiched in between. **(b)** Three Mitogen-Activated Kinase-14 structures with different DFG conformations: DFG-in (blue, PDB code 1A9U:A), DFG-out (yellow, 1KV1:A), and DFG-other (red, 3BV2:A). **(c)** Close-up of the DFG region. **(d)** A view of the DFG residues from the top (N-lobe side). The green line represents an activation loop backbone loosely dividing the binding pocket into subregions ”inside” and ”outside”. With DFG-in, DFG-Asp is inside and and DFG-Phe outside; conversely with DFG-out. In the DFG-other conformation shown, both residues are pushed outside of the binding pocket towards the C helix.

The conformational landscape of PKs has been studied extensively by X-ray crystallography [16–20], and also by simulations [21–25]. There are over 6000 structures in the PDB, and several databases provide classifications of kinase structures according to their active or inactive state, and the nature of their inactive conformation. The KLIFS database, at the time of writing, included 6295 structures, classified by a decision tree algorithm trained on manually curated data [26–28]. Dunbrack and coworkers maintain a similar database, annotated based on clustering of distances and dihedral angles [29, 30]. These resources are highly valuable for drug design, as structural classification can facilitate comparisons between kinases and inferences about which ligands are likely to be useful leads. On the other hand, manual or partly-manual annotation by human experts can be difficult to generalize to large-scale studies,

as more and more structures become available.

In this context, the accuracy of automatically labeling the structural kinome is paramount. Here, we describe machine learning tools for this purpose. Machine learning (ML) is increasingly proposed as a tool for large-scale classification and prediction in structural biology [31]. Prominent applications include predictions of protein structure [32–34], function [35] and interaction [36, 37], protein design [38–40] and drug discovery [41, 42].

As a prerequisite to develop and apply ML classification tools, we created our own kinase structural collection, somewhat larger and more diverse than previous ones. The collection groups kinase catalytic domains present in PDB structures. Its construction was made reproducible and extensible, thanks to open-source software for extracting structural variables, or "features". The collection contains pairs of UniProt and PDB entries, with sequences mapped to a single kinase Hidden Markov Model (HMM) [43] from Pfam [44], which can be thought of as a sequence profile or alignment and serves as a reference. A comprehensive set of sequence and structure descriptors and metadata accompany each pair. The structural domains were then superimposed and clustered, based on the DFG motif's coordinates, which facilitated the exploration and annotation of the entire dataset in all its diversity. The annotations included active/inactive labels, and in/out/other labels for the DFG motif. These were compared to existing kinase classifications, revealing good agreement but also a few discrepancies, which are discussed.

With annotated structures in place, several ML models were constructed, designed to discriminate either between active/inactive structures, or between the main classes of DFG structures: DFG-in, DFG-out, and "intermediate" or "DFG-other". The models were based on decision tree ensembles trained by gradient boosting [45, 46]. During the learning process, the models came to rank input variables, which we could exploit post-hoc using an iterative, unsupervised selection protocol known as the "Boruta algorithm" [47]. The variables were structural features such as dihedral torsion angles and residue–residue distances. As a result, our models were interpretable and revealed intuitive structural hallmarks associated with PK conformations, similar to those employed in manual, expert annotations. The ML models were carefully trained to avoid overfitting and achieved excellent accuracy, with just 3 errors for active/inactive classification (among 3284 cases) and 17 errors for inactive

DFG classifications (among 8826 cases). Inactive errors arose from structures near the edge of the DFG-in conformational ensemble.

To illustrate the usefulness of the annotated dataset and the trained models, we chose a current problem: annotating a large collection of kinase structures predicted recently by the Alphafold2 tool [33]. This tool was shown to provide high-quality protein models and is being widely used in structural biology. However, when it predicts a kinase structure, Alphafold2 provides coordinates but does not indicate or "label" the conformational state. To help interpret its predictions, our ML models were used to annotate 3850 structures predicted by AlphaFold2, corresponding to curated UniProt sequences (from Swiss-Prot [48]). The structures were obtained from a database of predicted structures made available by the Alphafold developers [49]. All had been judged [49] to have a high overall accuracy, although the accuracies of the active site conformations are intrinsically harder to evaluate and not precisely known. Our ML models categorized the predicted structures into 2749/1101 active/inactive structures, and 3761 DFG-in, 74 DFG-out, and 15 DFG-other structures. These proportions are close to the active/inactive but not the DFG-in proportions seen in the PDB.

Beyond this illustrative problem, we expect our database and ML models will be of use for other kinase applications, including lead selection in drug design. In addition, the methodology and software can be applied to other problems in structural biology, such as feature extraction and selection from other domains and sequence patterns, increasing the ease, transparency, and reproducibility of structural biology data mining.

3.2 Results

3.2.1 An extensive data collection of kinase catalytic domains

Kinase sequences were obtained from the the SIFTS database (Structure Integration with Function, Taxonomy, and Sequence) [50], which contained 60567 sequences in all. Among these, we found 664 PK sequences. They were represented in the PDB by 8016 chains, 7547 of which were X-ray structures. 6300 of these passed our filtering for size and coverage (70% of the catalytic

domain contained in the structure). Counting the sequences with at least one PK domain resulted in 431 unique sequences.

Supplementary Figure 3.9 depicts statistics of this collection. Sequences were mapped to a kinase Hidden Markov Model (HMM) [43] from Pfam [44] (see Methods), which can be thought of as a sequence profile or alignment and serves as a reference. Coverage of the HMM positions was high and almost uniform, and similar for the "structural" sequences (those in the PDB) and the "canonical" sequences (those from Uniprot). Conversely, sequence coverage by HMM positions was high, as were the similarity scores for domain matching (Supp. Fig. 3.9b,c). Domain sizes peaked at 250, rarely descending below 200. Finally, structural sequences matched canonical sequences closely (Supp. Fig. 3.9f): PDB structures contained few mutations. Together, these results indicate successful domain capture and high-quality sequence data.

We compared our data to four existing resources: (1) ABC [51], (2) KinaMetrix [52], (3) KLIFS [28] and (4) KinCore [29] (Table 3.1). KLIFS used human and mouse proteins, KinCore used proteins from ten model organisms, and ABC and KinaMetrix were not limited to particular organisms. Our data collection was somewhat larger and more diverse, with for example 88 more unique sequences than the most recent KinCore (see also Supplementary Fig. 3.10 with categorized sequence counts). Labels in the different datasets (active/inactive and so on) are described and compared further on.

Table 3.1: Kinase resource comparison

Resource	Chains	Structures	Sequences	Families	Organisms
ABC	4223	2921	274	9	15
KinaMetrix	3569	3555	321	9	28
KLIFS	8402	5836	330	13	3
KinCore	9136	6040	343	9	10
This work	9459	6300	431	9	53

3.2.2 Annotating the substrate binding pocket

Our kinase dataset included 8273 PDB chains that had a bound ligand and 1186 with no ligand. We reviewed the most frequently mapped ligands, distinguishing (1) ATP-like and (2) non-ATP-like ligands (Fig.3.2e). Among the first were ATP, ADP, AMP, ACP (AMP-PCP, or phosphomethyl phosphonic acid

adenylate ester), ANP (AMP-PNP, or gamma-imino-ATP), and AGS (ATP-S_γ). The second group contained many inhibitors, such as imatinib (STI) and staurosporine (STU).

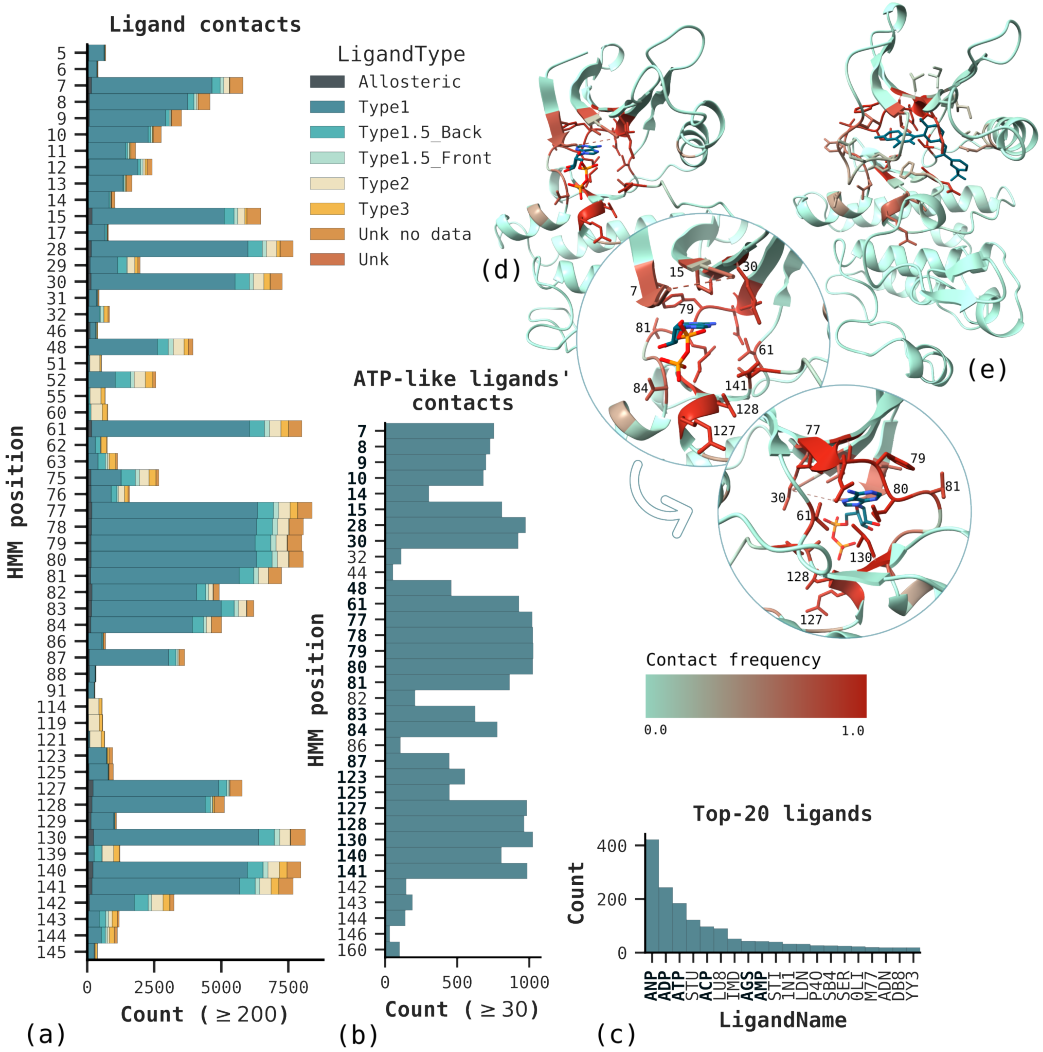

Figure 3.2: Ligand contacts. **(a)** The number of chains corresponding to an HMM position, grouped by the type of ligand contacted. "Unk no data" refers to entries outside KinCore, while "Unk" refers to a few non-annotated cases. We omitted structure-ligand pairs present in KinCore but absent in our data. **(b)** The number of chains per HMM position contacting ATP-like ligands. Positions with ≥250 counts (in bold) are part of the ATP binding pocket. **(c)** ABL1 structure (PDB 2HIW:A) bound to a Type-2 inhibitor in the DFG-out conformation. The color scale shows the number of counts. Counts were scaled to the 0–1 range to compare the two structures. **(d)** ABL1 structure (PDB 2G2I:A), bound to ADP, in the DFG-in conformation. The circles with denoted HMM positions zoom into the binding pocket. The structure in the lower circle is rotated by about 180°. The color scale depicts the frequency of contacts with ATP-like ligands. **(e)** The most frequently contacted ligands, with ATP-like ligands in bold.

Protein positions frequently contacting the ATP-like ligands were reviewed (Fig. 3.2b). This resulted in a binding pocket defined by 25 HMM positions (Table 3.2). Fig. 3.2d shows the ligand contact frequency of individual residues in the ADP-bound ABL1 structure. The overall, position-wise, ligand-binding frequency is shown in Fig. 3.2c, and seen to be similar. By mapping ligand types reported by KinCore, we verified that these same positions also contact a large group of Type-1 inhibitors that compete with ATP (3.2a), and often formed contacts with other inhibitor groups. On the other hand, HMM positions 51, 55, 60, 114, 119, and 121 (HMM numbering) mostly contacted DFG-out-specific Type-2 inhibitors. Src numbering is also provided in Table 3.2.

Table 3.2: Positions that contact ATP-like ligands.

HMM positions	Src positions	Structural region
7 8	276 277	B1
9 10	278 279	B1-B2
14 15	283 284	B2
28 30	296 298	B3
48	313	aC
61	326	aC-B4
77 78 79 80 81	341 342 343 344 345	Hinge
83 84 87	347 348 351	HD
123 125 127 128 130	389 391 393 394 396	CL
140 141	406 407	ALN

For the 25 pocket positions, we computed the 300 mutual C_β–C_β distances, distances to DFG-Phe (C_β and C_ζ atoms; 50 distances) and DFG-Asp (C_γ atom), as well as distances between the atom pairs (DFG-Phe C_ζ, DFG-Asp C_γ) and ($\beta 3$ Lys N_ζ, αC-helix Glu C_δ), for a total of 377 distance variables for future use.

3.2.3 Clustering and manually curating the DFG conformations

In view of classification, the kinase structures were clustered, using a distance metric based on the DFG-Asp and DFG-Phe positions. Structures were first superimposed, based on the C_α positions of the 30 most covered HMM positions: 100–115 (H_E), 125–132 (CL), 137–140 (CL-ALN), and 181–182 (H_F).

We filtered to 8927 PDB chains (94.5% of our dataset) where all these positions were present. Other structures were discarded. We then computed a pairwise distance defined by the rms deviation between the two structures, averaged over the DFG-Phe and DFG-Asp residues. Fig. 3.3b shows the distribution of rms deviations, both for the 30 C_α atoms and for the DFG-Asp/DFG-Phe pair. The C_α deviations were mostly small, below 2 Å, while the DFG deviations extended up to 25 Å. Agglomerative clustering with a 2.0 Å threshold yielded 161 clusters (Fig. 3.3a). Cluster sizes were far from uniform, with just a few dominant clusters. Clusters 41, 46, and 86 contained 84% of the PDB chains. There were 52 singleton clusters and 108 clusters with 5 elements or fewer.

Clusters were then assigned to a DFG class. The intuition behind the assignment is illustrated in Fig. 3.4 and Fig. 3.8; details are in Methods. We performed visual inspection, aligning up to 20 randomly-chosen cluster members to a "central", reference structure. Clusters were almost always characterized by a close similarity between entire activation loops. Thus, the DFG-Asp/DFG-Phe clustering metric was a good proxy for the overall loop conformation. None of the clusters contained more than one DFG conformation, except for a few cases with substitutions (e.g., chains 6DC0:A,B from the cluster 22, which were pseudokinases) and missing atoms in the DFG motif, due to disorder (e.g., chain 6RUU:C from cluster 86 was mislabeled by us as DFG-out). We then assigned cluster labels, corresponding to DFG-in, DFG-out, and DFG-other, in a semi-supervised manner, and transferred the consensus label to the entire cluster. There were 46 DFG-in clusters (7931 chains in all), 33 DFG-other clusters (108 chains), and 33 DFG-out clusters (857 chains). Most DFG-in chains were in clusters 5–49; the largest DFG-in clusters were 41 and 46. Most DFG-out clusters were in the ranges 58–86, 88–104, and 134–145; the largest DFG-out cluster was 86. Most DFG-other chains were in clusters 106–128. Some were scattered across smaller clusters, including some (66–68, 97–98) that were in the mostly DFG-out ranges above, and thus were structurally rather similar to DFG-out.

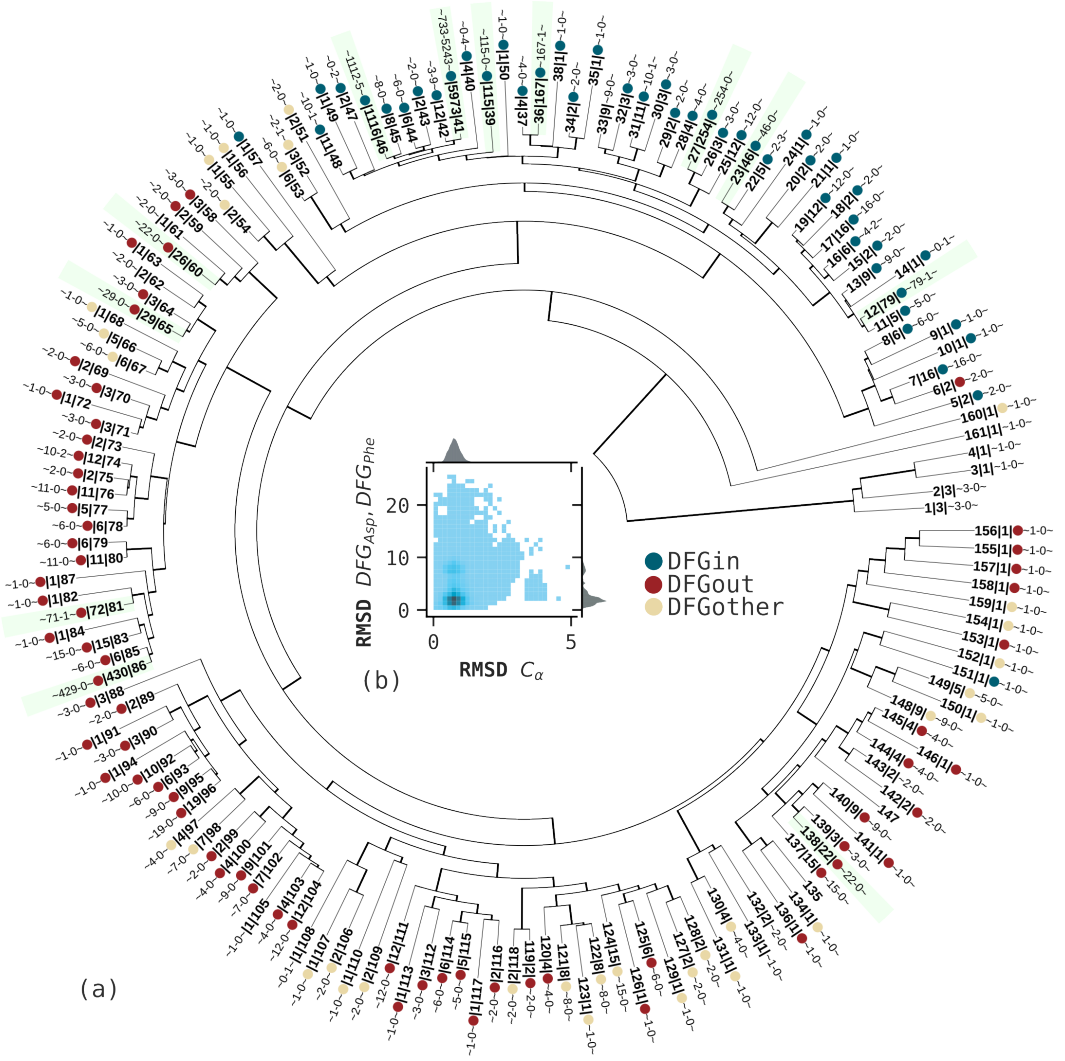

Figure 3.3: Clustering of PK domains. **(a)** Dendrogram representation, with leaves annotated, radially, by ID, size, DFG label (colored dot), number of inactive and active structures (the last two swap places in the [90,270]° range). DFG-in, DFG-other, DFG-out conformations are denoted by blue, ivory, and red dots. Several clusters have no DFG label due to missing atoms. Clusters with 20 or more domains are highlighted in light green (12 in total). **(b)** The joint rmsd histogram with 50 bins per variable. C_α rmsd peaks around 1 Å, while the DFG rmsd has a large peak in the [0, 3] Å range and a smaller one around 8 Å.

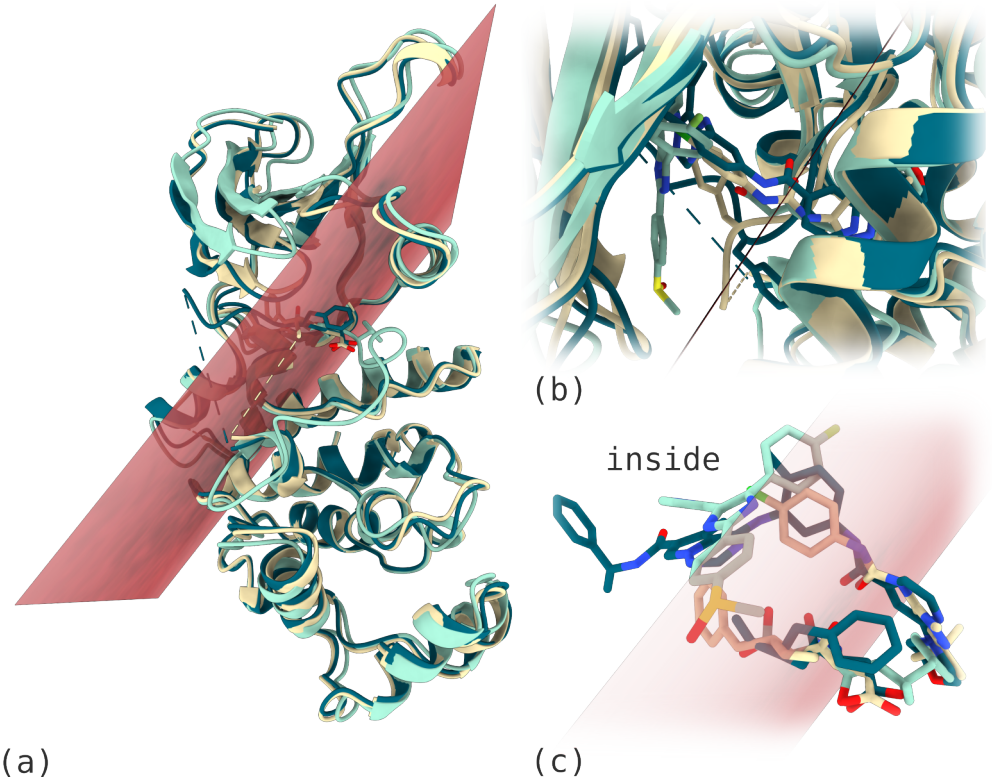

Figure 3.4: The intuition behind the DFG motif annotation. **(a)** A plane (in red) divides the space between the N-lobe and the C-lobe into "inside" and "outside" regions. The structures are the same as in Fig. 3.1b. **(b)** A view from above the plane going through the DFG motif's backbone. Except for Aurora, the plane puts the αC-helix "outside". Ligands are shown, using the same colors as the proteins. **(c)** A close-up of the DFG residues and ligands with the dividing plane, showing that ligands binding to DFG-out and DFG-other structures can extend "outside", while the DFG-in ligand (light blue) is "inside".

3.2.4 Relating clusters to existing resources

Our collection of labeled structures is in Supplementary Material. We compared structure labels to the existing resources ABC [51], KLIFS [28], KinCore [29, 30], and KinaMetrix [52]. All four recognized DFG-in and DFG-out categories. All but ABC recognized a third category: DFG-out-like (KLIFS), DFG-inter (KinCore), and ω_{CD} (KinaMetrix). These categories did not always agree with our DFG-other label (as discussed below). Table 3.3 shows that all resources covered most of our clusters. For instance, 145 clusters (105 of 109 non-singletons) contained KLIFS structures, while 135 (96 non-singletons) contained KinCore structures.

There were three type of discrepancies within the clusters. (1) Internal: a cluster contained chains with opposing labels from the same external resource; for example, Kincore considers certain chains of cluster 148 to be DFG-in (3BV2:A), DFG-out (3BV3:A), and DFG-other (4UMT:A). (2) External: two or more external resources proposed different labels for the same kinase chain. (3) Adverse: external resources disagreed with our own labels. Within the non-singleton clusters (Table 3.3), there were 31, 16, 10, and 14 internal conflicts for KLIFS, KinCore, ABC, and KinaMetrix, respectively, or 67 in all (62% of non-singleton clusters). KLIFS, KinCore, and KinaMetrix had 74, 72, and 57 externally conflicting clusters, or 75 in all (69% of non-singleton clusters). Finally, KLIFS, KinCore, and KinaMetrix disagreed with our labels in the case of 72, 53, and 41 clusters, respectively ("adverse" labels). Supplementary Table 3.6 lists all the conflicting clusters with examples of structures.

Table 3.3: Cluster coverage by external resources

Resource	Present	Non singleton	Internally conflicting	References
KLIFS	145	105	29	[26–28]
KinCore	135	96	16	[30]
ABC	117	91	10	[52]
KinaMetrix	108	82	13	[51]

92 clusters contained structures from all sources, while the number of clusters unique to KLIFS, KinCore, or KinaMetrix was 5, 2, and 4, respectively (Supp. Fig. 3.11a). Most of the shared clusters (53 out of 92) contained external conflicts (Supp. Fig. 3.11c), where a few of the external labels disagreed.

Internal conflicts were less common (15, 4, and 4 unique to KLIFS, KinCore, or KinaMetrix, respectively; 5 intersecting; Supp. Fig. 3.11b). As for clusters with adverse external annotations, there were 25, 15, and 4 such clusters unique to KLIFS, KinCore, or KinaMetrix, while 22 were common to all three sources. Notice that "internal" inconsistencies imply adverse annotations: if a resource had two labels in a cluster, at least one would conflict with our label.

Supp. Fig. 3.11c shows that 5 and 42 clusters had only internal or external conflicts, respectively. Only 14 clusters had only adverse annotations (91–93, 95, 100–104, 113, 128, 136, 140–141), which all concerned discrepancies between DFG-out and DFG-other labels. Most internal and external discrepancies were also of this type. On the other hand, all resources discriminated well between the DFG-in and DFG-out structures: e.g., there were only 12 DFG-in/DFG-out conflicts between KLIFS and KinCore within our clusters. We return to these findings in the Discussion.

KinCore used a clustering method that differed from ours, and was based on dihedral angles of the DFG motif and its preceding residue. KinCore's clusters covered 22 DFG-in, 20 DFG-out, and none of our DFG-other clusters. Rather, KinCore regarded all the 103 overlapping DFG-other entries as "Noise".

3.2.5 ML models to label kinase conformations automatically

Using the labelled dataset created above, we trained four binary classifiers: (1) `KinActive` (Active vs. Inactive), (2) `DFGin`, (3) `DFGout`, and (4) `DFGother`. Each model consisted in an ensemble of 100 trees, with maximum depths of 15, 7, 4, and 16, respectively. An example tree is shown in Supplementary Fig. 3.12. All the models used the same feature pool of structural variables (set 3 in Table 3.5) and went through the same training pipeline.

To train the active/inactive `KinActive` model, we used the curated active/inactive labels provided by McSkimming et al. [53], who used them to train a Random Forest model called `kinconform`. Our model quickly learned to discriminate between active and inactive conformations. The final model relied on 78 features involving 70 positions. It reproduced and slightly surpassed previously reported performance (see below). The model made errors for the chains 2NP8:A, 3NYX:A (false positive), and 6KZI:A (false negative),

all coming from cluster 41 (Fig. 3.5d).

We trained the `DFG` models using the dataset and labels obtained above, excluding cases with DFG-Asp substitutions and/or missing DFG-Phe. Each model used the same number of data instances, labeled by 0 (negative class) or 1 (positive class). The final, trained models relied on 65 (DFG-in), 108 (DFG-out), and 80 (DFG-other) features, involving 54, 73, and 44 unique sequence positions. Selected features for each model are listed in Supplementary Material.

Each classifier outputted a probability for its DFG conformation. The probabilities from the three DFG models then served as inputs to train a logistic regression (`LR`) model `DFGclassifier`, which predicted the final DFG label. Intuitively, `LR` learned to balance probability values so that their linear combination predicted the final label correctly. Note that `DFGclassifier` was cross-validated as a separate model, training and testing `DFGin`, `DFGout`, `DFGother`, and `LR` on the same cross validation folds.

Table 3.4: Model performance

Model	+/-*	$F_1^{CV_{10}}$	Features	$F_1^{CV_{10}}$**	$\text{Prec}^{CV_{10}}$**	$\text{Rec}^{CV_{10}}$**	Errors
`KinActive`	1522/1762	0.9983	78	0.9992	0.9989	0.9994	3
`DFGin`	7874/952	0.9996	65	0.9996	0.9991	1.0000	7
`DFGout`	847/7979	0.9917	108	0.9953	0.9918	0.9988	8
`DFGother`	105/8721	0.8494	80	0.8976	0.9200	0.8762	21
`DFGclassifier`	-	-	3	0.9975	0.9975	0.9975	17

* Positive and negative examples.
** Using the selected features.

The final `KinActive` and `DFGclassifier` models achieved outstanding performance (Table 3.4), with only 3 and 17 errors each during cross validation. `DFGin` reached the maximum possible recall rate, with no false negatives. `DFGout` had a single FN and seven FP errors. `DFGother` was less accurate, with 13 FN and 8 FP errors. Fig. 3.5b depicts the confusion matrices, showing that 25 out of 36 errors involved entries with DFG-other labels. `DFGclassifier` failed to resolve 16 of 31 misclassifications (when accounting for common chains) and made a single new one. In turn, 16 of 17 `DFGclassifier` errors corresponded to DFG-other entries, mostly from underrepresented clusters (e.g., 51-54, 66-68, 152-154, 160). In most of these cases, either there were two high-probability outputs (e.g., 3GGF:A,B, which consti-

tute cluster 51, had comparable DFG-in and DFG-other probabilities) or all probabilities were very small (e.g., 6TUA:A from cluster 67). We manually reviewed the entries mislabeled by `DFGclassifier` to check whether the mistakes were legitimate. This resulted in us spotting two annotation mistakes that the model rightfully corrected: 6TUA:A (DFG-in) and 5DE2:B (DFG-out). These had been incorrectly clustered, due to a substituted DFG motif and missing DFG-Phe atoms.

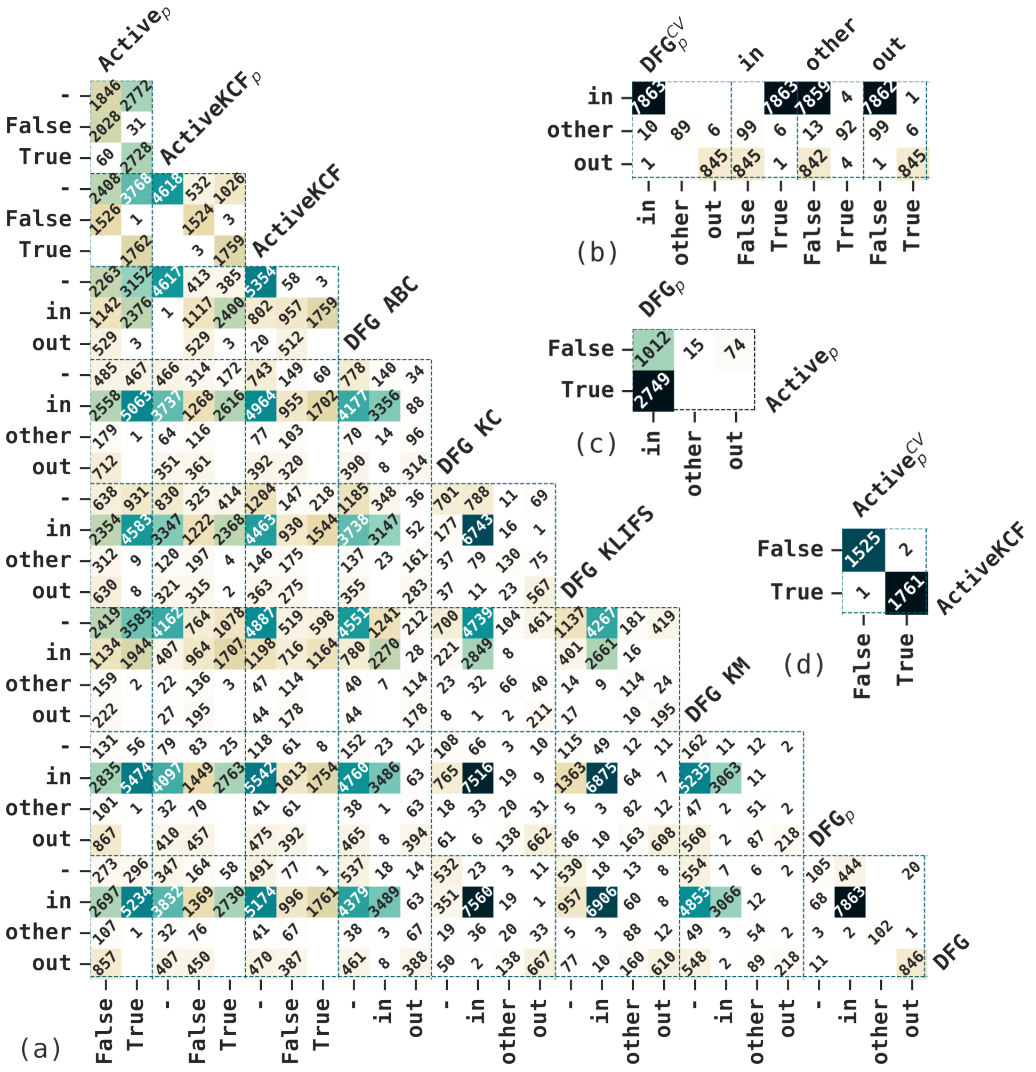

Figure 3.5: Label comparisons. **(a)** A pairwise comparison of PK labels, involving 2 types: (1) active (True/False), (2) DFG (in/other/out). For (1), we show labels predicted by `KinActive` and the `kinconform` model (ActiveKCF pred), and curated labels (ActiveKCF) from [53]. For (2) we include DFG labels from other resources, curated (DFG) and predicted (DFG$_p$) by `DFGclassifier`. Note that predicted labels were derived after refitting models on the full dataset. "-" indicates a missing label. **(b)** Confusion matrices for the `DFG` models. X-axis depicts curated labels while labels predicted during the CV by the `DFGclassifier` and its base classifiers are on the Y-axis **(c)** `DFGclassifier` and `KinActive` labels predicted for the AlphaFold2 models. **(d)** A confusion matrix for the `KinActive` model comparing predicted (Y-axis) and predicted (X-axis) labels.

We inferred labels for all structures using the models trained on all available data. Fig. 3.5a compares predicted and curated labels, and labels from other resources. The `KinActive` and `DFGclassifier` models retained one (3NYX:A) and three errors (3GGF:A,B and 4EH3:A). Although `KinActive` and `kinconform` predictions largely agreed, there were 91 differences, including more active structures (60) predicted by the earlier `kinconform` tool.

Inactive chains were partitioned into 2835 DFG-in, 101 DFG-other, and 867 DFG-out. All active chains were predicted as DFG-in except one (5UQ1:C, DFG-other, cluster 52). 99.6% of active PKs resided in DFG-in cluster 41, which indicates stable structural features for the activation loop in the active conformation. This cluster contained 5243 active and 733 inactive structures, demonstrating that features other than the DFG motif discriminate between active and inactive states. The feature selection results confirm this (see below).

Comparing the active labels to the KinCore clusters (Supp. Fig. 3.13), we observed that structures predicted to be active were split primarily between KinCore's BLAminus (4496) and ABAminus (527) clusters. On the other hand, `KinActive` predicted 224 and 229 BLAminus and ABAminus structures as inactive (see the Discussion below).

3.2.6 Interpreting the ML models

To interpret the obtained models, we ranked the features selected ("learned") by each model. Such post-selection ranking may attribute zero importance to some features since the feature compositions during and after selection are typically different. Therefore, after consulting the distributions of feature importance, we narrowed our focus to the top 10 most important features per model (Fig. 3.6a). Supplementary Table 3.7 gives a list of residues and features that contributed strongly to the decision making.

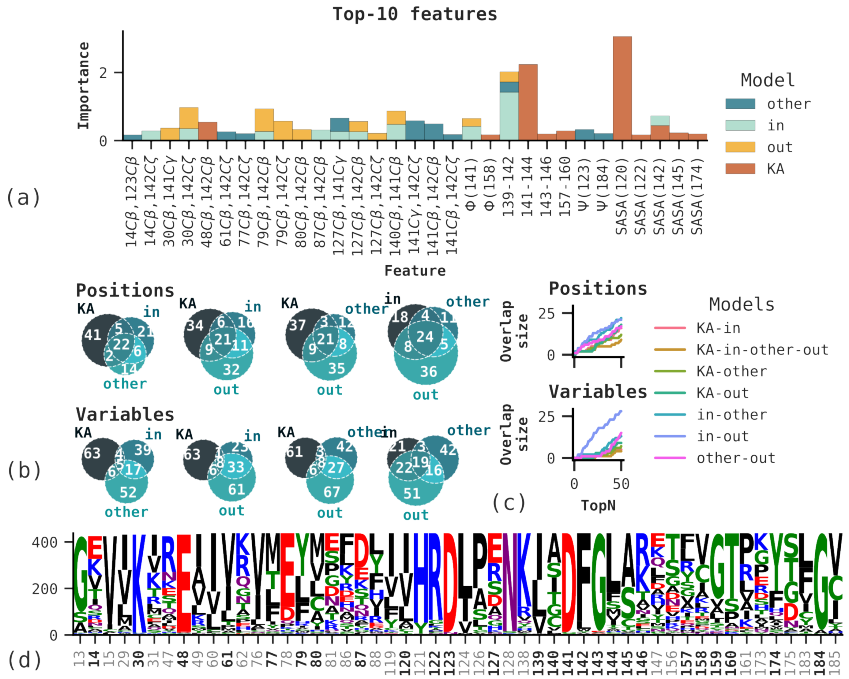

Figure 3.6: Top selected features. **(a)** Top-10 features ranked by importance for each model. The X axis shows importance summed over the models. **(b)** Venn diagrams depicting overlaps between sets of selected features (above) and corresponding residues (below). **(c)** Number of overlapping positions (above) and features (below), versus the size N of the top-N set considered. See Fig. 3.14 for a full PK domain logo. **(d)** Sequence logo built from UniProt sequences from our database using positions depicted in (a), highlighted in bold, with their nearest neighbors.

Among the top-10 sets, 17 out of 22 features selected by the `DFG*` models were distance variables, 12 of which were between the binding pocket residues and DFG-Asp/DFG-Phe. For instance, $D_{30C_\beta,142C_\zeta}$ (where 30 is the β3 Lys and 142 is the DFG-Phe), $D_{79C_\beta,142C_\beta}$, and $D_{127C_\beta,142C_\beta}$ were crucial for both the `DFGin` and `DFGout` models. The distances between hinge (77, 79, 80) and DFG-Phe residues mainly contributed to the `DFGout` predictions. Curiously, the $D_{140C_\beta,141C_\beta}$ distance had a high importance for `DFGin` and `DFGout`, but not for the `DFGother` model, which relied more on DFG-Asp/DFG-Phe distances ($D_{141C_\gamma,142C_\zeta}$, $D_{141C_\beta,142C_\beta}$, and $D_{141C_\beta,142C_\zeta}$). The `KinActive` top-10 set contained a single distance variable, ranked 3rd: $D_{48C_\beta,142C_\zeta}$, where 48 is the αC-helix Glu.

Two neighboring pseudo-dihedral angles, $Pd_{139-142}$ and $Pd_{141-144}$, were ranked 1st and 2nd for `DFGin` and `KinActive`. The $Pd_{141-142}$ angle was also crucial to `DFGout` and `DFGother`, but to a lesser extent. Finally, the solvent-accessible surface area at position 120 ranked higher than the HRD motif variables, except for the $Pd_{141-144}$ distance, which exceeded all other features. In summary, the most important features corresponded to well-known residues and structural regions conserved across PK domains: β3 Lys, αC-helix Glu, HRD and DFG motifs (Fig. 3.6b).

As Fig. 3.6a demonstrates, the top-10 sets from the four binary models contained overlapping features. Six out of eight overlaps concerned the `DFGin` and `DFGout` models. This effect persisted when considering other top-N sets (Fig. 3.6d), with `DFGin` and `DFGout` models sharing most variables. In contrast, all other model combinations had 2–4 overlapping variables up to N=20. However, when we considered the positions involved, the overlap increased linearly with N, from 10 up to 20 at N=50. Complete selected sets retained the observed patterns, such as larger overlaps between the `DFG*` models compared to the `KinActive-DFG*` pair (Fig. 3.6c). Indeed, `KinActive` had only 9, 11, and 14 features common to the `DFGin`, `DFGother`, and `DFGother` models. In contrast, `DFG*` models shared 60 (35%) and 41 (39%) features and positions.

Having analyzed feature compositions, we considered how their values influenced the model outputs. For this, we selected the following features from the top-10 sets: (1) for `KinActive`, we picked $SASA_{120}$, $SASA_{142}$, $D_{48C_\beta,142C_\beta}$, and $Pd_{141-144}$, (2) for `DFGin` and `DFGout` we picked the same set of overlapping features $D_{30C_\beta,142C_\zeta}$, $D_{79C_\beta,142C_\zeta}$, $D_{140C_\beta,141C_\beta}$, and $Pd_{139-142}$. We constructed a

balanced background dataset for each model, randomly sampling 300 positive and negative instances. We analyzed the connection between SHAP contributions, feature values, and actual classes (Supplementary Fig. 3.15). We also explored feature pairwise interactions within the SHAP space, picking the top feature with the highest interaction magnitude per variable selected for analysis (Supplementary Fig. 3.16).

We could establish practical decision boundaries by examining the distribution of feature values. For instance, non-zero values of $SASA_{120}$ and $SASA_{142}$, and $D_{48C_\beta,142C_\beta} > 8$ Å usually indicated inactive conformations (Supp. Fig. 3.16a and Supp. Fig. 3.15a). In turn, $D_{30C_\beta,142C_\zeta} < 11$ Å, $D_{140C_\beta,141C_\beta} > 5$ Å and $D_{79C_\beta,142C_\beta} < 15$ Å make the DFG-out conformation highly probable (Supp. Fig. 3.16b). Finally, $D_{30C_\beta,142C_\zeta} > 11$ Å and $D_{140C_\beta,141C_\beta} < 5$ Å are clear DFG-in indicators (Supp. Fig. 3.16c).

Association with the SHAP contributions further strengthened these intuitions. Thus, for `KinActive`, $SASA_{120} = 0$ and $SASA_{142} = 0$ have relatively high SHAP values, driving the predictions towards the active class (Supp. Fig. 3.15a). Decreasing $D_{140C_\beta,141C_\beta}$ and increasing $D_{30C_\beta,142C_\zeta}$ had opposite effects on the SHAP contributions, contributing to the final DFG-in and DFG-out labels (Supp. Fig. 3.15b,c). Similar arguments concern other analyzed features, where, in most cases, there was a tangible clustering of classes within the (feature, SHAP(feature)) space.

3.2.7 Classifying a large dataset of AlphaFold2 kinase structures

As an application, we considered a large set of kinase structures predicted by AlphaFold2 [33]. All but 5% of these structures were manually annotated with DFG labels. The dataset is available from an online public repository (see Section 3.4.13) [54]. To illustrate our ML models, we proceeded to relabel all the kinase structures from the Swissprot portion of the AlphaFold2 database. We applied the same filtering criteria for domain discovery as above. This yielded 3960 initial sequences with domain hits, 3850 of which we managed to download from the AlphaFold2 database. The data covered 21 protein families from at least 282 organisms. The most abundant organism was *A. thaliana*, with 571 Ser/Thr kinases (Supplementary Fig. 3.18). The other most abundant

organisms were human, mouse, rat, *C. elegans*, and rice.

We extracted the kinase domains from each sequence and structure and calculated structural variables defined above (set 3 in Table 3.5). We then proceeded to predict the DFG and active/inactive labels using our `DFGclassifier` and `KinActive` models. There were 1101 and 2749 inactive/active predictions, respectively (≈40% inactive), while 3761, 15, and 74 chains were predicted to be DFG-in, DFG-other, and DFG-out, respectively (roughly 250:1:5). In our own database of experimental kinase structures from the PDB, the proportions were 3934/5531 inactive/active (≈42% inactive) and 77:1:8. Thus, Alphafold2 models had the right inactive proportion, but were heavily skewed towards DFG-in, compared to the experimental structures. Alphafold2 label combinations were internally consistent, with no DFG-other or DFG-out structures predicted to be active (Fig. 3.5c). Curiously, the balance between Ser/Thr and Tyr PKs was reversed for the DFG-out-predicted entries, with 49 Tyr kinases, many of which were orthologs (10 of OTK, all coming from various *Drosophila* species), NTRK1 (4), NTRK2 (2), NTRK3 (6), DDR1 (4).

3.3 Discussion

This work provided an up-to-date assessment of kinase conformational space, as defined by its activation loop and DFG motif. An extensive collection of catalytic domain sequences and structures was built. Its creation was transparent and open, facilitating future applications or extensions, such as fitting ML models or analyzing the structural kinome. We clustered and classified the structures in a supervised way, relying mainly on the geometry of the DFG motif, which turned out to be a good proxy for the overall activation loop geometry. We then proceeded to build several machine learning models to do the same classification automatically, using ensembles of decision trees. The models started from a pool of several thousand structural variables (features), covering a large part of the catalytic domain. The models were carefully trained, and the most important variables for classification were identified ("learned"). They included variables involving the DFG and HRD motifs, the β_3 Lys, and the α_C Glu, which are kinase hallmarks.

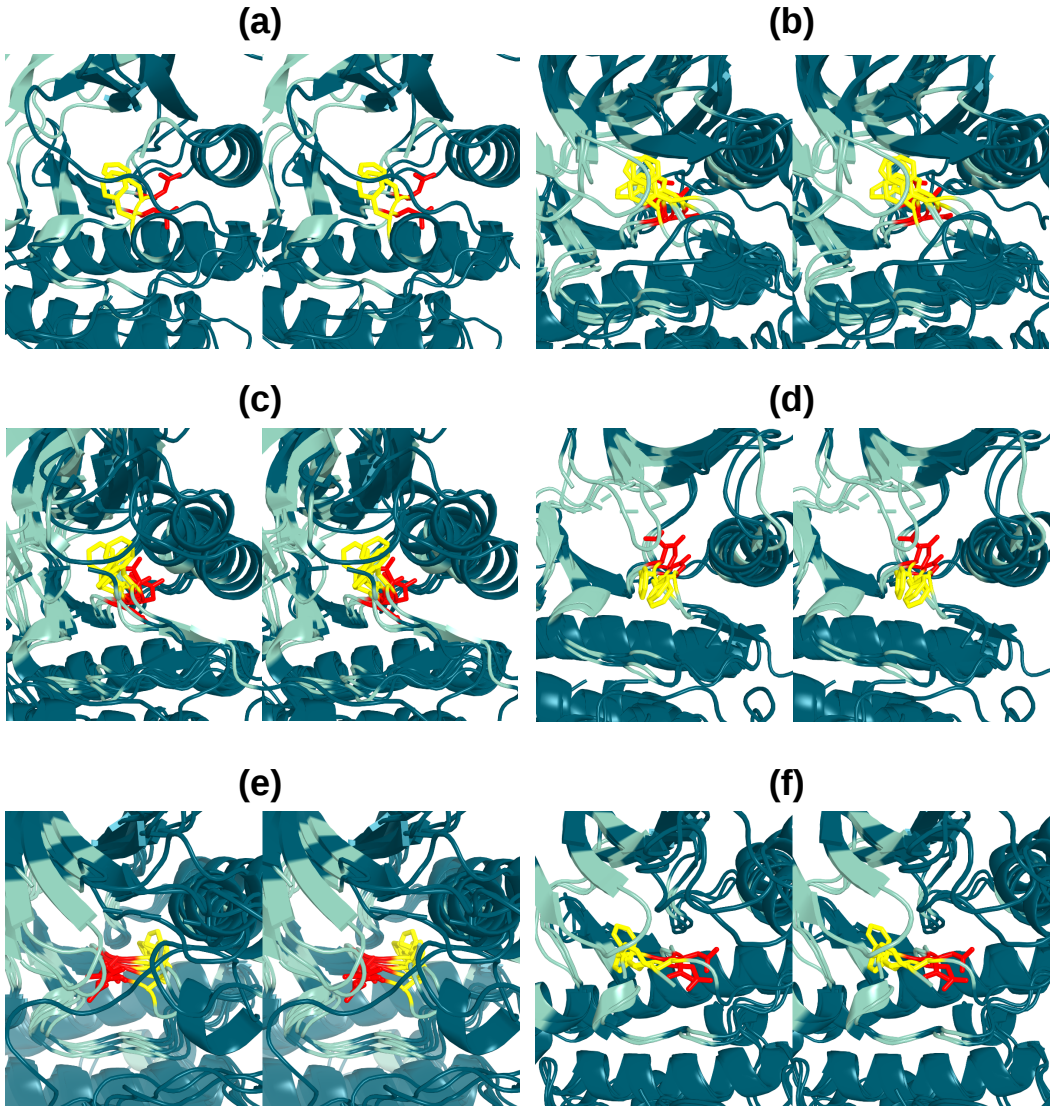

Figure 3.7: Examples of clusters and conformations in stereo. DFG-Asp/Phe in red/yellow. **(a)** 4EL9:A, 5UWD:A from clusters 90, 91. **(b)** Representative structures from DFG-out clusters 92–96: 3DK6:A, 3DK7:A, 2WOB:A, 6YKY:A, 4OBP:B. **(c)** Representative structures with a unique conformation where DFG-Phe is pushed out and up: 2G2H:A, 2JIT:B, 3MN3:A, 4B9D:A, 4MXX:B. **(d)** Structures intermediate between DFG-out and -other: 2G2F:B, 5B2L:A, 6YG4:A. **(e)** Cluster 41 with structures labeled by KinCore as ABAminus, BLAminus, BLAplus, BLBminus, and BLBplus. DFG-Asp and DFG-Phe occupy similar subpockets but have different dihedral angles: 3GU6:A, 3PXK:A, 3PXZ:A, 5FEE, 6TPE:A, 7M5Z:B. **(f)** DFG-out clusters 65, 81, and 86, labeled by KinCore as BBAminus, but with DFG-Asp and DFG-Phe occupying different subpockets: 3G6G:A, 4R5Y:A, 5HG8: A.

The top variables could be analyzed and interpeted in some detail. Consider first the solvent accessibility (SASA) of residues 120 and 142. Supp. Fig. 3.16a and Fig. 3.17a show that zero accessibility is an active state signature. Position 120, before the HRD motif, is covered by the activation loop (AL) in most active structures. Conversely, the AL is often disordered or misplaced in inactive states. Active structures require the DFG-in orientation so that DFG-Phe (142) is buried between the αC-helix and the β_4 strand. In contrast, inactive structures often have the DFG-Phe displaced, making it accessible to solvent. Furthermore, the sequence logo in Fig. 3.6d shows that positions 120 and 142 are always nonpolar, and presumably stabilize the active conformation via hydrophobic effects.

Another variable is the $D_{48C_\beta,142C_\beta}$ distance, which was strongly associated with the inactive state, since it captures the DFG-Phe displacement away from the α_C-helix (Supp. Fig. 3.15a and Fig. 3.17a). Combining it with $Pd_{141-144}$ yielded clear decision boundaries (Supp. Fig. 3.16a). The joint distributions of $SASA_{120}$ with $SASA_{142}$ and $Pd_{142-145}$ further suggested that an interplay between residues close to the HRD and DFG motifs determines active/inactive states.

The `DFG` models relied mostly on distance variables and the DFG backbone orientation, captured by the 139–142 pseudo-dihedral angle. Among the top-10 selected variables were the distances $D_{30C_\beta,141C_\gamma}$ (`DFGout`), $D_{30C_\beta,142C_\zeta}$, $D_{79C_\beta,142C_\beta}$ and $D_{140C_\beta,141C_\beta}$ (`DFGin` and `DFGout`). These variables are also readily interpretable. For instance, decreasing (increasing) the distance between DFG-Phe and the binding pocket residues, such as β_3 Lys30 while increasing (decreasing) the $D_{140C_\beta,141C_\beta}$ distance makes a domain more likely to occupy the DFG-out (DFG-in) conformation (Supp. Fig. 3.15). Interestingly, the top features often interacted with each other. Such interacting pairs had the highest SHAP interaction values, and their joint distributions imposed better decision boundaries than single-value distributions.

While KLIFS relies entirely on the DFG motif, the KinCore resource uses two distances, $D_1 = D_{52C_\alpha,142C_\zeta}$ and $D_2 = D_{30C_\alpha,142C_\zeta}$, to classify DFG conformations via two threshold values [29]. This approach does not consider explicitly the DFG-Asp orientation, causing it to mislabel some DFG-other conformations. It is not applicable to cases where β_3 Lys (position 570) or α_C Glu+4 (115) are missing, nor to non-canonical DFG-Phe substitutions missing

the C_ζ atom. Our work shows that accurate DFG motif labeling (in contrast to active/inactive) does not require variables anchored to the α_C helix. It can be derived by combining the mutual DFG-Asp/DFG-Phe orientation with their positioning relative to the binding pocket residues.

Most DFG label discrepancies involved the DFG-other conformation. This category lacks a precise definition, instead serving as an out-group with flexible boundaries. As a result, methods often conflict with each other and mislabel DFG-other or mix it wih DFG-in or DFG-out for similar structures. Although these discrepancies mainly concerned rare conformations, they may still cause accuracy losses in large-scale applications that take the in/out/other labels as established input. For example, consider the closely related clusters 90 and 91, represented by chains 4EL9:A and 5UWD:A (Fig. 3.7a). In line with the paper accompanying 4EL9 [55], we labeled these structures as DFG-out. In contrast, KinCore labeled 4EL9:A as DFG-out and 5UWD:A (where DFG-Asp is outside) as DFG-other, while KLIFS labeled both structures DFG-other. We found clusters 92–96 and 100–104 to also encompass DFG-out entries mislabeled as DFG-other (Fig. 3.7b,c). Mislabeling the latter group was especially puzzling, as it contradicts the existing literature, including a pivotal paper on the Src-like inactive conformation of Abl1 [56–59].

Literature sources may lead to conflicting cluster information as well. To illustrate, Levinson et al. labeled 2G2F:B (referred to as molecule E) as "intermediate" [58], while Pemovska et al. labeled 4WA9:A from the same cluster 120 as DFG-out [60]. Finally, for a group of chains from clusters 121–123 labeled as DFG-other in our work (Fig. 3.7d), KLIFS assigns DFG-other labels, KinCore labels 121–123 as DFG-out, while the structures' authors mostly agree on the DFG-out state (5BDL [61], 5Z1E [62], 6QFR [62], 6YG4 [63]). More generally, it is challenging to provide a unifying framework for the DFG conformational landscape of all kinases, causing clashes between methods that put all non-standard conformations into a single category.

In conclusion, our structural clusters may serve as reference points for newly solved kinase structures. In the future, it could be beneficial to further adapt these results to drug design efforts, by relating the obtained clusters to the active site subpockets. Our annotations could be complemented or refined by dihedral angle based clustering, as in the KinCore approach. We expect the ML models will be useful for future, large-scale applications to the structural

kinome, similar to the present Alphafold application.

3.4 Materials and Methods

3.4.1 Building a collection of PK domains

To obtain PK catalytic domains, we queried the SIFTS database [50], which collects PDB entries [64] along with their corresponding Uniprot sequences [65, 66]. We queried SIFTS with the PK profile PF00069 from the Pfam database [44], using the `PyHMMer` software [67]. During this process, we retained mappings between each sequence and the numbering of nodes in the corresponding Pfam HMM, for future reference. The HMM can be thought of as a sequence profile or alignment. Correspondance between HMM position numbers and residue numbers in human Src are given below. We retained hits that had (1) a catalytic domain size above 150 residues, (2) a similarity bit score to the query above 50, and (3) a coverage of both the sequence and the HMM nodes above 70%. We then fetched PDB structures associated with these hits. We mapped UniProt and PDB sequence numberings via pairwise sequence alignments performed with `mafft` [68]. These mappings allowed us to transfer the HMM node mappings from UniProt to each PDB sequence, which identified the domain boundaries within the extracted structures. We used the annotated and transferred domain boundaries to extract domains from UniProt sequences and PDB structures. Thus, each extracted domain comprised: (1) a UniProt sequence, called the "canonical" sequence, (2) a PDB sequence (which can be slightly different), (3) a PDB structure, and (4) associated metadata, such as IDs and coverage information. We retained only domains with less than 10% of the PDB sequence mutated relative to the canonical one.

3.4.2 Clustering catalytic domain structures

To facilitate the annotation of structures (as active, inactive, DFG-in, and so on), the collected structures were grouped into into clusters, based on the DFG motif conformation. The distance between two structures was defined by the rms deviation between them, after superposition, calculated using atoms of the DFG-Asp and DFG-Phe residues. Pairwise superposition of each domain pair relied on the rms deviations between C_α atoms of the 30 HMM profile's

most frequently mapped residues. This procedure produced a matrix of distances between all domain pairs, which was clustered using the single linkage agglomerative clustering provided by the `scipy` library. Manually inspecting the results, we arrived at a distance threshold of 2.0 Å to form the final clusters.

The superposition protocol encountered certain special cases. In case a single position was missing from one PDB structure (eg, due to disorder), we relied on the 29 other positions for the superposition. In the case of a mismatch between DFG-Asp or DFG-Phe atoms within a domain pair (eg, a few atoms missing from a PDB model), our protocol used a common set of atoms, and backbone atoms at the minimum.

3.4.3 Semi-supervised DFG conformation labeling

To classify, or label each cluster, we performed the following steps:

1. Randomly selected up to 20 structures from the cluster.
2. Identified the sample center: the structure with the lowest mean distance to the other sample members.
3. Superposed sample structures onto the central one.
4. Visualized the superposed structures.
5. Verified that they all had the same DFG conformation.
6. Visually defined a plane dividing the ATP binding pocket into “in” and “out” regions: in practice, the plane was defined roughly by the activation loop backbone (see Fig. 3.4).
7. Annotated DFG-Asp and DFG-Phe side chains as “in” or “out” based on their positioning in either region.
8. Annotated the domain conformation based on the DFG-Asp and DFG-Phe labels, following the rules given below.
9. Transferred the derived label to all the cluster members.

From the DFG-Asp, Phe labels (step 7), we derived the DFG label (step 8) as follows:

- DFG-Asp “in” and DFG-Phe “out”: we assigned the DFG-in label.
- DFG-Asp “out” and DFG-Phe “in”, we assigned the DFG-out label.

- We assigned the DFG-other label in cases where both DFG-Asp and DFG-Phe were "in" or "out", including the cases where the separating plane would divide the Asp and Phe residues themselves.

To resolve difficult cases during steps 7-8: (1) we compared to "typical" DFG-in and DFG-out conformations (those from clusters 41 and 86, respectively; see Results); (2) we compared to already-annotated clusters nearby; (3) we considered pocket residue positions (specified in the Results), and (4) we consulted the literature if a structure had an accompanying paper specified in the PDB. In total, our manual annotations went through five revisions during the study.

3.4.4 Extracting sequence and structural variables

For each extracted domain sequence and structure, descriptor variables, or "features" were computed. These were then used as input for the ML models developed below. They included sequence variables, structural variables, and ligand variables. They could be subdivided into four subsets: (1) canonical sequence variables (see below; 792 in all); (2) PDB sequence variables (792 in all); (3) PDB structural variables, such as torsion angles and interatomic distances (1692 variables), and (4) ligand variables, such as the shortest ligand-residue distance (792 variables). Variables were anchored to the PK profile's HMM nodes, so that values from different kinases and structures could be treated as different instances of the same variable. The computation took less than an hour for all structures when using 20 cores for set (3) and a single core for sets (1), (2), and (4).

Table 3.5 lists the calculated variables. For sequence variables (sets 1 and 2), we used a numerical representation of each AA type, provided by the ProtFP resource [69, 70]. These representations were based on a large set of physical-chemical properties taken from the AAindex database [71], reduced through a principal component analysis by the authors. Although variable sets (1) and (2) turned out to be redundant for our ML models, we provide them for completeness and potential use in future applications. Set (3) comprised dihedral and pseudo-dihedral angles [72], solvent-accessible surface areas of individual residues, and residue–residue distances. The "ligand" variables of set (4) included the list of residues contacting the ligand, the number of contacts,

and the minimum distance of each residue to the ligand. A 5.5 Å distance threshold was used to define residue–ligand contacts.

Table 3.5: Calculated variables

Set	Variable	Positions	Description
1-2	ProtFP	1-264	3 ProtFP PCA components per position
3	ψ	1-264	Psi dihedral angle
3	ϕ	2-264	Phi dihedral angle
3	χ_1	1-264	Chi1 dihedral angle
3	$Pd(i)$	1-262	Pseudo-dihedral angle, residues i, i+1, i+2, i+3
3	$SASA$	1-264	Solvent-accessible surface area per position
3	$D(C_\beta\text{-}C_{\beta,\gamma,\zeta})$	Pocket residues	Residue-residue distance
4	Ligand	1-264	The name of the ligand contacted
4	Contacts	1-264	Residue-ligand atomic contact number
4	Distances	1-264	Minimum distance to the closest ligand

3.4.5 ML models for binary classification

Having labeled our structures and extracted sequence and structural variables, we proceeded to develop several ML models, designed and trained to reproduce the classification above. All models (except one) were binary classifiers that distinguished two classes. The first model distinguished active/inactive structures. The others distinguished structures either within a particular DFG class or not, namely "DFG-in" vs. "anything else", "DFG-out" vs. "anything else", "DFG-other" vs. "anything else". These binary models were based on the XGBoost tool (eXtreme Gradient Boosting) [73].

XGBoost is a supervised machine-learning algorithm that gradually builds up an ensemble of decision trees. Each tree is a collection of internal and terminal (leaf) nodes. Internal nodes contain learned thresholds that guide each input instance (a particular structure) down through the tree to a leaf node. Each leaf node computes the probability p of a particular label (say, active), and outputs the log-odds probability ratio, $log(\frac{p}{1-p})$. An overall probability is obtained by summing up the outputs of individual trees and transforming the result via the logistic function. For instance, if $f(x)$ is the sum of the output ratios, $P(\text{class} = 1) = 1/(1 + e^{-f(x)})$ is the probability of the positive class.

During training, starting from the current prediction, XGBoost adds one tree at a time to gradually drive the prediction closer to the actual value, which is known. Each tree is produced, or "grown" by recursively splitting each existing node, say P, into two children, say L and R. To split P, one considers the deviations, or "residuals" between the predictions of P and the correct labels of all structures (which are known during training). One searches for a feature and a threshold value that splits the input structures into two subsets (L and R) such that labeling errors are reduced as much as possible. XGBoost decides whether to add a new node to a tree depending on various pre-set criteria, such as a maximum tree depth, minimum leaf output, and minimum error reduction (detailed below). After a tree is grown, XGBoost finalizes its structure by recursively eliminating (or pruning) leaf nodes that yield suboptimal splits of residuals.

In more detail, the objective function to be minimized measures the difference between the current predictions and the actual labels, and contains also regularization terms that penalize a tree's complexity. Let y_i be the known label of a structure i, $\hat{y}_i$ the label predicted by the current ensemble, T the number of leaves across all indivudal trees, and w_j the prediction of each leaf j. The objective function can be written:

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t)}) + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 + \gamma T, \tag{3.1}$$

where λ and γ are pre-set constants and the last two terms favor tree pruning. Predictions at an iteration t are derived from those computed at a step $t-1$:

$$\hat{y}^{(t)} = y^{(t-1)} + w, \tag{3.2}$$

where w is the output value of a node at which a training structure arrives. Using this additive training paradigm, given a solution at step $t-1$, XGBoost attempts to find an optimal value for w that would minimize the objective. For this, it approximates the loss function using a second-order Taylor expansion, which transforms the objective into a quadratic equation,

$$obj^{(t)} \approx \sum_{j=1}^{T} (G_j * w_j + \frac{1}{2}(H_j + \lambda) * w_j^2) + \gamma T, \tag{3.3}$$

where G_j and H_j are the gradients and the Hessians of a loss function at step $t-1$ summed over all training instances ending up at a leaf node j. Solving

$$\frac{\delta}{\delta w_j} obj^{(t)} = 0 \tag{3.4}$$

leads to an optimal leaf output

$$w_j^* = \frac{-G_i}{H_i + \lambda} \tag{3.5}$$

The objective becomes

$$obj^{(t)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} \tag{3.6}$$

Similar to in other decision-tree-based ensembles [74], individual trees are trained greedily, considering one split at a time. To construct a good split, XGBoost assigns scores S_j to tree nodes j, each equal to

$$S_j = \frac{1}{2} \frac{G_j^2}{H_j + \lambda} = -G_j \dot{w}_j^* \tag{3.7}$$

(elements of the sum in $obj^{(t)}$). This leads to

$$\text{Gain} = \frac{1}{2} \left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right) - \gamma T \tag{3.8}$$

Finally, we used a binary-cross entropy for a loss function, $-y \log(p) - (1 - y) \log(1 - p)$, where y is a true class and p is the predicted probability of a positive class. This leads to

$$S_j = \frac{(\sum_{i \in j} (y_i - p_i))^2}{\sum_{i \in j} (p_i(1 - p_i)) + \lambda} \tag{3.9}$$

Intuitively, XGBoost trees minimize the residuals and learn to split the training data in ways that cluster them.

3.4.6 Feature selection, or "learning"

Feature selection was done using an iterative procedure known as the "all-relevant Boruta algorithm" [47, 75]. Given a current set of features, each iteration included several steps. (1) The first was to duplicate the current features and randomly scramble the copies so that values initially associated with one kinase become associated with another, randomly chosen. (2) The augmented feature set is then used to train the ML model. In practice, the features are fed to a series of decision trees, which make predictions about the classification of each kinase. (3) Evaluate each feature's importance numerically, with a metric defined below. (4) Compare the importance values of the original features and the permuted copies. Designate as "hits" the features, among the original ones, whose importance is higher than 95% of the permuted ones. (5) Use a statistical test to accept or reject features based on a history of the accumulated hits. The idea is to accept variables that have a consistently higher importance than random noise. (6) Remove both accepted and rejected features from the current pool. (7) Stop if the pool of remaining features is empty or the maximum number of iterations has been reached (100 in our case). Otherwise, return to the first step.

3.4.7 Feature importance measure

The feature importance is an estimate of an individual feature's impact on the algorithm's predictions. We used SHAP (SHapley Additive exPlanations) to define feature importance [76]. SHAP explanations use a combinatorial approach that compares the model's predictions with and without a particular feature, relative to a baseline prediction, e.g., the average prediction made by the model for all data instances in the training set. For each feature and structure, this approach defines its contribution to the output (for the XGBoost method, this is the log odds ratio, $log(\frac{p}{1-p})$).

In more detail, for a dataset with M input structures and N features, SHAP outputs an $M \times N$ matrix Φ of estimated contributions to the outputs. Summing Φ over structures (or instances), $\sum_i \Phi_{i,j}$, yields an N-sized feature importance vector. In addition, for the `TreeExplainer` tool provides an estimate of pairwise contributions. This may be seen as splitting an estimated SHAP value into diagonal and off-diagonal (interaction) terms [77]. The pro-

cess outputs an $M \times N \times N$ table F. Summing over structures, $\sum_i F_{i,j,k}$ yields an $N \times N$ matrix of interactions accumulated across data instances, which can be queried to find the feature pairs with the highest interactions.

3.4.8 Optimizing the ML models

The XGBoost models were trained to identify ("learn") features that classify effectively. They also contained parameters that were "non-trainable" and optimized manually. These included tree pruning rules and maximum tree depth; see Supplementary Table 3.8. Model optimization comprised the following stages:

1. Short optimization of non-trainable parameters on the full dataset of structures.
2. Cross-validation of the performance estimate.
3. Feature selection with eBoruta. From here on, only the selected features are used.
4. A second, longer, parameter optimization.
5. Cross-validation of the performance estimate.
6. Training the model on the full dataset.
7. Predicting classes for all available structures.
8. Ranking the selected features.

During steps 1 and 4, we searched for optimal values of parameters controlling the algorithm's execution. Each time, before and after feature selection, models were cross-validated using binary metrics defined below. After the algorithm's training on a full dataset, features were ranked for later comparison and interpretion. More details for each step are given in later sections.

3.4.9 Performance measures for binary classification

We used three performance measures: the F_1 score, model precision, and recall. Let TP and FN be the number of correctly predicted positive instances (true positives) and incorrectly predicted negative instances (false negatives). Instances are kinase structures to be classified. Let FP be the number of

incorrectly predicted positives (false positives). Precision is defined as

$$\text{prec} = \frac{TP}{TP + FP} \tag{3.10}$$

Recall, or true positive rate (TPR), or sensitivity, is defined as

$$\text{rec} = \frac{TP}{TP + FN} \tag{3.11}$$

The F_1-score combines these two measurements via their geometric mean:

$$F_1 = 2\frac{\text{prc} * \text{rec}}{\text{prc} + \text{rec}} = \frac{2TP}{2TP + FP + FN} \tag{3.12}$$

to provide a composite performance measurement typically used for class-imbalanced datasets, like those encountered here. The dataset was split into N non-overlapping parts based on PDB identifiers during cross-validation (CV) and N train/test cycles were run. In each cycle, we trained on all dataset instances except one, which was used for prediction. Accumulating predictions led to a fully-predicted dataset that was used to compute the performance metrics above. N was set to 5 or 10.

3.4.10 Optimizing non-trainable parameters

We used the `optuna` library [78] to optimize non-trainable (or "hyper-") parameters. `Optuna` solved the optimization problem $\underset{\theta}{argmax} F_1^{CV_5}$, seeking a parameter set θ that maximized the cross-validated F_1 score. Supplementary Table 3.8 lists the non-trainable parameters subject to optimization. We used 10 optimization rounds for calibrating parameters before feature selection, and 100 rounds after feature selection.

The number of trees for each classifier was determined using an early stopping technique. Namely, each time the algorithm was trained, input data was split into training and evaluation subsets ("folds") as during the CV. We used the evaluation fold to monitor the loss function after adding each tree, and stopped the training if the last 20 trees did not improve the loss or when reaching the maximum number of 100 trees. An example tree is shown in Supplementary Fig. 3.12.

3.4.11 Logistic regression classifier

We also built a final logistic regression (LR) meta-classifier of the DFG conformations. LR used only three input variables (hence, no feature selection was necessary): the class probabilities output by the binary, XGBoost classifiers. It was trained to predict three possible DFG states. As this is a multiclass objective, the binary metrics above required an adjustment. We used the so-called "micro" averaging strategy, where TP, FP, TN, and FN were combined into a single "confusion" matrix. 100 optimization steps were used to find the optimal hyperparameters (see Supplementary Table 3.8).

3.4.12 Comparing to other resources

We compared our data to existing resources using PDB identifiers. As a first step, we updated some obsolete PDB IDs found in some resources using old-to-new mappings obtained from the `wwpdb` FTP server (`https://files.wwpdb.org/pub/pdb/data/status/obsolete.dat`). For sequences in our own database, we already had PDB-UniProt relationships, confirmed by pairwise alignment and sequence-to-sequence matching (see above). For the external resources, we used the mappings provided by the resource authors where possible, or by SIFTS in their absence. We fetched the corresponding metadata for each UniProt ID, including organism and protein family. We manually reviewed the resulting family annotations to distinguish protein kinases, non-protein kinases, and non-kinase proteins. For numerical comparisons below, we excluded confirmed non-kinase and non-protein kinase entries.

3.4.13 Software and data availability

Supplementary Table 3.9 lists all used software and resources. All data-related operations, from fetching sequences and structures to extracting domains, computing variables, and pairwise structure superpositions, were performed with our feature extraction library `Xtractor`. We made available an exact protocol to build and navigate the PK data collection via a separate repository `KinActive`. We used a reimplemented Boruta algorithm (`eBoruta`) for feature selection.

The source code related to this study is available within `KinActive`, written

in Python v3.10, based on `Xtractor` v0.1.1 (https://github.com/edikedik/lXtractor) and `eBoruta` v0.1 (https://github.com/edikedik/eBoruta), and available from the zenodo repository [54]. The tool is also available on GitHub (https://github.com/edikedik/kinactive). Its documentation, including a tutorial covering the creation of the structural kinome collection, is hosted on https://kinactive.readthedocs.io/en/latest. The trained models and their parameters are also included.

3.5 Supplementary Material

Additional data are provided as Supplementary Material. Supplementary Figures and Tables were listed above. Additional data files are:
kinase_labels.tsv gives the full list of 9471 kinase domains considered here, with their manual annotation and predicted annotation. UniProt and PDB ID's are provided, as well as our own ID's, referred to as lXtractor ID's.
DFGin_features.tsv, DFGother_features.tsv, DFGout_features.tsv, KinActive_features.tsv provide the features selected by each ML model, along with their importance score.

3.6 Supplementary Information

3.6.1 Supplementary figures

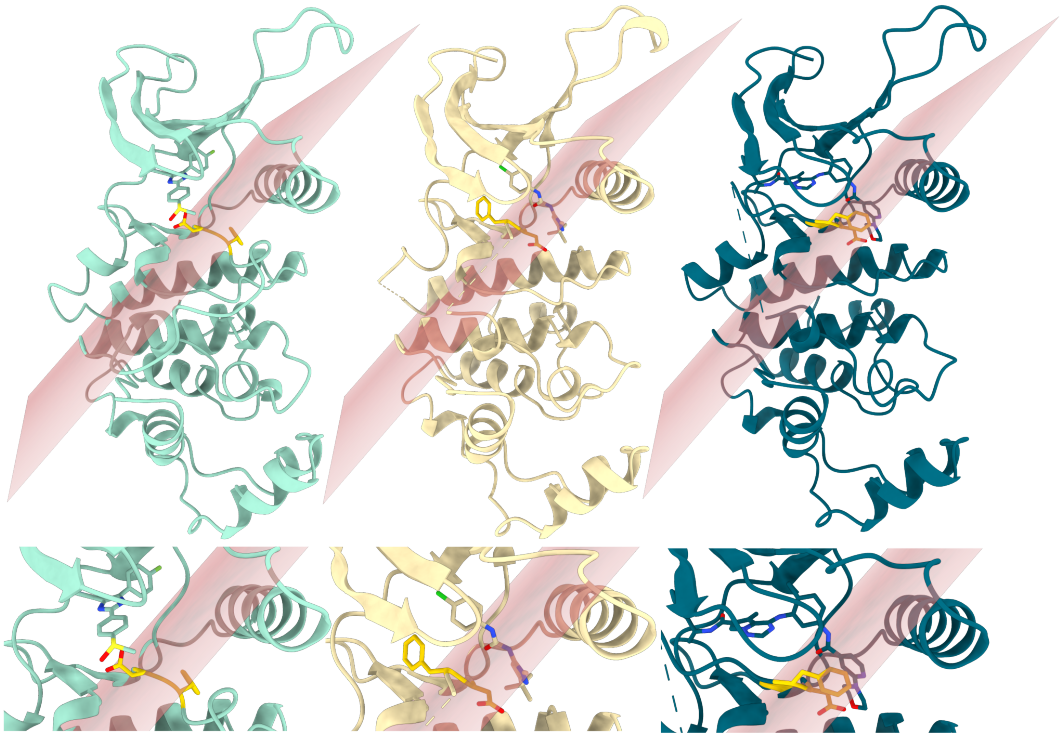

Figure 3.8: A complement to Fig. 3.4, where superposed structures are separated into subfigures; DFG-Asp and DFG-Phe are in yellow. **(a)** A structure in DFG-in orientation, where DFG-Asp is "inside" and DFG-Phe is "outside" (1A9U:A). **(b)** A domain in DFG-out conformation, where DFG-Asp and DFG-Phe swap places with respect to the plane (1KV1:A). **(c)** An example of the DFG-other orientation, where DFG-Asp and DFG-Phe are both "outside" (3BV2:A).

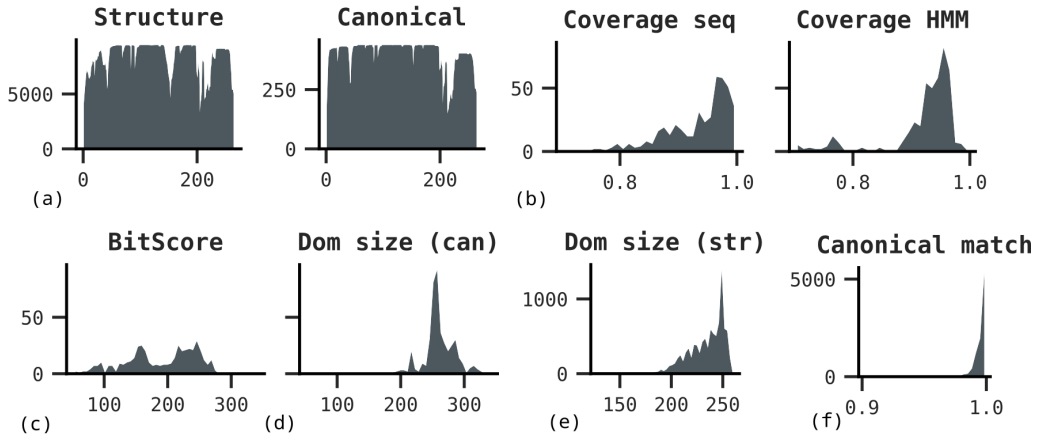

Figure 3.9: Sequence statistics. Y-axis is the total number of counts for all charts. **(a)** Coverage of the PK profile by structure and canonical sequences. **(b)** Sequence and node coverage. **(c)** BitScore distribution for the domains extracted from canonical sequences. **(d)** and **(e)** Domain size distribution in canonical and structure sequences. **(f)** Match percentage between canonical and structure sequences.

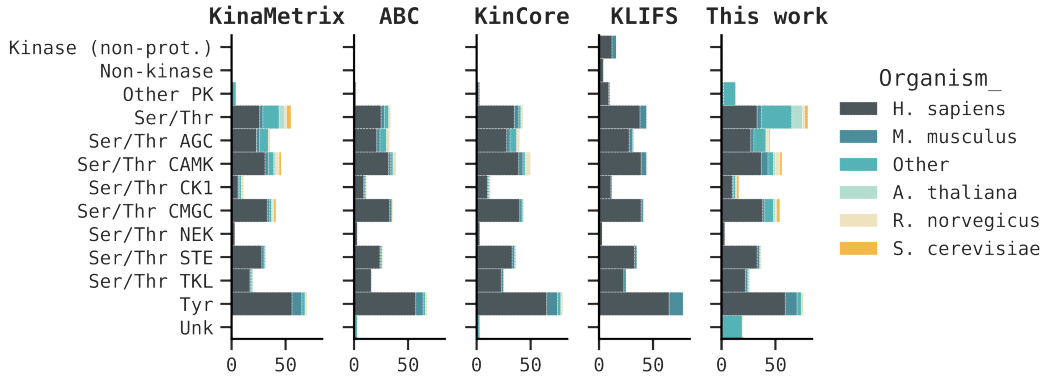

Figure 3.10: Kinases grouped by family (left), organism (right), and data resource (top).

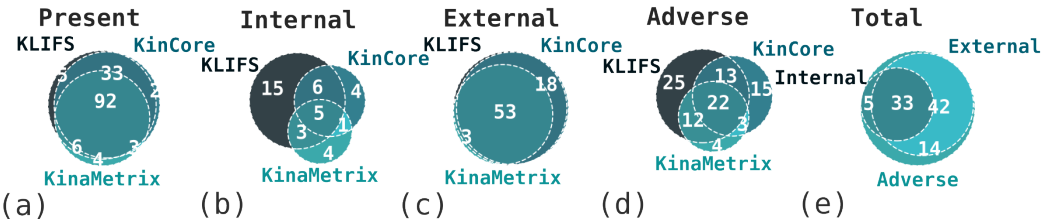

Figure 3.11: Venn diagrams representing connections between different cluster categories. From the left: (a) all clusters w here the external resource is "present" if a cluster contains at least one structure from that resource, (b- d) clusters associated with different DFG labeling conflicts (see the main text for details), (e) connection b etween the conflict types.

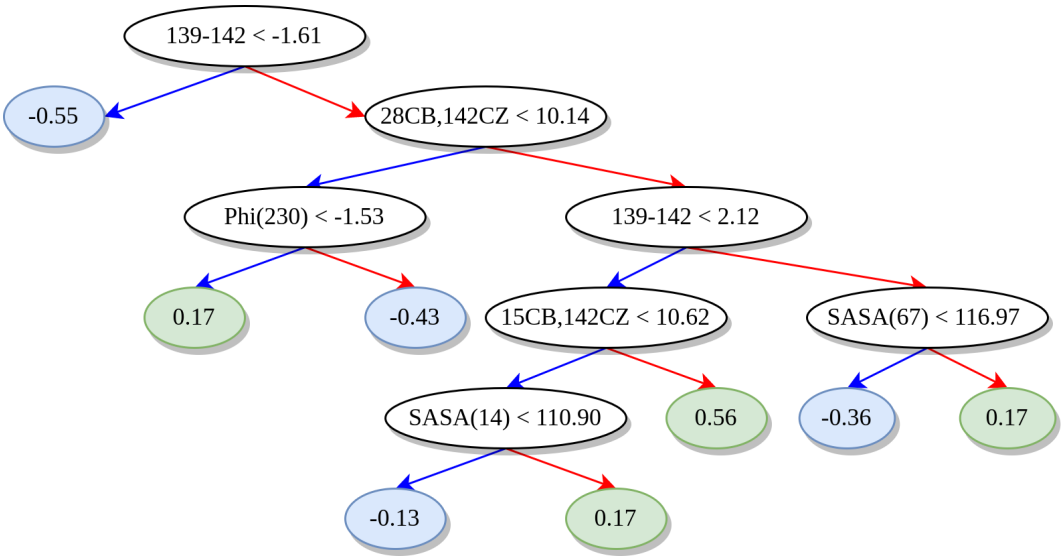

Figure 3.12: One of 100 trees from the DFG-in model. Nodes correspond to a feature: either a pseudo-dihedral spanning 4 residues (°), a residue SASA (Å^2), a backbone dihedral (°), or an atom–atom distance (Å). Red/blue arrows indicate true/false. Leaves are labeled with the decision score S (Eq. 3.9), which varies from -1 to 1 (strong decision).

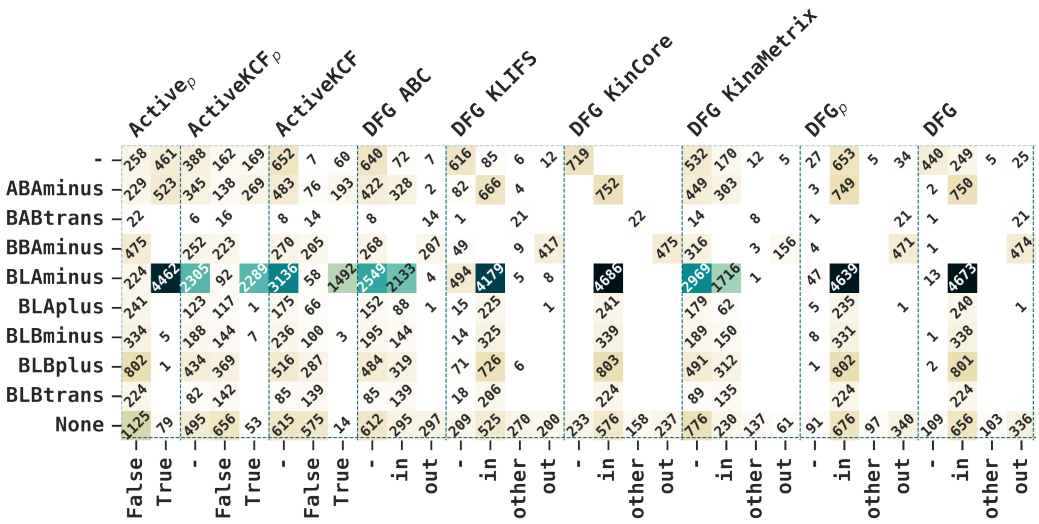

Figure 3.13: Comparison of the DFG and active/inactive labels with the KinCore clusters.

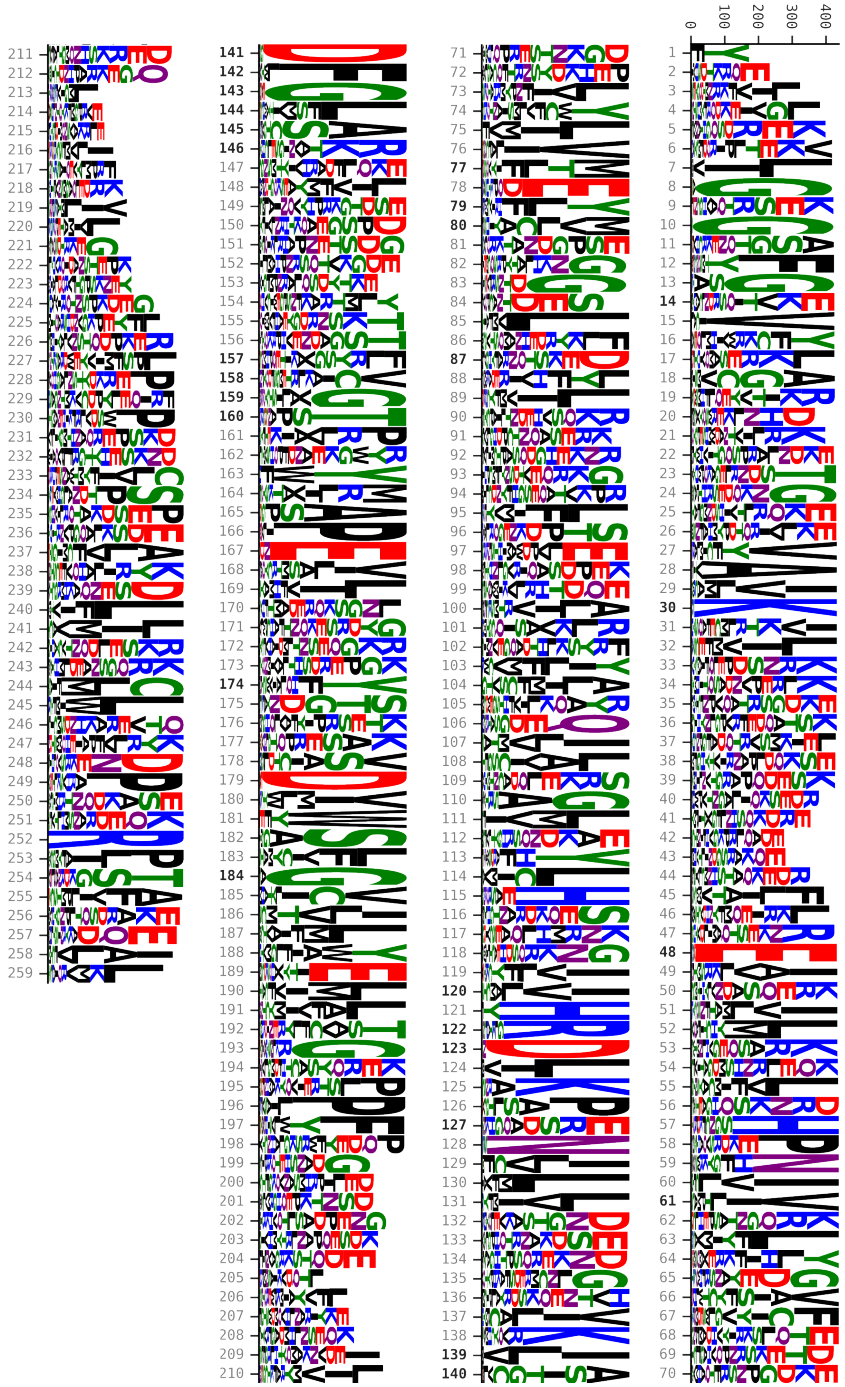

Figure 3.14: A PK domain sequence logo, complementing Fig. 3.6d. Y-axis depicts total counts. X-axis shows PF00069 reference positions, with selected top-10 selected positions highlighted in bold.

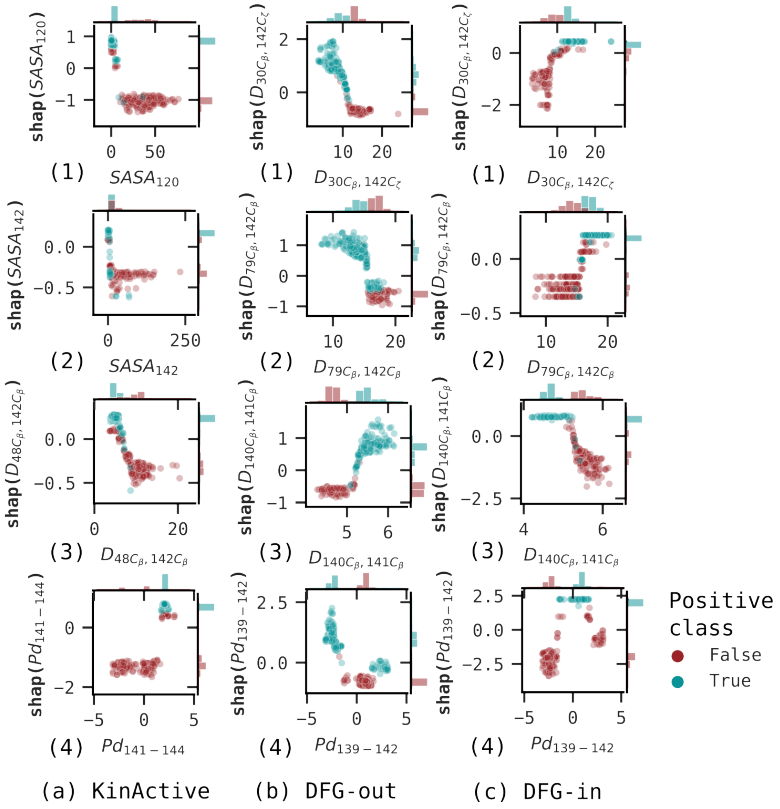

Figure 3.15: The distributions of (feature, shap(feature)) values for selected variables.

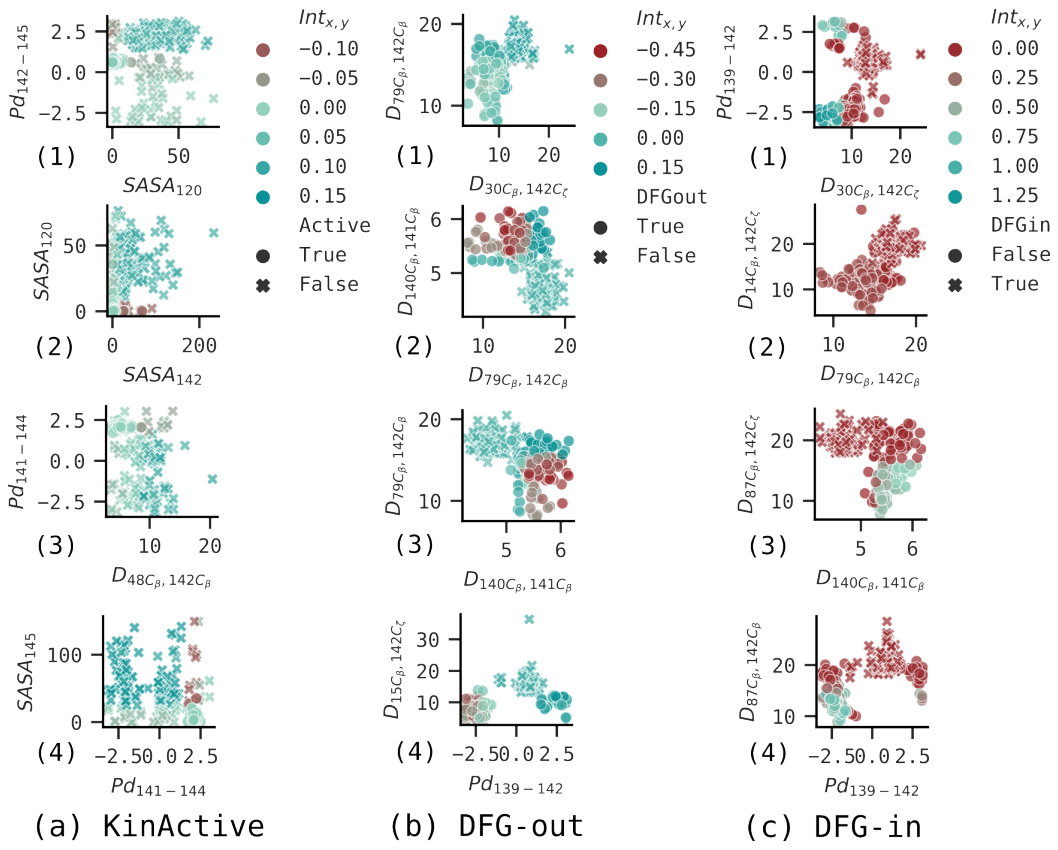

Figure 3.16: Correlations between selected top features for three of the ML models. Shown are the joint distributions of feature pairs, as histograms, colored by the magnitude of the pair's SHAP interaction. Positive/negative SHAP values push the model prediction towards the true/false class.

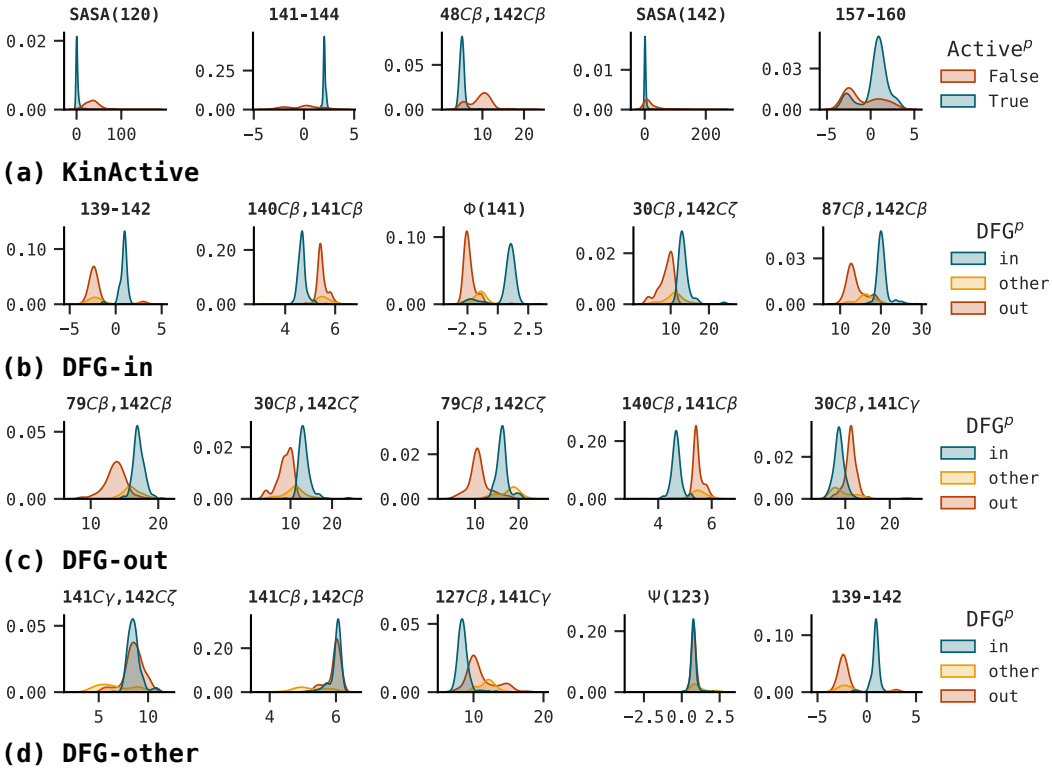

Figure 3.17: Value distributions of top-5 selected features per model, depicted as kernel density estimates. The variables are sorted in decreasing importance. Up to 300 domains were sampled for each label (e.g., 300 active and inactive domains for the KinActive model). The plots demonstrate that, for the most part, selected features allow discerning decision boundaries that separate different conformational states. They also show that DFG-other is typically intermediate to DFG-in and DFG-out, often overlapping both.

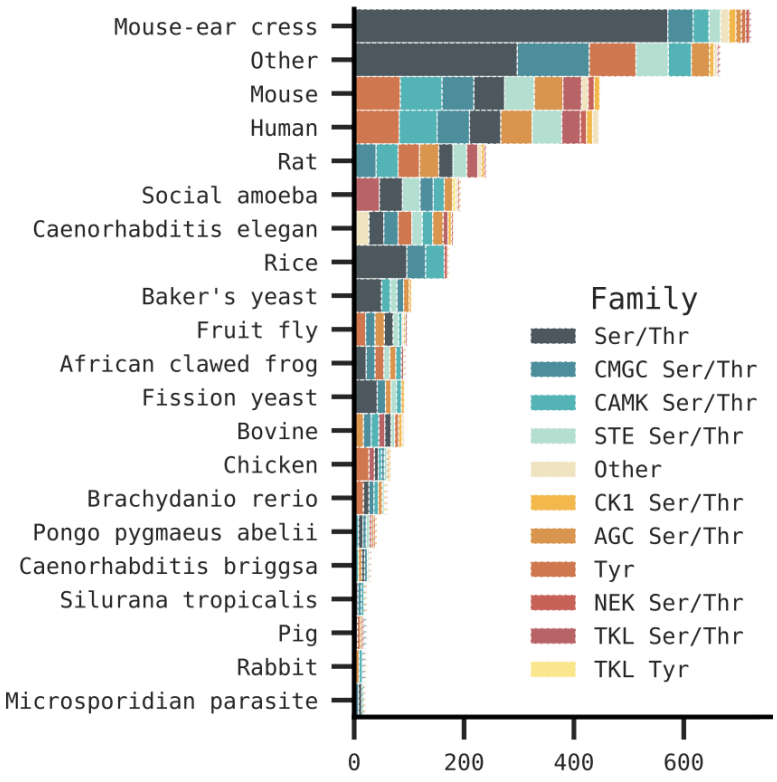

Figure 3.18: SwissProt Alphafold2 sequences grouped by organism and protein family. Non-PK families are labeled as "Other".

3.6.2 Supplementary Tables

Table 3.6: Conflicting clusters where (1) sources contain internal conflicts or (2) sources conflict with each other or (3) sources conflict with the curated label (DFG). The table lists up to two structures and chains per structure per label. We excluded the ABC resource as it doesn't recognize the DFGother conformation.

ID	DFG	Source	in	other	out
6	out	KLIFS	4RED:A,B		
		KinCore		4RED:A,B	
		KinaMetrix	4RED:A		
7	in	KLIFS	1VZO:A	3G51:A;4U3Y:A (10)	
		KinCore	3G51:A;6G77:A,B	1VZO:A;4U3Y:A (9)	
		KinaMetrix	1VZO:A;4U3Y:A		
8	in	KLIFS		1I44:A;3EKN:A (5)	
		KinCore	1I44:A;3EKN:A (6)		
		KinaMetrix		1I44:A;3EKN:A	
9	in	KLIFS		3OXI:A	
		KinCore	3OXI:A		
		KinaMetrix	3OXI:A		
10	in	KLIFS		4TYE:A	
		KinCore	4TYE:A		
11	in	KLIFS	1UA2:A,C;6S75:B	1UA2:B	
		KinCore	1UA2:A,B;6S75:B		
15	in	KLIFS		3GOP:A	
		KinCore	3GOP:A		
		KinaMetrix		3GOP:A	
16	in	KLIFS		2GHM:A;3TG1:A (6)	
		KinCore	2GHM:A;3TG1:A (6)		
		KinaMetrix	2GHM:A;3TG1:A		
17	in	KLIFS	3EKK:A;4LUD:A (13)	4F9W:A	
		KinCore	3EKK:A;4F9W:A (16)		
		KinaMetrix	3EKK:A;4F9W:A (5)	4O2P:A	
18	in	KLIFS		6BXI:A,B	
		KinCore	6BXI:A,B		
20	in	KLIFS		3VS1:A,B	
		KinCore	3VS1:A,B		
		KinaMetrix	3VS1:A		
21	in	KLIFS		5YT3:D	
		KinCore	5YT3:D		
23	in	KLIFS	1JQH:A,B;3BZ3:A (36)		
		KinCore	1JQH:A,B;2JKK:A (46)		
		KinaMetrix	1JQH:A;2JKK:A (8)	3I81:A	
25	in	KLIFS	2HAK:B,D;2QNJ:A,B	2HAK:A,C	
		KinaMetrix	2QNJ:A;3FE3:A	2HAK:A	
27	in	KLIFS	1B38:A;1B39:A (211)	6Q4F:A	
		KinCore	1B38:A;1B39:A (155)		
		KinaMetrix	1B38:A;1B39:A (216)		
28	in	KLIFS		3KMW:A;3REP:A	

Continued on the next page

Table 3.6: Conflicting clusters.

ID	DFG	Source	in	other	out
		KinCore	2HWP:A	3KMW:A;3REP:A	
		KinaMetrix	3KMW:A;3REP:A		
29	in	KLIFS		3CKW:A	
		KinCore	4D5H:A		
		KinaMetrix	4D5H:A	3CKW:A	
32	in	KLIFS	4DFY:A,E	3VUT:A	
		KinaMetrix	4DFY:A		
41	in	KLIFS	1A9U:A;1APM:E (5203)	3QUP:A;5K7I:B (6)	2WTK:B,E;6CCF:A,B
		KinCore	1A06:A;1A9U:A (5762)	3C4Y:B;4FGB:A	
		KinaMetrix	1A06:A;1A9U:A (2266)	2WTK:B;4OLI:A	
46	in	KLIFS	1AD5:A,B;1AGW:A,B (1016)	3OCS:A;5YT3:A (8)	
		KinCore	1AD5:A,B;1AGW:A,B (1097)	5YT3:A	7PUE:A
		KinaMetrix	1AD5:A;1AGW:A (397)		
47	in	KLIFS		4C58:A;4C59:A	
		KinCore	4C58:A;4C59:A		
48	in	KLIFS	3UIU:A,B;4FEQ:A (7)	3W18:B;4FG8:A	
		KinCore	3UIU:A,B;4FEQ:A (6)	4FG9:A,B	
		KinaMetrix	3UIU:A;4FEQ:A	4FG8:A	
51	other	KLIFS		3GGF:A,B	
		KinCore	3GGF:A,B		
		KinaMetrix	3GGF:A		
52	other	KLIFS		3DJ5:A;3DJ6:A	
		KinCore	3DJ5:A;3DJ6:A		
		KinaMetrix	3DJ5:A;3DJ6:A		
53	other	KLIFS	4UZD:A,B	2J4Z:B;4UYN:A	
		KinCore	2J4Z:B;4UYN:A (5)		
		KinaMetrix		3UC3:A;4UYN:A	
54	other	KLIFS		6BX6:A	2JAV:A
		KinaMetrix		2JAV:A	
57	in	KLIFS		5B2M:A	
		KinCore	5B2M:A		
58	out	KLIFS		4EBV:A	4EBW:A;4I4F:A
		KinCore		4EBW:A	4EBV:A;4I4F:A
		KinaMetrix		4EBV:A;4EBW:A	4I4F:A
60	out	KLIFS			1IRK:A;3LW0:A,B (26)
		KinCore			1IRK:A;3LW0:A,B (26)
		KinaMetrix		1IRK:A;3LW0:A (11)	
64	out	KLIFS		7S25:A	
		KinCore			7S25:A,B
65	out	KLIFS		2RF9:A	2RF9:B;3NAX:A (24)
		KinCore	4QQC:A		2RF9:A,B;3NAX:A (28)
		KinaMetrix		2RF9:A	3NAX:A;3QC4:A (7)
66	other	KLIFS		4NUS:A;5D9K:A,B	2XNO:A
		KinCore			4NUS:A;5D9K:A,B
		KinaMetrix		4NUS:A;5D9K:A	
67	other	KLIFS	6TUA:A		2XK3:A;2XK4:A (5)
		KinCore	6TUA:A		2XK3:A;2XK4:A (5)
		KinaMetrix		4A4X:A	

Continued on the next page

Table 3.6: Conflicting clusters.

ID	DFG	Source	in	other	out
68	other	KinCore			3D7T:B
		KinaMetrix		3D7T:B	
73	out	KLIFS			2AC3:A
		KinCore		1LUF:A	2AC3:A
		KinaMetrix			1LUF:A;2AC3:A
76	out	KLIFS		4YZ9:C	3GCP:A;3HUC:A (9)
		KinCore		3GCP:A;3HUC:A (9)	4YZ9:C
		KinaMetrix		2H34:A;3GCP:A (5)	
77	out	KLIFS		1GZN:A	1GZK:A;1GZO:A (4)
		KinCore			1GZK:A;1GZN:A (5)
		KinaMetrix			1GZK:A;1GZN:A
78	out	KLIFS		4YZ9:A,B	3ZZW:B;6Y23:A,B
		KinCore		6Y23:A,B	3ZZW:B;4YZ9:A,B
		KinaMetrix			4YZ9:A
86	out	KLIFS		4EH6:A;6RUU:C	1BI7:A;1BI8:A,C (380)
		KinCore	2EWA:A	3TWJ:B,D	1FPU:A,B;1G3N:A,E (412)
		KinaMetrix		2EWA:A	1FPU:A;1G3N:A (144)
88	out	KLIFS	2R5T:A;3HDM:A		
		KinCore		2R5T:A;3HDM:A	
89	out	KinaMetrix		3UC4:A	3UDB:A
90	out	KLIFS		3UBD:A;4EL9:A	
		KinCore			3UBD:A;4EL9:A
91	out	KLIFS		5UWD:A	
		KinCore		5UWD:A	
92	out	KLIFS		3DK6:A,B;4YC8:A,B (7)	
		KinCore		3DK6:A,B;4YC8:A,B (10)	
		KinaMetrix		3DK6:A;4YC8:A	
93	out	KLIFS		3DK3:A,B;3DK7:A (6)	
		KinCore		3DK3:A,B;3DK7:A (6)	
		KinaMetrix		3DK3:A;3DK7:A	
94	out	KLIFS		2WQB:A	
		KinCore			2WQB:A
		KinaMetrix		2WQB:A	
95	out	KLIFS		3DK7:B;4FZF:B (6)	
		KinCore		4FZF:B;4NZW:B (5)	
		KinaMetrix		4FZF:B;4I92:A (5)	
96	out	KLIFS		5CEK:A;5CEM:A	4OBO:B;4OBP:B (16)
		KinCore		6BRJ:A	4OBO:B;4OBP:B (18)
98	other	KLIFS		2WTV:A,B;3H10:D	3H10:A,B
		KinCore		2WTV:A,B;3H10:A,B	
		KinaMetrix		2WTV:A;3H10:A	
99	out	KLIFS		3FME:A	4RLO:B
		KinaMetrix		3FME:A	
100	out	KLIFS		1BYG:A;3OCT:A	
		KinCore		1BYG:A;3OCT:A	
		KinaMetrix		1BYG:A	
101	out	KLIFS		2CLQ:A,B;3SXR:A,B (8)	
		KinCore		2CLQ:A,B;3SXR:A,B (8)	

Continued on the next page

Table 3.6: Conflicting clusters.

ID	DFG	Source	in	other	out
		KinaMetrix		2CLQ:A;3MN3:A (5)	
102	out	KLIFS		2JIT:B;2JIU:B (7)	
		KinCore		2JIT:B;2JIU:B (7)	
103	out	KLIFS		4APC:A,B;4B9D:A,B	
		KinCore		4APC:A,B;4B9D:A,B	
		KinaMetrix		4APC:A;4B9D:A	
104	out	KLIFS		1FVR:A,B;2G2F:A (10)	
		KinCore		1FVR:A,B;2G2F:A (12)	
		KinaMetrix		1FVR:A;2G2F:A (5)	
110	other	KLIFS		5B2K:A	
		KinCore			5B2K:A
111	out	KLIFS		1M52:A,B;1OPK:A (12)	
		KinCore			1M52:A,B;1OPK:A (12)
		KinaMetrix		1M52:A;1OPK:A (6)	
112	out	KLIFS		3ZZW:A;4GT4:B	
		KinCore		3ZZW:A;4GT4:B	
		KinaMetrix			3ZZW:A
113	out	KinaMetrix		3P86:A	
114	out	KLIFS		3SOA:A;6S75:A	
		KinCore		3SOA:A	6S75:A;7M0K:A,B
		KinaMetrix		3SOA:A	
115	out	KLIFS		7M0L:A,B	3A60:B
		KinCore			7M0L:A,B
116	out	KLIFS		5J7S:A;5Y8U:A	
		KinCore			5Y8U:A
117	out	KLIFS		3WZU:A	
		KinCore			3WZU:A
120	out	KLIFS		2G2F:B;4WA9:A,B	
		KinCore			2G2F:B;4WA9:B
121	other	KLIFS		5B2L:A;5Y90:A (8)	
		KinCore			5B2L:A;5Y90:A (8)
		KinaMetrix		5B2L:A	
122	other	KLIFS		3ET7:A;3FZR:A (8)	
		KinCore			3ET7:A;3FZR:A (7)
		KinaMetrix		3ET7:A;3FZR:A (6)	
123	other	KLIFS		3FZP:A	
		KinCore			3FZP:A
		KinaMetrix		3FZP:A	
124	other	KLIFS		2BAQ:A;2GTM:A (15)	
		KinCore	2BAQ:A;2GTM:A (12)	6BSD:A	3IW5:A
		KinaMetrix		2BAQ:A;2GTM:A (13)	
125	out	KLIFS		4EH7:A;4GEO:A (5)	4I20:A
		KinCore			4GEO:A;4I1Z:A (5)
		KinaMetrix		4EH7:A;4GEO:A	4I1Z:A;4I20:A
126	out	KLIFS		3MTL:A	
		KinCore			3MTL:A
		KinaMetrix			3MTL:A
127	other	KLIFS		4AOT:A,B	

Continued on the next page

Table 3.6: Conflicting clusters.

ID	DFG	Source	in	other	out
		KinCore			4AOT:A,B
		KinaMetrix		4AOT:A	
128	other	KLIFS			2XNM:A;2XNN:A
130	other	KLIFS		1YW2:A;4FA2:A (4)	
		KinCore	1YW2:A;4FA2:A (4)		
		KinaMetrix		1YW2:A;4FA2:A	
131	other	KLIFS		3PIY:A	
		KinCore	3PIY:A		
		KinaMetrix		3PIY:A	
136	out	KinCore		2QQ7:B	
137	out	KLIFS	2J4Z:A;4UZH:A	3UNZ:A,B;3UO6:A,B (13)	
		KinCore		2J4Z:A;3UNZ:A,B (15)	
		KinaMetrix		2J4Z:A;3UNZ:A (8)	
138	out	KLIFS		2BMC:A,B;3EFW:A,B (22)	
		KinCore		2BMC:A,B;3EFW:A,B (22)	
		KinaMetrix	3K5U:A	2BMC:A;3EFW:A (11)	
139	out	KLIFS	3LAU:A;3W10:A		1MUO:A
		KinCore		1MUO:A;3LAU:A	
		KinaMetrix		3LAU:A;3W10:A	
140	out	KLIFS		2J50:A,B;3DAJ:A (9)	
		KinCore		2J50:A,B;3DAJ:A (7)	
		KinaMetrix		2J50:A;3DAJ:A (6)	
141	out	KLIFS		4ZTR:A	
		KinCore		4ZTR:A	
		KinaMetrix		4ZTR:A	
142	out	KLIFS		2W5B:A	2W5H:A
		KinCore		2W5B:A;2W5H:A	
		KinaMetrix		2W5B:A;2W5H:A	
144	out	KLIFS	5EW9:A		6C83:B;6CPG:A,D
		KinCore		5EW9:A;6C83:B	
		KinaMetrix		5EW9:A	
145	out	KLIFS		5E7R:A;6YG2:A	4ZJJ:D;4ZLO:A
		KinCore			4ZJJ:D;4ZLO:A
		KinaMetrix		4ZLO:A	
146	out	KLIFS			6ANL:A
		KinCore		6ANL:A	
148	other	KLIFS		3BV2:A;3BV3:A (9)	
		KinCore	3BV2:A;3O8P:A	4UMT:A;4UMU:A	3BV3:A;4EH8:A
		KinaMetrix		4EH8:A;4UMT:A (4)	3BV2:A;3BV3:A
149	other	KLIFS		3ZFY:A,B;5L6O:A (5)	
		KinCore	3ZFY:A,B;5L6O:A		
		KinaMetrix		3ZFY:A;5L6O:A	
150	other	KLIFS		3OEF:X	
		KinCore	3OEF:X		
		KinaMetrix		3OEF:X	
151	in	KLIFS		4O0U:A	
		KinCore	4O0U:A		
		KinaMetrix		4O0U:A	

Continued on the next page

Table 3.6: Conflicting clusters.

ID	DFG	Source	in	other	out
153	out	KLIFS		7O2V:A	
		KinCore			7O2V:A
154	other	KLIFS			5W5J:B
		KinCore	5W5J:B		
155	out	KLIFS		4JAI:A	
		KinCore			4JAI:A
		KinaMetrix			4JAI:A
156	out	KLIFS		3O51:A	
		KinCore			3O51:A
		KinaMetrix		3O51:A	
158	out	KLIFS		7AYH:A	
		KinCore			7AYH:A
159	other	KLIFS		3C4C:B	
		KinCore			3C4C:B
		KinaMetrix		3C4C:B	

Table 3.7: Profile positions. The columns, in left-to-right order: (1) structural region name, (2) PF00069 profile position, (3) Src residue and position, (4) Aurora residue and position, (5) the most frequently contacting ligand name, (6) the total number of ligand contacts, (7) the total importance of selected variables for the `DFGin`, `DFGother`, and `DFGout` models corresponding to this position, (8) the total importance for the `KinActive` model, (9), the top-1 important variable for the `DFG*` models, and (10) the top-1 important variable for the `KinActive` model.

Reg.	Pos.	Src	Aurora	Lig.	#Cont.	I^{tot}_{DFG}	I^{tot}_{KA}	F^{top-1}_{DFG}	F^{top-1}_{KA}
	1		F_{133}	42J	50	0.00	0.00		
	2		E_{134}	551	54	0.00	0.00		
	3		I_{135}	C1V	91	0.00	0.00		
	4	E_{273}	G_{136}	C1V	60	0.00	0.00		
	5	V_{274}	R_{137}	LU8	632	0.00	0.00		
β_1	6	K_{275}	P_{138}	ADP	402	0.12	0.00	Ψ_6	
	7	L_{276}	L_{139}	ANP	5611	0.06	0.00	$D_{7C_\beta,48C_\beta}$	$D_{7C_\beta,80C_\beta}$
	8	G_{277}	G_{140}	ANP	4497	0.00	0.00		
	9	Q_{278}	K_{141}	ANP	3425	0.07	0.00	$D_{9C_\beta,141C_\beta}$	
	10	G_{279}	G_{142}	ANP	2681	0.05	0.00	$SASA_{10}$	
	11	C_{280}	K_{143}	ANP	1731	0.01	0.00	$Pd_{10,11,12,13}$	
	12	F_{281}	F_{144}	ANP	2319	0.14	0.00	$SASA_{12}$	
	13	G_{282}	G_{145}	ANP	1592	0.09	0.00	Ψ_{13}	
β_2	14	E_{283}	N_{146}	ANP	997	0.54	0.00	$Pd_{14,15,16,17}$	

Continued on the next page...

Table 3.7: Profile positions.

Reg.	Pos.	Src	Aurora	Lig.	#Cont.	I^{tot}_{DFG}	I^{tot}_{KA}	F^{top-1}_{DFG}	F^{top-1}_{KA}
	15	V_{284}	V_{147}	ANP	6257	0.45	0.00	$Pd_{14,15,16,17}$	$D_{15C_\beta,142C_\beta}$
	16	W_{285}	Y_{148}	PDY	234	0.31	0.00	$Pd_{14,15,16,17}$	
	17	M_{286}	L_{149}	P4O	769	0.30	0.00	$Pd_{14,15,16,17}$	
	18	G_{287}	A_{150}	FGF	24	0.00	0.00		
	19	T_{288}	R_{151}	I46	24	0.00	0.00		$SASA_{19}$
	20		E_{152}	STU	18	0.00	0.00		
	21	W_{289}	K_{153}	5WE	32	0.00	0.00		
	22	N_{290}	Q_{154}	LU8	68	0.06	0.00	Ψ_{22}	
	23	G_{291}	S_{155}	LVY	28	0.00	0.00	$SASA_{23}$	
	24	T_{292}	K_{156}	SER	26	0.00	0.00		
	25	T_{293}	F_{157}	SER	71	0.00	0.00		
	26	R_{294}	I_{158}	B97	168	0.00	0.00		$SASA_{26}$
	27	V_{295}	L_{159}	LU8	157	0.00	0.01	$Pd_{27,28,29,30}$	$SASA_{27}$
β_3	28	A_{296}	A_{160}	ANP	7437	0.28	0.04	$D_{28C_\beta,142C_\beta}$	$D_{28C_\beta,142C_\beta}$
	29	I_{297}	L_{161}	LU8	1883	0.00	0.00	$Pd_{27,28,29,30}$	
	30	K_{298}	K_{162}	ANP	6963	0.83	0.00	$D_{30C_\beta,142C_\beta}$	$D_{30C_\beta,48C_\beta}$
	31	T_{299}	V_{163}	VFS	427	0.00	0.00		
	32	L_{300}	L_{164}	ANP	798	0.00	0.00		
	33	K_{301}	F_{165}	C1V	91	0.04	0.00	χ^1_{33}	Ψ_{33}
	34	P_{302}	K_{166}	JBJ	185	0.00	0.00		
	35	G_{303}	A_{167}	ANP	70	0.00	0.00		$SASA_{35}$
	36	T_{304}	Q_{168}	JBJ	125	0.00	0.00		
	37	M_{305}	L_{169}	1F8	126	0.00	0.00		
	38	S_{306}	E_{170}	1F8	79	0.00	0.00		
	39	P_{307}	K_{171}	SCN	46	0.04	0.01	Φ_{39}	$SASA_{39}$
	40		A_{172}	VNS	77	0.00	0.00		$SASA_{40}$
	41		G_{173}	SCN	73	0.07	0.01	χ^1_{41}	Ψ_{41}
	42		V_{174}	B5G	123	0.00	0.00		
αC	43	E_{308}	H_{176}	CO3	26	0.04	0.00	$Pd_{43,44,45,46}$	
	44	A_{309}	Q_{177}	ANP	418	0.04	0.00	$Pd_{43,44,45,46}$	$SASA_{44}$
	45	F_{310}	L_{178}	VNS	577	0.12	0.00	Φ_{45}	
	46	L_{311}	R_{179}	LU8	328	0.04	0.00	$Pd_{43,44,45,46}$	
	47	Q_{312}	R_{180}	B96	180	0.00	0.00	$SASA_{47}$	
	48	E_{313}	E_{181}	ADP	3870	0.32	0.68	$D_{14C_\beta,48C_\beta}$	$D_{48C_\beta,142C_\beta}$
	49	A_{314}	V_{182}	VNS	488	0.00	0.00		
	50	Q_{315}	E_{183}	LU8	83	0.00	0.00		

Continued on the next page...

Table 3.7: Profile positions.

Reg.	Pos.	Src	Aurora	Lig.	#Cont.	$\mathbf{I^{tot}_{DFG}}$	$\mathbf{I^{tot}_{KA}}$	$\mathbf{F^{top-1}_{DFG}}$	$\mathbf{F^{top-1}_{KA}}$
	51	V_{316}	I_{184}	STI	527	0.01	0.17	$SASA_{51}$	$SASA_{51}$
	52	M_{317}	Q_{185}	STI	2514	0.00	0.00	$SASA_{52}$	
	53	K_{318}	S_{186}	2AN	96	0.00	0.00	$Pd_{53,54,55,56}$	
	54	K_{319}	H_{187}	DTU	28	0.22	0.00	$SASA_{54}$	
	55	L_{320}	L_{188}	STI	662	0.11	0.00	$SASA_{55}$	
	56	R_{321}	R_{189}	5WE	52	0.00	0.00	$Pd_{53,54,55,56}$	
	57	H_{322}	H_{190}	6J9	35	0.00	0.00		
	58	E_{323}	P_{191}	CCX	19	0.00	0.00		
	59	K_{324}	N_{192}	1HW	12	0.00	0.00		
	60	L_{325}	I_{193}	STI	746	0.05	0.00	$SASA_{60}$	
	61	V_{326}	L_{194}	ANP	7730	0.33	0.00	$D_{61C_\beta,141C_\beta}$	$D_{48C_\beta,61C_\beta}$
	62	Q_{327}	R_{195}	SB4	730	0.20	0.00	χ^1_{62}	
	63	L_{328}	L_{196}	SB4	1083	0.15	0.00	$SASA_{63}$	
	64	Y_{329}	Y_{197}	LU8	183	0.00	0.00	$SASA_{64}$	
β_4	65	A_{330}	G_{198}	LU8	158	0.00	0.00		$Pd_{65,66,67,68}$
	66	V_{331}	Y_{199}	LU8	168	0.00	0.00		$Pd_{65,66,67,68}$
	67	V_{332}	F_{200}	LU8	141	0.00	0.00		$Pd_{65,66,67,68}$
	68	S_{333}	H_{201}	1F8	108	0.00	0.03		$Pd_{68,69,70,71}$
	69		D_{202}	551	63	0.04	0.03	Ψ_{69}	$Pd_{68,69,70,71}$
	70	E_{334}	A_{203}	A8Q	37	0.00	0.03	$SASA_{70}$	$Pd_{68,69,70,71}$
	71	E_{335}	T_{204}	M77	17	0.00	0.03		$Pd_{68,69,70,71}$
	72	P_{336}	R_{205}	C1V	82	0.00	0.00	$SASA_{72}$	
β_5	73	I_{337}	V_{206}	1F8	107	0.00	0.00		
	74	Y_{338}	Y_{207}	C1V	108	0.00	0.00		
	75	I_{339}	L_{208}	LU8	2561	0.00	0.00		
	76	V_{340}	I_{209}	LU8	1410	0.17	0.00	Φ_{76}	
	77	T_{341}	L_{210}	ANP	8028	0.09	0.02	$D_{77C_\beta,141C_\beta}$	$D_{48C_\beta,77C_\beta}$
	78	E_{342}	E_{211}	ANP	7870	0.04	0.00	$D_{78C_\beta,79C_\beta}$	
	79	Y_{343}	Y_{212}	ANP	7824	0.23	0.00	$D_{79C_\beta,142C_\beta}$	$D_{79C_\beta,142C_\beta}$
	80	M_{344}	A_{213}	ANP	7873	0.15	0.00	$D_{80C_\beta,142C_\beta}$	$D_{7C_\beta,80C_\beta}$
	81	S_{345}	P_{214}	ANP	7083	0.22	0.14	$D_{81C_\beta,142C_\beta}$	$D_{48C_\beta,81C_\beta}$
	82	K_{346}	L_{215}	STU	4755	0.14	0.00	$D_{30C_\beta,82C_\beta}$	
	83	G_{347}	G_{216}	ANP	6086	0.00	0.01	$D_{83C_\beta,142C_\beta}$	$D_{83C_\beta,127C_\beta}$
	84	S_{348}	T_{217}	ANP	4882	0.12	0.00	$D_{84C_\beta,142C_\beta}$	$D_{48C_\beta,84C_\beta}$
	85	L_{349}	V_{218}	IMD	273	0.00	0.00		

Continued on the next page...

Table 3.7: Profile positions.

Reg.	Pos.	Src	Aurora	Lig.	#Cont.	I^{tot}_{DFG}	I^{tot}_{KA}	F^{top-1}_{DFG}	F^{top-1}_{KA}
	86	L_{350}	Y_{219}	ANP	658	0.00	0.00		
	87	D_{351}	R_{220}	ANP	3528	0.32	0.03	$D_{87C_\beta,142C_\beta}$	$D_{87C_\beta,142C_\beta}$
	88	F_{352}	E_{221}	D5Q	317	0.00	0.00		
	89	L_{353}	L_{222}	IMD	107	0.00	0.00		
	90	K_{354}	Q_{223}	IMD	169	0.08	0.00	χ^1_{90}	
	91	T_{357}	K_{224}	YY3	285	0.00	0.00		
	92	G_{358}	L_{225}	LDN	80	0.00	0.00		
	93	K_{359}	S_{226}	LDN	98	0.00	0.00		
	94	Y_{360}	K_{227}	9XA	26	0.00	0.00		
	95	L_{361}	F_{228}	A8Q	123	0.00	0.00		
	96	R_{362}	D_{229}	STI	35	0.00	0.00		
	97	L_{363}	E_{230}	STI	34	0.00	0.00		
	98	P_{364}	Q_{231}	MS7	30	0.16	0.00	Ψ_{98}	
	99	Q_{365}	R_{232}	551	109	0.00	0.00		
	100	L_{366}	T_{233}	A8Q	125	0.00	0.00	Φ_{100}	$SASA_{100}$
	101	V_{367}	A_{234}	MS7	40	0.07	0.00	χ^1_{101}	
	102	D_{368}	T_{235}	CCX	26	0.00	0.00	$SASA_{102}$	
	103	M_{369}	Y_{236}	551	108	0.00	0.00		
	104	A_{370}	I_{237}	A8Q	115	0.00	0.00		
H_E	105	A_{371}	T_{238}	GLN	8	0.00	0.00		
	106	Q_{372}	E_{239}	CCX	8	0.09	0.00	Φ_{106}	
	107	I_{373}	L_{240}	551	97	0.00	0.00		
	108	A_{374}	A_{241}	LGY	3	0.00	0.00		
	109	S_{375}	N_{242}	6J9	21	0.00	0.00		
	110	G_{376}	A_{243}	ARS	5	0.00	0.00		
	111	M_{377}	L_{244}	ARS	11	0.00	0.00		
	112	A_{378}	S_{245}	6J9	22	0.07	0.01	$Pd_{112,113,114,115}$	$SASA_{112}$
	113	Y_{379}	Y_{246}	4OR	47	0.07	0.00	$Pd_{112,113,114,115}$	
	114	V_{380}	C_{247}	STI	557	0.22	0.00	$SASA_{114}$	
	115	E_{381}	H_{248}	XK4	7	0.07	0.00	$Pd_{112,113,114,115}$	
	116	R_{382}	S_{249}	PG6	17	0.00	0.00	$SASA_{116}$	
	117	M_{383}	K_{250}	4OR	38	0.00	0.01		$SASA_{117}$
	118	N_{384}	R_{251}	ANP	31	0.06	0.00	$SASA_{118}$	
	119	Y_{385}	V_{252}	STI	569	0.00	0.00		$SASA_{119}$
	120	V_{386}	I_{253}	STI	256	0.00	2.99		$SASA_{120}$
	121	H_{387}	H_{254}	STI	642	0.23	0.06	$Pd_{121,122,123,124}$	$Pd_{121,122,123,124}$

Continued on the next page...

Table 3.7: Profile positions.

Reg.	Pos.	Src	Aurora	Lig.	#Cont.	$\mathbf{I^{tot}_{DFG}}$	$\mathbf{I^{tot}_{KA}}$	$\mathbf{F^{top-1}_{DFG}}$	$\mathbf{F^{top-1}_{KA}}$
	122	R_{388}	R_{255}	STI	192	0.09	0.16	Φ_{122}	$SASA_{122}$
	123	D_{389}	D_{256}	ANP	870	0.46	0.14	Ψ_{123}	$SASA_{123}$
	124	L_{390}	I_{257}	AF3	7	0.12	0.03	$Pd_{121,122,123,124}$	$Pd_{121,122,123,124}$
	125	R_{391}	K_{258}	ANP	931	0.04	0.21	$Pd_{123,124,125,126}$	$D_{125C_\beta,142C_\beta}$
	126	A_{392}	P_{259}	IMD	335	0.00	0.00	$Pd_{123,124,125,126}$	
	127	A_{393}	E_{260}	ANP	5551	0.61	0.01	$D_{127C_\beta,142C_\beta}$	$D_{83C_\beta,127C_\beta}$
	128	N_{394}	N_{261}	ANP	4915	0.31	0.01	$D_{127C_\beta,128C_\beta}$	$D_{128C_\beta,141C_\beta}$
	129	I_{395}	L_{262}	LU8	1026	0.00	0.00		
	130	L_{396}	L_{263}	ANP	7892	0.11	0.00	$D_{130C_\beta,142C_\beta}$	$D_{123C_\beta,130C_\beta}$
CL	131	V_{397}	L_{264}	T3B	272	0.00	0.00		
	132	G_{398}	G_{265}	499	26	0.04	0.00	χ^1_{132}	
	133	E_{399}	S_{266}	9E1	61	0.00	0.00	Φ_{133}	χ^1_{133}
	134	N_{400}	A_{267}	V62	59	0.05	0.00	$Pd_{134,135,136,137}$	
	135	L_{401}	G_{268}	CCX	20	0.05	0.00	$Pd_{134,135,136,137}$	
	136	V_{402}	E_{269}	CCX	19	0.05	0.00	$Pd_{134,135,136,137}$	
	137	C_{403}	L_{270}	A8Q	66	0.11	0.00	Ψ_{137}	χ^1_{137}
	138	K_{404}	K_{271}	V62	59	0.00	0.00	$Pd_{137,138,139,140}$	
	139	V_{405}	I_{272}	STI	1195	1.71	0.05	$Pd_{139,140,141,142}$	$Pd_{139,140,141,142}$
	140	A_{406}	A_{273}	ANP	7621	2.66	0.05	$Pd_{139,140,141,142}$	$Pd_{139,140,141,142}$
	141	D_{407}	D_{274}	ANP	7320	4.75	2.53	$Pd_{139,140,141,142}$	$Pd_{141,142,143,144}$
	142	F_{408}	F_{275}	LU8	3121	4.28	3.80	$Pd_{139,140,141,142}$	$Pd_{141,142,143,144}$
	143	G_{409}	G_{276}	ATP	1172	0.12	2.77	Ψ_{143}	$Pd_{141,142,143,144}$
	144	L_{410}	W_{277}	ANP	1107	0.18	2.69	Ψ_{144}	$Pd_{141,142,143,144}$
	145	A_{411}	S_{278}	ANP	373	0.11	0.49	Φ_{145}	$SASA_{145}$
	146	R_{412}	V_{279}	ANP	328	0.10	0.40	Φ_{146}	Φ_{146}
AL_N	147	L_{413}	H_{280}	JBJ	181	0.00	0.07		$Pd_{145,146,147,148}$
	148	I_{414}	A_{281}	4BM	292	0.00	0.04		$Pd_{145,146,147,148}$
	149	E_{415}	P_{282}	ANP	195	0.00	0.10		χ^1_{149}
	150	D_{416}	S_{283}	VNS	106	0.00	0.00		$Pd_{147,148,149,150}$
	151	N_{417}	S_{284}	JBJ	43	0.00	0.00	$SASA_{151}$	
	152	E_{418}	R_{285}	4BM	95	0.00	0.00		
	153			NH4	30	0.00	0.01		$Pd_{153,154,155,156}$
	154	Y_{419}	R_{286}	746	95	0.00	0.01		$Pd_{153,154,155,156}$
	155	T_{420}	T_{287}	ATP	49	0.00	0.01		$Pd_{153,154,155,156}$
	156	A_{421}	T_{288}	ANP	50	0.00	0.01		$Pd_{153,154,155,156}$
	157	R_{422}	L_{289}	ANP	79	0.00	0.28		$Pd_{157,158,159,160}$

Continued on the next page...

AL_C

Table 3.7: Profile positions.

Reg.	Pos.	Src	Aurora	Lig.	#Cont.	I^{tot}_{DFG}	I^{tot}_{KA}	F^{top-1}_{DFG}	F^{top-1}_{KA}
	158	Q_{423}	C_{290}	ANP	94	0.00	0.62	$SASA_{158}$	$Pd_{157,158,159,160}$
	159	G_{424}	G_{291}	ANP	67	0.00	0.39		$Pd_{157,158,159,160}$
	160	A_{425}	T_{292}	ANP	168	0.04	0.39	$Pd_{160,161,162,163}$	$Pd_{157,158,159,160}$
	161	K_{426}	L_{293}	SCN	41	0.04	0.11	$Pd_{160,161,162,163}$	$Pd_{158,159,160,161}$
	162	K_{430}	D_{294}	IMD	51	0.17	0.00	Φ_{162}	
	163	W_{431}	Y_{295}	IMD	40	0.08	0.00	$Pd_{160,161,162,163}$	
	164	T_{432}	L_{296}	SCN	36	0.08	0.01	$Pd_{164,165,166,167}$	χ^1_{164}
	165	A_{433}	P_{297}	1M3	9	0.08	0.00	$Pd_{164,165,166,167}$	
	166	P_{434}	P_{298}	IRG	53	0.05	0.00	$Pd_{164,165,166,167}$	
	167	E_{435}	E_{299}	I46	50	0.05	0.00	$Pd_{164,165,166,167}$	
	168	A_{436}	M_{300}	SCN	43	0.00	0.00		
	169	A_{437}	I_{301}	CPS	47	0.02	0.00	Φ_{169}	
	170	L_{438}	E_{302}	IRG	63	0.00	0.00		
	171	Y_{439}	G_{303}	I46	91	0.00	0.00		
	172	G_{440}	R_{304}	CCX	56	0.00	0.00		
	173	R_{441}	M_{305}	SCN	42	0.00	0.00		
	174	F_{442}	H_{306}	TPO	44	0.00	0.16		$SASA_{174}$
	175	T_{443}	D_{307}	ANP	17	0.00	0.00		
	176	I_{444}	E_{308}	IMD	12	0.00	0.00		
	177	K_{445}	K_{309}	IMD	12	0.00	0.00	χ^1_{177}	
	178	S_{446}	V_{310}	1N1	13	0.00	0.00		
	179	D_{447}	D_{311}	AF3	6	0.00	0.00		
	180	V_{448}	L_{312}	THR	12	0.00	0.00		
	181	W_{449}	W_{313}	1M3	9	0.00	0.00		
H_F	182	S_{450}	S_{314}	AF3	9	0.09	0.00	$SASA_{182}$	
	183	F_{451}	L_{315}	CYS	9	0.00	0.00		
	184	G_{452}	G_{316}	CYS	2	0.00	0.00		
	185	I_{453}	V_{317}	1M3	7	0.06	0.00	Ψ_{185}	
	186	L_{454}	L_{318}	551	99	0.00	0.00	$Pd_{186,187,188,189}$	
	187	L_{455}	C_{319}	MS7	84	0.00	0.00	$Pd_{186,187,188,189}$	
	188	T_{456}	Y_{320}	LEU	15	0.00	0.00	$Pd_{186,187,188,189}$	
	189	E_{457}	E_{321}	IMD	112	0.00	0.00	$Pd_{186,187,188,189}$	
	190	L_{458}	F_{322}	A8Q	121	0.00	0.00		
	191	T_{459}	L_{323}	MS7	49	0.00	0.00		
	192	T_{460}	V_{324}	LOT	35	0.00	0.00		

Continued on the next page...

Table 3.7: Profile positions.

Reg.	Pos.	Src	Aurora	Lig.	#Cont.	$\mathbf{I^{tot}_{DFG}}$	$\mathbf{I^{tot}_{KA}}$	$\mathbf{F^{top-1}_{DFG}}$	$\mathbf{F^{top-1}_{KA}}$
	193	G_{462}	G_{325}	IMD	77	0.00	0.00		
	194	R_{463}	K_{326}	IMD	101	0.00	0.00		
	195	V_{464}	P_{327}	IMD	118	0.00	0.00	$Pd_{195,196,197,198}$	
	196	P_{465}	P_{328}	LOW	23	0.00	0.00	$Pd_{195,196,197,198}$	
FL	197	Y_{466}	F_{329}	PDY	46	0.01	0.00	Φ_{197}	
	198	P_{467}	E_{330}	PDY	59	0.02	0.00	Φ_{198}	
	199	G_{468}	A_{331}	PDY	47	0.09	0.00	$Pd_{199,200,201,202}$	
	200	M_{469}	N_{332}	IMD	36	0.09	0.00	$Pd_{199,200,201,202}$	
	201	V_{470}	T_{333}	IMD	32	0.09	0.00	$Pd_{199,200,201,202}$	
	202	N_{471}	Y_{334}	CPS	39	0.09	0.00	$Pd_{199,200,201,202}$	
	203	R_{472}	Q_{335}	CPS	37	0.00	0.00		
	204	E_{473}	E_{336}	SCN	21	0.00	0.00		
	205			XJV	2	0.00	0.00		
	206			CPS	19	0.00	0.00		
	207			IHP	12	0.00	0.00		
H_G	208			LEU	8	0.00	0.00		
	209	V_{474}	T_{337}	1M3	23	0.00	0.00		
	210	L_{475}	Y_{338}	CPS	36	0.08	0.00	$SASA_{210}$	
	211	D_{476}	K_{339}	ANP	15	0.00	0.00	Φ_{211}	
	212	Q_{477}	R_{340}	SCN	11	0.00	0.00		
	213	V_{478}		I46	56	0.00	0.00		
	214	E_{479}		PDY	34	0.00	0.00		
	215			IHP	30	0.00	0.00		
	216			LEU	15	0.00	0.00		
	217			I46	48	0.00	0.00		
	218			SCN	26	0.00	0.02		$SASA_{218}$
	219			IWU	25	0.00	0.00		
	220			1M3	14	0.00	0.00		
	221			SCN	15	0.00	0.00		
	222		I_{341}	1M3	12	0.00	0.00	Ψ_{222}	
	223	R_{480}	S_{342}	C15	8	0.00	0.00	χ^1_{223}	
	224	G_{481}	R_{343}	IRG	51	0.00	0.00	$Pd_{224,225,226,227}$	χ^1_{224}
	225	Y_{482}	V_{344}	IRG	61	0.00	0.00	$Pd_{224,225,226,227}$	
	226	R_{483}	E_{345}	I46	73	0.00	0.00	$Pd_{224,225,226,227}$	
	227	M_{484}	F_{346}	IRG	40	0.00	0.00	$Pd_{224,225,226,227}$	
	228	P_{485}	T_{347}	XHV	15	0.00	0.00		

Continued on the next page...

Table 3.7: Profile positions.

Reg.	Pos.	Src	Aurora	Lig.	#Cont.	I^{tot}_{DFG}	I^{tot}_{KA}	F^{top-1}_{DFG}	F^{top-1}_{KA}
	229	C_{486}	F_{348}	PDY	31	0.08	0.00	χ^1_{229}	
	230	P_{487}	P_{349}	3YY	17	0.00	0.00		
	231	P_{488}	D_{350}	STI	24	0.00	0.00		
	232	E_{489}	F_{351}	STI	33	0.00	0.00	$SASA_{232}$	
	233	C_{490}	V_{352}	MS7	45	0.01	0.01	$SASA_{233}$	Φ_{233}
	234	P_{491}	T_{353}	STI	38	0.00	0.00		
	235	E_{492}	E_{354}	YDJ	19	0.00	0.01		$SASA_{235}$
	236	S_{493}	G_{355}	YDJ	5	0.06	0.00	Ψ_{236}	
	237	L_{494}	A_{356}	STI	28	0.00	0.00		
H_H	238	H_{495}	R_{357}	IWU	15	0.00	0.00		
	239	D_{496}	D_{358}	IWU	8	0.00	0.00		
	240	L_{497}	L_{359}	PG6	2	0.00	0.00		
	241	M_{498}	I_{360}	KWD	17	0.00	0.00		
	242	C_{499}	S_{361}	IWU	15	0.00	0.00	$Pd_{242,243,244,245}$	
	243	Q_{500}	R_{362}	THR	14	0.00	0.00	$SASA_{243}$	
	244	C_{501}	L_{363}	THR	3	0.00	0.00	χ^1_{244}	
	245	W_{502}	L_{364}	THR	7	0.00	0.00	$Pd_{242,243,244,245}$	
	246	R_{503}	K_{365}	9HB	10	0.05	0.00	$SASA_{246}$	
	247	K_{504}	H_{366}	I46	56	0.09	0.00	χ^1_{247}	
	248	E_{505}	N_{367}	I46	59	0.00	0.00		
	249	P_{506}	P_{368}	IRG	55	0.00	0.00		
	250	E_{507}	S_{369}	I46	58	0.00	0.00		
	251	E_{508}	Q_{370}	3U9	16	0.00	0.00	$SASA_{251}$	
	252	R_{509}	R_{371}	I46	27	0.17	0.00	$SASA_{252}$	
	253	P_{510}	P_{372}	XK4	10	0.00	0.00		
	254	T_{511}	M_{373}	XK4	9	0.11	0.00	$SASA_{254}$	
	255	F_{512}	L_{374}	XK4	11	0.00	0.00		
	256	E_{513}	R_{375}	ANP	36	0.13	0.00	χ^1_{256}	
	257	Y_{514}	E_{376}	ALA	9	0.00	0.00		$SASA_{257}$
H_I	258	L_{515}	V_{377}	ALA	5	0.02	0.00	$SASA_{258}$	
	259		L_{378}	6J9	21	0.00	0.00		
	260		E_{379}	LGY	5	0.00	0.00		
	261		H_{380}	GLN	4	0.00	0.00		
	262		P_{381}	GLN	3	0.00	0.00	$SASA_{262}$	
	263		W_{382}	GLN	5	0.00	0.00		
	264		I_{383}	GLN	7	0.00	0.00		

Table 3.8: Parameters subject to automatic fine-tuning with `optuna`.

Parameter	Min/Choice1	Max/Choice2	Description
		Logistic Regression	
`C`	0	1	Controls the regularization strength. Lower values imply stronger regularization.
`l1_ratio`	0	1	The ρ parameter balances l_1 and l_2 regularization as described in the main text.
`class_weight`	-	`balanced`	Enables weighting samples according to their class frequencies.
		XGBoost	
`learning_rate`	10^{-5}	10	The step in the gradient descent algorithm.
`max_depth`	4	12	Maximum tree depth.
`gamma`	10^{-3}	10	Regularization parameter – a complexity cost associated with introducing nodes.
`reg_lambda`	10^{-3}	5	Weight of the l_2 regularization.
`reg_alpha`	10^{-3}	5	Weight of the l_1 regularization
`colsample_bytree`	0.1	1.0	A fraction of features sampled for an individual tree.
`colsample_bylevel`	0.1	1.0	A fraction of features sampled when introducing a node of an individual tree.
`scale_pos_weight`	0	5	For binary classification, the loss multiplier for 1-labeled instances.

Table 3.9: Software and data sources

Name/Title	Category	Purpose/Context/Usage	Ref.
		Standalone software	
`ChimeraX`	Structure visualization	Visualizing structural superpositions and individual structures.	[79]
`mafft`	Sequence alignment	Pairwise sequence alignments.	[68]
		Python libraries	
`biotite`	Toolkit	Handling structure-related data.	[80]
`ete3`	Phylogeny	Dendrograms' visualization.	[81]
`matplotlib`	Data visualization	Producing plots.	
`optuna`	Machine-learning	Hyperparameters' optimization.	[78]
`pandas`	Data manipulation	Constructing and manipulating tabular data.	[82]
`PyHMMer`	Sequence HMM modeling	Extracting domains	[67]
`scikit-learn`	Machine-learning	Logistic Regression and performance metrics.	[83]
`scipy`	Scientific computing	Agglomerative clustering algorithms.	[84]
`seaborn`	Data visualization	Producing plots.	[85]
`shap`	Machine-learning	Estimating SHAP values to compute feature importance.	[76, 77]
`xgboost`	Machine-learning	Using as a part of a feature selection algorithm.	[73]
		Developed software	
`eBoruta`*	Machine-learning	Feature selection and ranking.	-
`lXtractor`	Toolkit	Creating the PK data collection.	-
`KinActive`	Toolkit	Reproducing and using the results of this study.	-
		Data sources	
UniProt	Database	Protein canonical sequences and their metadata	[65]
PDB	Database	Protein structures and their metadata.	[64]
SIFTS	Database	ID mapping between UniProt and PDB.	[50]
Pfam	Database	PF00069 profile used with PyHMMer	[44]

* Inspired by two open-source implementations: `BorutaPy` (https://github.com/scikit-learn-contrib/boruta_py) and `BorutaShap` (https://github.com/Ekeany/Boruta-Shap).

References

1. Manning, G. The protein kinase complement of the human genome. en. *Science* **298,** 1912–1934. ISSN: 00368075, 10959203 (2002).
2. Stancik, I. A. *et al.* Serine/Threonine protein kinases from bacteria, archaea and eukarya share a common evolutionary origin deeply rooted in the tree of life. en. *J. Mol. Biol.* **430,** 27–32. ISSN: 00222836 (2018).
3. Irby, R. B. & Yeatman, T. J. Role of Src expression and activation in human cancer. en. *Oncogene* **19,** 5636–5642. ISSN: 0950-9232, 1476-5594 (2000).
4. Wilson, L. J. *et al.* New perspectives, opportunities, and challenges in exploring the human protein kinome. en. *Canc. Res.,* 16 (2017).
5. Huse, M. & Kuriyan, J. The conformational plasticity of protein kinases. *Cell* **109,** 275–282. ISSN: 0092-8674 (2002).
6. Nagar, B. *et al.* Structural basis for the autoinhibition of c-Abl tyrosine kinase. *Cell* **112,** 859–871. ISSN: 0092-8674 (2003).
7. Johnson, L. N. Protein kinase inhibitors: contributions from structure to clinical compounds. *Quart. Rev. Biophys.* **42,** 1–40. ISSN: 0033-5835 (2009).
8. Xie, T., Saleh, T., Rossi, P. & Kalodimos, C. G. Conformational states dynamically populated by a kinase determine its function. *Science* **370,** eabc2754. ISSN: 0036-8075 (2020).
9. Kanev, G. K. *et al.* The landscape of atypical and eukaryotic protein kinases. *Trends Pharm. Sci.* **40,** 818–832. ISSN: 0165-6147 (2019).
10. Meng, Y., Lin, Y.-l. & Roux, B. Computational study of the "DFG-flip" conformational transition in c-Abl and c-Src tyrosine kinases. *J. Phys. Chem. B* **119,** 1443–1456. ISSN: 1520-6106 (2015).
11. Narayan, B. *et al.* The transition between active and inactive conformations of Abl kinase studied by rock climbing and milestoning. *Biochem. Biophys. Acta* **1864,** 129508. ISSN: 0304-4165 (2020).
12. Liu, Y. & Gray, N. S. Rational design of inhibitors that bind to inactive kinase conformations. *Nat. Chem. Biol.* **2,** 358–364. ISSN: 1552-4450 (2006).
13. Bixby, D. & Talpaz, M. Seeking the causes and solutions to imatinib-resistance in chronic myeloid leukemia. en. *Leukemia* **25,** 7–22. ISSN: 0887-6924, 1476-5551. (2020) (2011).

14. Wang, J. & Zhuang, S. Src family kinases in chronic kidney disease. eng. *Amer. J. Physio.: Renal Physiol.* **313.** Edition: 2017/06/14 Publisher: American Physiological Society, F721–F728. ISSN: 1522-1466 (2017).

15. Soverini, S., Mancini, M., Bavaro, L., Cavo, M. & Martinelli, G. Chronic myeloid leukemia: the paradigm of targeting oncogenic tyrosine kinase signaling and counteracting resistance for successful cancer therapy. *Molec. Canc.* **17,** 49 (2018).

16. Schindler, T. *et al.* Crystal structure of Hck in complex with a Src family-selective tyrosine kinase inhibitor. *Molec. Cell* **3,** 639–648 (1999).

17. Schindler, T. *et al.* Structural Mechanism for STI-571 Inhibition of Abelson Tyrosine Kinase. *Science* **289,** 1938–1942 (2000).

18. Hubbard, S. R. & Hill, J. H. Protein tyrosine kinase structure and function. *Ann. Rev. Biochem* **69,** 373–398 (2000).

19. Levinson, N. M. *et al.* A Src-like inactive conformation in the Abl tyrosine kinase domain. *PLoS Biology* **4,** 753–767 (2006).

20. Cowan-Jacob, S. *et al.* Structural biology contributions to the discovery of drugs to treat chronic myelogenous leukaemia. *Acta Cryst. D* **63,** 80–93 (2007).

21. Ozkirimli, E. & Post, C. B. Src kinase activation: a switched electrostatic network. *Prot. Sci.* **15,** 1051–1062 (2006).

22. Yang, S., Banavali, N. K. & Roux, B. Mapping the conformational transition in Src activation by cumulating the information from multiple molecular dynamics trajectories. *Proc. Natl. Acad. Sci. USA* **106,** 3776–3781 (2009).

23. Aleksandrov, A. & Simonson, T. Molecular dynamics simulations show that conformational selection governs the binding preferences of imatinib for several tyrosine kinases. en. *J. Biol. Chem.* **285,** 13807–13815. ISSN: 0021-9258, 1083-351X. (2020) (2010).

24. Agafonov, R. V., Wilson, C., Otten, R., Buosi, V. & Kern, D. Energetic dissection of Gleevec's selectivity toward human tyrosine kinases. *Nat. Struct. Mol. Biol.* **21,** 848–853. ISSN: 1545-9993 (2014).

25. Morando, M. A. *et al.* Conformational selection and induced fit mechanisms in the binding of an anticancer drug to the c-Src kinase. *Scient. Rep.* **6,** 24439 (2016).

26. Van Linden, O. P. J., Kooistra, A. J., Leurs, R., de Esch, I. J. P. & de Graaf, C. KLIFS: a knowledge-based structural database to navigate kinase–ligand interaction space. *J. Medicin. Chem.* **57.** Publisher: American Chemical Society, 249–277. ISSN: 0022-2623 (2014).

27. Kooistra, A. J. *et al.* KLIFS: a structural kinase-ligand interaction database. *Nucl. Acids Res.* **44,** D365–D371. ISSN: 0305-1048 (2016).

28. Kanev, G. K., de Graaf, C., Westerman, B. A., de Esch, I. J. P. & Kooistra, A. J. KLIFS: an overhaul after the first 5 years of supporting kinase research. *Nucl. Acids Res.* **49,** gkaa895–. ISSN: 0305-1048 (2020).

29. Modi, V. & Dunbrack, R. L. Defining a new nomenclature for the structures of active and inactive kinases. *Proc. Natl. Acad. Sci. USA* **116,** 6818 (2019).

30. Modi, V. & Dunbrack, R. L. Kincore: a web resource for structural classification of protein kinases and their inhibitors. *Nucl. Acids Res.* **50,** gkab920–. ISSN: 0305-1048 (Oct. 2021).

31. Sapoval, N. *et al.* Current progress and open challenges for applying deep learning across the biosciences. *Nat. Commun.* **13,** 1728 (2022).

32. Noé, F., Fabritiis, G. D. & Clementi, C. Machine learning for protein folding and dynamics. *Curr. Opin. Struct. Biol.* **60,** 77–84. ISSN: 0959-440X (2020).

33. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature,* 1–11. ISSN: 0028-0836 (2021).

34. Roney, J. P. & Ovchinnikov, S. State-of-the-art estimation of protein model accuracy using AlphaFold. *Phys. Rev. Lett.* **129,** 238101. ISSN: 0031-9007 (2022).

35. Gligorijević, V. *et al.* Structure-based protein function prediction using graph convolutional networks. *Nat. Commun.* **12,** 3168 (2021).

36. Casadio, R., Martelli, P. L. & Savojardo, C. Machine learning solutions for predicting protein–protein interactions. *Wiley Interdisc. Rev.: Comp. Mol. Sci.* **12.** ISSN: 1759-0876 (2022).

37. Wei, J., Chen, S., Zong, L., Gao, X. & Li, Y. Protein–RNA interaction prediction with deep learning: structure matters. *Briefings Bioinf.* **23,** bbab540. ISSN: 1467-5463 (2021).

38. Malbranke, C., Bikard, D., Cocco, S., Monasson, R. & Tubiana, J. Machine learning for evolutionary-based and physics-inspired protein design: Current and future synergies. *Curr. Opin. Struct. Biol.* **80,** 102571. ISSN: 0959-440X. eprint: 2208.13616 (2023).

39. Ming, Y. *et al.* A review of enzyme design in catalytic stability by artificial intelligence. *Briefings in Bioinformatics* **24.** ISSN: 1467-5463 (2023).

40. Gao, W., Mahajan, S. P., Sulam, J. & Gray, J. J. Deep learning in protein structural modeling and design. *arXiv.* eprint: 2007.08383 (2020).

41. Staszak, M. *et al.* Machine learning in drug design: use of artificial intelligence to explore the chemical structure–biological activity relationship. *Wiley Interdisc. Rev.: Comp. Mol. Sci.* **12.** ISSN: 1759-0876 (2022).

42. Vamathevan, J. *et al.* Applications of machine learning in drug discovery and development. *Nat. Rev. Drug Disc.* **18,** 463–477. ISSN: 1474-1776 (2019).

43. Durbin, R., Eddy, S. R., Krogh, A. & Mitchison, G. *Biological sequence analysis* (Cambridge University Press, Cambridge, United Kingdom, 2002).

44. Mistry, J. *et al.* Pfam: the protein families database in 2021. *Nucl. Acids Res.* **49,** gkaa913–. ISSN: 0305-1048 (2020).

45. Friedman, J. H. Greedy function approximation: a gradient boosting machine. *The Annals Stat.* **29.** ISSN: 0090-5364 (2001).

46. Friedman, J. H. Stochastic gradient boosting. *Comp. Stat. & Data An.* **38,** 367–378. ISSN: 0167-9473 (2002).

47. Kursa, M. B. & Rudnicki, W. R. Feature selection with the Boruta package. *J. Stat. Soft.* **36,** 1–13 (2010).

48. Boutet, E., Lieberherr, D., Tognolli, M., Schneider, M. & Bairoch, A. UniProtKB/Swiss-Prot. *Meth. Mol. Biol.* **406,** 89–112. ISSN: 1064-3745 (2007).

49. Varadi, M. *et al.* AlphaFold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucl. Acids Res.* **50,** D439–D444. ISSN: 0305-1048 (2021).

50. Velankar, S. *et al.* SIFTS: structure integration with function, taxonomy and sequences resource. eng. *Nucl. Acids Res.* **41.** Edition: 2012/11/29 Publisher: Oxford University Press, D483–D489. ISSN: 1362-4962 (2013).

51. Möbitz, H. The ABC of protein kinase conformations. *Biochem. Biophys. Acta* **1854,** 1555–1566. ISSN: 1570-9639 (2015).

52. Rahman, R., Ung, P. M.-U. & Schlessinger, A. KinaMetrix: a web resource to investigate kinase conformations and inhibitor space. *Nucl. Acids Res.* **47,** D361–D366. ISSN: 0305-1048 (2019).

53. McSkimming, D. I., Rasheed, K. & Kannan, N. Classifying kinase conformations using a machine learning approach. *BMC Bioinf.* **18,** 86 (2017).

54. Reveguk, I. & Simonson, T. *Classifying protein kinase conformations with machine learning: software* version v0.1 (Zenodo, 2023). https://doi.org/10.5281/zenodo.8172910.

55. Utepbergenov, D. *et al.* Insights into the inhibition of the p90 ribosomal S6 kinase (RSK) by the flavonol glycoside SL0101 from the 1.5 Å crystal structure of the N-terminal domain of RSK2 with bound inhibitor. *Biochemistry* **51,** 6499–6510. ISSN: 0006-2960 (2012).

56. Rudolph, M. J., Amodeo, G. A. & Tong, L. An inhibited conformation for the protein kinase domain of the Saccharomyces cerevisiae AMPK homolog Snf1. *Acta Cryst. D* **66,** 999–1002. ISSN: 1744-3091 (2010).

57. Melo-Hanchuk, T. D. *et al.* NEK1 kinase domain structure and its dynamic protein interactome after exposure to Cisplatin. *Scient. Rep.* **7,** 5445 (2017).

58. Young, M. A. *et al.* Structure of the kinase domain of an Imatinib-resistant Abl mutant in complex with the Aurora kinase inhibitor VX-680. *Canc. Res.* **66,** 1007–1014. ISSN: 0008-5472 (2006).

59. Levinson, N. M. & Boxer, S. G. A conserved water-mediated hydrogen bond network defines bosutinib's kinase selectivity. *Nat. Chem. Biol.* **10,** 127–132. ISSN: 1552-4450 (2014).

60. Pemovska, T. *et al.* Axitinib effectively inhibits BCR-ABL1(T315I) with a distinct binding conformation. *Nature* **519,** 102–105. ISSN: 0028-0836 (2015).

61. Sogabe, Y. *et al.* A crucial role of Cys218 in configuring an unprecedented auto-inhibition form of MAP2K7. *Biochem. Bioph. Res. Comm.* **473,** 476–481. ISSN: 0006-291X (2016).

62. Shraga, A. *et al.* Covalent docking identifies a potent and selective MKK7 inhibitor. *Cell Chem. Bio.* **26,** 98–108.e5. ISSN: 2451-9456 (2019).

63. Schröder, M. *et al.* Catalytic domain plasticity of MKK7 reveals structural mechanisms of allosteric activation and diverse targeting opportunities. *Cell Chem. Bio.* **27,** 1285–1295.e4. ISSN: 2451-9456 (2020).

64. Berman, H. M. *et al.* The protein data bank. eng. *Nucl. Acids Res.* **28.** Publisher: Oxford University Press, 235–242. ISSN: 0305-1048. http://www.rcsb.org/pdb/ (2000).

65. The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucl. Acids Res.* **45,** D158–D169. ISSN: 0305-1048 (2016).

66. The UniProt Consortium. UniProt: a worldwide hub of protein knowledge. *Nucl. Acids Res.* **47,** D506–D515. ISSN: 0305-1048 (2018).

67. Larralde, M. & Zeller, G. PyHMMER: a Python library binding to HMMER for efficient sequence analysis. *Bioinformatics.* ISSN: 1367-4803 (2023).

68. Katoh, K. & Standley, D. M. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molec. Biol. & Evol.* **30,** 772–780. ISSN: 0737-4038 (2013).

69. Van Westen, G. J. *et al.* Benchmarking of protein descriptor sets in proteochemometric modeling (part 1): comparative study of 13 amino acid descriptor sets. eng. *J. Chemoinf.* **5.** Publisher: BioMed Central, 41–41. ISSN: 1758-2946 (2013).

70. Westen, G. J. v. *et al.* Benchmarking of protein descriptor sets in proteochemometric modeling (part 2): modeling performance of 13 amino acid descriptor sets. *J. Chemoinf.* **5,** 42. ISSN: 1758-2946 (Sept. 2013).

71. Kawashima, S. & Kanehisa, M. AAindex: amino acid index database. eng. *Nucl. Acids Res.* **28.** Publisher: Oxford University Press, 374–374. ISSN: 0305-1048 (2000).

72. Dewitte, R. S. & Shakhnovich, E. I. Pseudodihedrals: simplified protein backbone representation with knowledge-based energy. *Prot. Sci.* **3,** 1570–1581. ISSN: 0961-8368 (1994).

73. Chen, T. & Guestrin, C. XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 785–794 (2016).

74. Breiman, L. Random forests. *Mach. Learn.* **45,** 5–32. ISSN: 0885-6125 (2001).

75. Speiser, J. L., Miller, M. E., Tooze, J. & Ip, E. A comparison of random forest variable selection methods for classification prediction modeling. *Exp. Syst. Appl.* **134,** 93–101. ISSN: 0957-4174 (2019).

76. Lundberg, S. & Lee, S.-I. A Unified Approach to Interpreting Model Predictions. *arXiv.* eprint: 1705.07874 (May 2017).

77. Lundberg, S. M., Erion, G. G. & Lee, S.-I. Consistent individualized feature attribution for tree ensembles. *arXiv.* eprint: 1802.03888 (2018).

78. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. *Optuna: a next-generation hyperparameter optimization framework* in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019).

79. Pettersen, E. F. *et al.* UCSF ChimeraX: structure visualization for researchers, educators, and developers. *Prot. Sci.* **30,** 70–82. ISSN: 0961-8368 (2021).

80. Kunzmann, P. & Hamacher, K. Biotite: a unifying open source computational biology framework in Python. *BMC Bioinf.* **19,** 346. ISSN: 1471-2105 (2018).

81. Huerta-Cepas, J., Serra, F. & Bork, P. ETE 3: reconstruction, analysis, and visualization of phylogenomic data. *Molec. Biol. & Evol.* **33,** 1635–1638. ISSN: 0737-4038 (2016).

82. Pandas development team, T. *pandas-dev/pandas: Pandas* version latest (Zenodo, 2020). https://doi.org/10.5281/zenodo.3509134.

83. Pedregosa, F. *et al.* Scikit-Learn: machine learning in Python. *J. Mach. Learn. Res.* **12,** 2825–2830. ISSN: 1532-4435 (2011).

84. Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Meth.* **17,** 261–272 (2020).

85. Waskom, M. L. Seaborn: statistical data visualization. *J. Open Source Soft.* **6,** 3021 (2021).

Chapter 4

Uncovering the DFG-out sequence propensity determinants

Abstract

Protein kinases cycle between active and inactive states, distinguished by a few structural elements, including the activation loop and its conserved DFG motif. In the inactive state, this motif can occupy a DFG-in conformation, a rarer DFG-out conformation, and some even rarer others. The minor, DFG-out conformation is a valuable drug target that can provide selectivity. The sequence determinants that favor DFG-out in specific kinases are not fully understood. To help elucidate them, we assembled and annotated a large collection of kinase domains and applied a machine-learning procedure based on ensembles of decision trees to extract putative determinants. We discovered five within the tyrosine kinase subfamily, which have been proposed earlier to affect activation loop folding and stabilization. Thus, machine learning can shed light on conformational determinants, help understand kinase regulation and improve drug design strategies.

4.1 Introduction

Protein kinases (PKs) play a key role in cellular signaling pathways, where they selectively phosphorylate target proteins to switch on or off protein–protein interactions [1–4]. Kinase mutations can lead to severe disorders like cancer [5–7], driving the development of small molecule inhibitors [8–11]. They cycle between an inactive and an active state through a complex processes involving substrate binding, oligomerization, and autophosphorylation [12, 13]. While the active conformation is highly conserved across PKs, the inactive state displays more variability, often distinguished by a few structural elements, especially the αC-helix and the so-called activation loop [14].

The DFG motif, at the beginning of the activation loop, plays an important role. In the active state, its DFG-Asp side chain points inward, stabilizing the ATP substrate. Its DFG-Phe side chain points outward, occupying a hydrophobic pocket between the αC helix and the β_4 strand (Fig. 4.1). Most PKs retain this "DFG-in" conformation in their inactive state [15, 16]. Others switch to a DFG-out conformation, where the Asp and Phe side chain positions are swapped, impeding substrate binding. Thus, conformational free energy can favor DFG-in or DFG-out, depending on the kinase.

The DFG conformation is also important for inhibitor binding, can modulate it and be modulated. Type-I inhibitors target the DFG-in conformation, competing with substrates. Type-II inhibitors such as Gleevec target the rarer DFG-out conformation, helping to stabilize it and potentially offering a greater selectivity [17]. Characterizing kinase structures and inhibitors has therefore involved DFG motif annotations [15, 16, 18–21]. However, the factors influencing inactive conformations are complex and still not fully understood for every kinase.

Abl and Src, for example, have different DFG propensities, despite their high sequence and structural similarity [22, 23]. Molecular dynamics simulations quantified the in/out free energy difference for Src, and showed that Gleevec (Imatinib) binding stabilizes the DFG-out state [24, 25]. Mutagenesis experiments highlighted several positions that increased Gleevec affinity to Src, however the in/out conformational preferences could not be measured separately [23]. Other positions were shown to stabilize the active conformation [26]. Hari et al. surveyed Map kinases [27], identifying two residues whose

type correlates with the in/out preference: a residue preceding the DFG motif and a "gatekeeper" residue within the activation loop that controls access to the hydrophobic binding pocket [28]. Binding experiments validated these positions but did not explain their precise effect. It remains unclear whether they determine the in/out free energy difference and by what mechanism. Haldane et al. analyzed residue covariation during evolution to predict the propensity for a large set of kinases to assume the DFG-out conformation [29]. However, structural information is often obtained in the presence of ligands that alter the conformational propensities.

The present study reexamines the determinants of DFG inactive preferences using machine learning and comprehensive datasets. In line with previous work, we seek a small number of "discriminating" residues that determine or at least correlate with the intrinsic preference for a DFG-in or DFG-out inactive conformation. We first improved the accuracy and coverage of a recent collection and annotation of kinase structures [16]. The improved collection contained 9920 structural domains, corresponding to 454 unique sequences. Next, we labeled the domain sequences as tyrosine or serine-threonine kinases (TK or STK) and their structures as active/inactive, DFG-in/DFG-out, and apo or holo (a ligand is bound in the inactive site). Finally, we used ensembles of decision trees to re-label the sequences automatically, according to their intrinsic DFG-in/DFG-out structural preferences, and extracted information on which sequence elements correlated with the in/out preferences. These elements may help determine the activation loop conformation and could be targeted in efforts to redesign it.

As a first, test problem, we verified that the decision tree models correctly discriminated between TK and STK sequences, obtaining perfect classifiers for this problem. The classifiers relied on positions with diverging conservation patterns in TKs and STKs, some of which had been described and interpreted earlier. We then applied the method to the DFG-in/DFG-out problem, using kinase sequences for which structural information was available, along with orthologous sequences lacking such information. While the models achieved high classification accuracy, interpreting them within a conservation framework proved challenging. However, for TK sequences, the models achieved near-perfect classification by relying on just a few positions thought to be crucial for stabilizing the DFG-out state. Moreover, the models captured intricate

sequence patterns underlying the DFG propensities. As a further application, we applied the models to 4089 PK sequences found in Swiss-Prot [30] for which structure predictons by Alphafold2 were available [31, 32]. The predicted DFG in/out labels agreed well with the predicted structures.

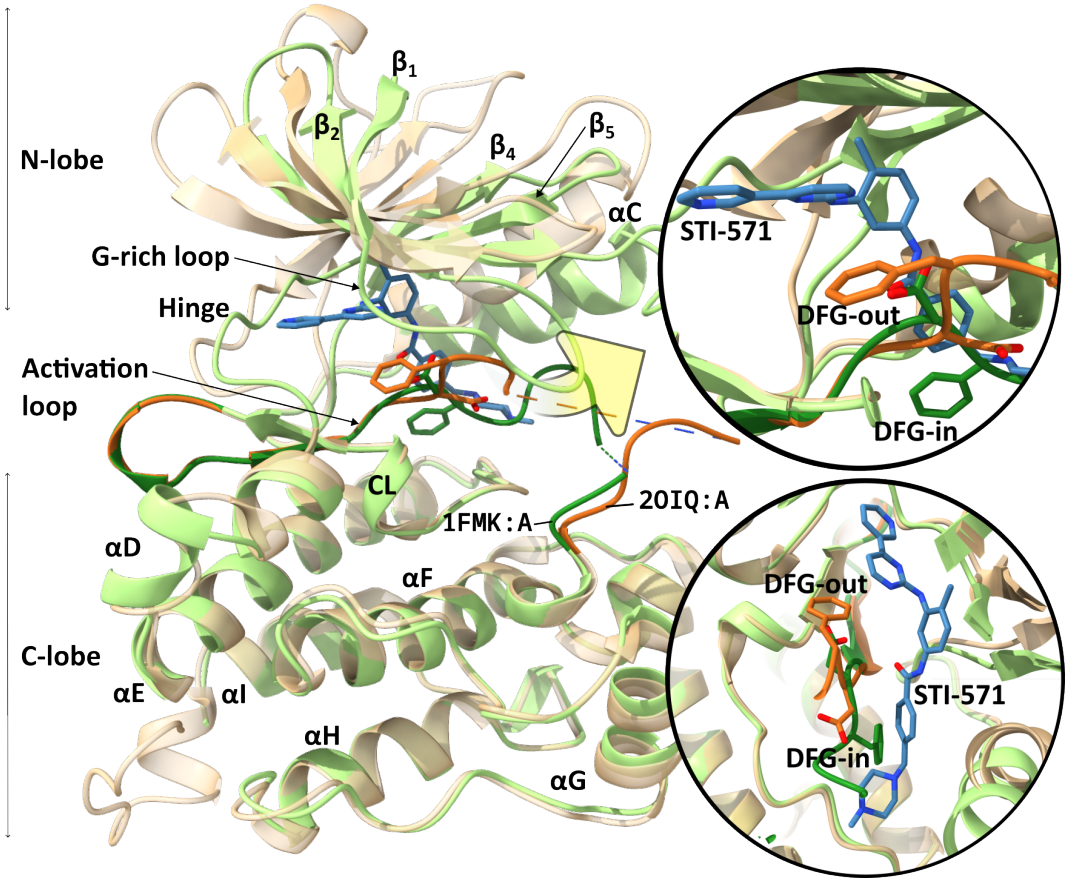

Figure 4.1: Inactive DFG-in (1FMK, green, apo) and DFG-out (2OIQ, wheat/orange, holo) Src tyrosine kinase's conformations. DFG motif resides within the activation loop between N and C lobes, highlighted with darker green and orange colors. In the DFG-in conformation, the DFG-Phe occupies the binding pocket, excluding the possibility of Imatinib's (STI-571) binding. In contrast, in the DFG-out conformation, DFG-Phe is "flipped," freeing the space for type-II inhibitors.

4.2 Materials and Methods

4.2.1 Kinase domain discovery

The SIFTS database (Structure Integration with Function, Taxonomy and Sequence) [33] was queried with a Hidden Markov Model (HMM) from Pfam,

which can be thought of as a sequence profile. We used model PF00069, referred to as $\mathcal{P}_K$, which describes catalytic domains of kinases in general. This model can overlook valid TK domains, especially when there are large inserts. Therefore, in contrast to our previous work [16], we also used the PF07714 profile, referred to as $\mathcal{P}_{TK}$, which describes TK domains. Each Uniprot sequence was aligned to both profiles. We retained hits with a similarity bit score above 50 for either profile, spanning at least 150 residues, where profile and sequence coverage were above 70%. For sequences that had valid hits from both profiles, we created a mapping based on the HMM node numbers that frequently aligned to the same protein sequence position, assigning $\mathcal{P}_K$ numbering to $\mathcal{P}_{TK}$ hits.

Hits from SIFTS have associated PDB structures. Following the transfer of HMM node numbers from UniProt to PDB sequences, we extracted both sequence and structural domains from the PDB entries. For structures with gaps (due to unresolved regions), the PDB sequences were filled in based on the Uniprot sequences (not necessarily fully identical, as mutations were employed in some structural studies). Each domain was characterized by structural variables, described below. For residues or groups with several alternate conformations in a PDB entry, structural analyses were done using the first, main conformation.

4.2.2 Ligand discovery and apo/holo structures

Atoms in each PDB structure were sorted into three main categories: polymer, ligand, and solvent. Residues were polymeric if they appeared as such in the Chemical Component Dictionary (CCD, `ftp://ftp.wwpdb.org/pub/pdb/data/monomers/components.cif`). Solvent residues were defined by our curated list. Remaining residues were annotated as ligands if they met the following criteria: they had at least 5 atoms, their closest distance to a protein atom was below 5 Å, they contacted at least 5 protein atoms and 3 protein residues. This procedure can occasionally result in some atoms not falling in any of the categories. We manually checked such cases for legitimate ligands (e.g., disjoint ligands appearing due to structure inaccuracy).

We split the binding pocket into several subpockets (Table 4.1). For each subpocket, we defined an associated ligand as “bound” based on the number

C of connected residues and the distance D to any subpocket residues (Table 4.1). We considered that a ligand was likely to influence the DFG conformation if it belonged to one of the following subpockets or subpocket pairs: xDFGx, αC Glu, β_3 Lys, αC-β_4, Hinge & Hinge+, Hinge & Gatekeeper. Structures with no ligands or ligands not within these groups were labeled as (effectively) apo.

Table 4.1: Subpocket definitions

Name	$\mathcal{P}_\mathcal{K}$ pos.	Criterion
β_3 Lys	30	$D \leq 7$
αC Glu	48	$C \geq 1$
αC-β_4	52, 60–62	$C \geq 1$
Gatekeper	77	$C \geq 1$
Hinge	78–80	$C \geq 2$
Hinge+	81,82,84	$C \geq 1$
Type2	114,118,121	$C \geq 1$
xDFGx	139-143	$C \geq 1$

4.2.3 Sequence alignment and clustering

We used the CD-HIT clustering algorithm [34] to group domain sequences at three identity levels: 70%, 95%, and 100%. CD-HIT algorithm iteratively constructs cluster via comparing sequence identities in two steps: (1) a fast k-mer-based comparison, and (2) a slower sequence-to-sequence alignment. New clusters are formed by sequences that fail to meet the identity threshold with all existing cluster representatives. The CD-HIT program was executed using the options `-g 1 -n 5`.

For MSAs' construction we used the `mafft` tool (Multiple Alignment using Fast Fourier Transform), providing a good balance between speed and accuracy. It employs the Fast Fourier Transform to rapidly identify homologous regions and then applies progressive alignment to construct the final MSA [35]. For pairwise sequence alignments, we used the `biotite` tool [36] that implements local sequence alignment via dynamic programming (Smith-Waterman algorithm [37]). Finally, we used `PyHMMer` tool [38] for sequence-to-HMM alignments, which is a reimplementation of the well-known HMMer program [39].

4.2.4 Annotating PK families

To annotate PK families, we relied on UniProt family annotations. When these were not available, we employed HMM models from the PANTHER database (v17.0) [40], categorized as: non-receptor serine-threonine kinases (class PC00167; 2015 HMM member families and subfamilies), non-receptor tyrosine kinases (PC00168; 95), receptor serine-threonine kinases (PC00205; 23), and receptor tyrosine kinases (PC00233; 18). These HMM profiles were aligned to each full-length UniProt sequence with a PK domain previously annotated (above). Depending on the best-scoring hit, we assigned a "Tyr" (TK) or "Ser/Thr" (STK) label.

4.2.5 Assigning DFG in/out and active/inactive labels

We used our recent ML models `DFGclassifier` and `KinActive` [16] to categorize each PDB structure as active/inactive and DFG-in/DFG-out. The models rely on variables extracted from each structure, including backbone dihedral angles, pseudo-dihedral angles, interatomic distances, and solvent-accessible surface areas (SASA).

4.2.6 Enriching sequence data with orthologs

Given the limited number of sequences with 3D structures, we collected orthologous sequences lacking an associated X-ray structure, from both PANTHER and UniProt. PANTHER offers an ortholog matching from 143 supported organisms (`http://pantherdb.org/services/oai/pantherdb/ortholog/matchortho`). Using sequence and organism identifiers as inputs, we obtained UniProt accessions of prospective orthologs. Concurrently, the UniRef50 database groups sequences at a 50% identity threshold [41, 42]. Utilizing the `bioservices` Python library [43], we determined the corresponding UniRef50 cluster for each of our sequences. Within these clusters, we identified ortholog candidates by matching gene annotations; specifically, we retained a candidate ortholog if its gene names overlapped with those of the query sequence. We aligned each query-ortholog pair and computed the match identity.

4.2.7 Making and encoding sequence alignments

To create and encode multiple sequence alignments (MSAs), we used the mappings between UniProt sequences and HMM node numbers, detailed above. For a given position profile p, we denote $\mathcal{P}(p)$ the corresponding amino acid and $\mathcal{P}(p,i)$ or $PFP(p,i)$ ($i = 1$–3) one of three numerical variables that correspond to physicochemical properties, or "Protein FingerPrints" associated with $\mathcal{P}(p)$. These variables results from a principal component analysis (PCA) to transform the high-dimensional data amino acid data in the AAindex database [44] into a lower-dimensional space [45, 46]. Combined for a set of sequences, $\mathcal{P}(p,i)$ and $PFP(p,i)$ lead to an MSA and its numerical representation, respectively.

4.2.8 Constructing training datasets

To curate our training datasets, we pursued the following strategy:

1. **Clustering of sequences**: We clustered patched domain sequences, previously predicted as inactive, using a 95% identity threshold.
2. **Defining seed collections**: The resulting clusters were partitioned based on their apo/holo and TK/STK labels. These divisions formed our initial, or "seed" sequence collections.
3. **Ortholog enrichment**: For each seed collection, we added orthologous sequences. We selected (seed, ortholog) pairs that exhibited sequence identity within the [70, 100)% range. For orthologs matching multiple kinase sequences, the ortholog was assigned to the sequence with the highest match score. Seed sequence labels were then transferred to the associated orthologs.
4. **Sequence encoding and dataset creation**: We generated both raw and numerically-encoded MSAs. The raw MSAs aided in examining positions during feature selection (below), whereas the encoded MSAs served as input for the ML models.
5. **Sample weighting**: To counter sequence redundancy, we assigned weights to each sequence, inversely proportional to the size N_i of its 95% identity cluster.

4.2.9 ML models training and feature selection

Let X be an $N \times M$ table of features (an encoded MSA) and Y an array of N response variables (e.g., DFG-in and DFG-out classes encoded as 0 and 1). Let f be an ML model, which relies on a set of trainable parameters θ and a set of non-trainable, or "hyper" parameters λ. The model outputs predictions $Y = f(X; \theta, \lambda)$, which are compared to the known values Y_{exp}. A performance metrics $\rho(Y, Y_{\text{exp}})$ estimates how well the predictions match the known labels. Hyperparameters are chosen to maximize ρ. Our model training comprised the following steps:

1. **Hyperparameter tuning**: Solve an optimization problem $\lambda' = \underset{\lambda}{argmax}\ \rho$.
2. **Performance estimation**: compute $\rho(f(X), Y_{\text{exp}})$.
3. **Feature selection**: Select features relevant to the objective, producing a new dataset X' with fewer variables.
4. **Hyperparameter tuning**: Further optimize hyperparameters using the new dataset X'.
5. **Performance estimation**: compute $\rho(f(X), Y_{\text{exp}})$.
6. **Final model training**: Use X' and the optimized hyperparameters to train the final model.
7. **Feature ranking**: Order the selected MSA columns or features according to their impact on the output ρ.

We used two ML algorithms: XGBoost (eXtreme Gradient Boosting; `XGB`) [47] and Random Forest [48] (`RF`). Both use an ensemble of decision trees trained greedily to partition Y based on the values of X, but have different training paradigms. The first employs the gradient boosting algorithm [49], which sequentially grows a collection of decision trees. Each added tree is trained to better account for errors made by its predecessors through explicitly minimizing a binary cross-entropy loss function. We provided a detailed desciption in our previous work [16]. In contrast, the `RF` trains decision trees independently on random subsamples of X, different for each tree. It outputs a consensus prediction averaged across the ensemble. The training of a single decision tree is exemplified in the Supplementary Material.

We used the Optuna algorithm [50] to find optimal hyperparameters. It is a Bayesian approach that sequentially optimizes an arbitrary objective function, $F_1^{CV}(Y, Y')$ in our case (see below), for a predetermined number of steps

(100 steps). To filter for relevant features, we employed the Boruta algorithm [51], which retains those features that demonstrate consistently higher importance values compared to feaures with randomly scrambled labels. A feature is marked as relevant if its importance surpasses a pre-defined percentile of the scrambled features. We used the 95th percentile for `XGB` and 100th for the `RF`. We used our own Boruta implementation implemented as an open-source `eBoruta` tool. To estimate feature importance, we used Shapley's additive explanations (SHAP), which assess each variable's contribution to the output using a game theory approach [52]. We utilized the `shap` Python library [53] to compute SHAP values and interactions [54]. For further details, see our recent work [16].

4.2.10 Cross-validation

During cross-validation (CV), we split the dataset into N non-overlapping parts ("folds"). Each fold was constructed from sequence clusters obtained at the 95% sequence identity threshold and randomly sampled to reproduce the class ratios of the full dataset (i.e., "balanced" sampling). During each iteration of CV, one fold was left out for testing while the model was trained on the remaining folds. We used ten CV folds in all cases except for hyperparameter tuning, where five folds were used. New folds were generated each time the CV was used to estimate the performance (see below), including hyperparameter tuning.

4.2.11 Performance metrics

The ML models were scored on the accumulated CV predictions using one of three binary performance metrics: precision (prc), recall (rec), and F_1-score. For true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions, the precision is defined as

$$\text{prc} = \frac{TP}{TP + FP} \tag{4.1}$$

Recall, or true positive rate (TPR), or sensitivity, is defined as

$$\text{rec} = \frac{TP}{TP + FN} \tag{4.2}$$

The F_1-score is the harmonic mean of precision and recall:

$$F_1 = 2\frac{\text{prc} * \text{rec}}{\text{prc} + \text{rec}} = \frac{2TP}{2TP + FP + FN} \tag{4.3}$$

4.2.12 Constructing and visualizing phylogenetic trees

We produced MSAs of TK and STK patched domain sequences using `mafft`. MSAs were input to the `iqtree2` tool [55] to build phylogenetic trees. This tool iteratively modifies a phylogenetic tree's structure to maximize its likelihood according to a specific probabilistic model (Pfam.Q in our case) that describes the evolutionary changes. We used the `ete3` toolkit [56] to visualize the results.

4.2.13 Contact Frequency Maps

We used Contact Frequency Maps (CFMs) [57] to compare contact frequencies between structural domains labeled as DFG-in and DFG-out. To construct CFMs, we computed the minimum distance between residue pairs for each of the N $\mathcal{P}_K$ positions. We then employed a 6 Å threshold to denote the presence (1) or absence (0) of a contact, resulting in an N×N matrix for each protein. These were subsequently averaged across the DFG-in and DFG-out domain structures to derive CFM_{in} and CFM_{out}. We then merged these into a single matrix, placing the CFM_{in} data below and CFM_{out} data above the diagonal. During the averaging process, any missing values were omitted, which could possibly highlight infrequent contacts. In contrast, treating these as an absence of contacts would give less weight to unmapped or unstructured regions. We constructed CFMs for TK and STK domains separately.

A contact frequency difference map (CFDM) can emphasize differences in contact frequencies by removing the contacts common to two conformations. To obtain it, we subtracted CFM_{out} from CFM_{in}. $\text{CFDM}_{in/out}$ values range between -1 (exclusive to DFG-out) and 1 (exclusive to DFG-in).

4.3 Results

4.3.1 An updated collection of kinase domains

We updated our recent kinase collection [16] using the latest SIFTS [33] version encompassing 62790 UniProt sequences mapped to 206987 PDB entries (as of 7/2023). We also improved the domain discovery protocol to better account for tyrosine kinase sequences (see Methods). Applying this protocol to all SIFTS sequences resulted in 739 hits using a general PK domain HMM profile $\mathcal{P}_K$ and 699 using a profile $\mathcal{P}_{TK}$ specific for TKs, for a total of 710 PK-domain-containing sequences. These corresponded to 7929 initial PDB entries and 12002 protein chains, 9920 of which contained valid PK domains with at least 100 residues and 80% identity to UniProt sequences. There were a total of 454 unique UniProt entries associated with at least one domain structure.

Using highly accurate ML models developed earlier [16], we annotated the structures as active or inactive and their DFG conformations as DFG-in, DFG-out, or DFG-other. There were 5724/4196 active/inactive structures and 8733/1082/105 in/out/other structures. 8657 (87%) of the domains were ligand-bound, while the other 1263 were apo. There were 9769 ligand–domain pairs in total. 852 domains had multiple contacting ligands, ranging from 2 to 9 (5S7I:B), and 95 ligands were shared between multiple domains (191 in total). We subdivided bound ligands into effectively apo (ligand away froom the activation loop) or holo (8767), relying on rule-based subpocket definitions (see Methods). The ligands spanned the ATP binding pocket region and the DFG-out-specific subpocket housing Type-II inhibitors, which was occupied in 744 domains (Fig. 4.8). Overall, there were 8534 holo and 1386 apo (or effectively apo) domains.

There were 7172 Ser/Thr PKs (STKs), with the CMGC (2082) and CAMK (977) subfamilies being the most abundant, and 1465 lacking a subfamily annotation. The other 2748 entries were tyrosine kinases (TKs). Table 4.2 reports the sequence and structure labels. All active and most of the inactive domains adopted the DFG-in orientation. TKs were usually observed in the inactive state (65%), with or without ligands and STKs less frequently (38% and 33% in the apo and holo subsets). 61% of the TK sequences were found in inactive structures, compared to 44% of STK sequences. This will become crucial for the ML applications below. The fraction of sequences with inactive DFG-out

Table 4.2: Updated collection's labels.

Group	Active	Apo	DFG	Dom.	Struc.	Seq.	Clust.$_{95\%}$	Clust.$_{70\%}$
STk	False	False	in	1438	1162	127	119	95
			other	66	55	16	15	13
			out	472	325	59	58	48
		True	in	330	222	77	76	60
			other	17	16	7	7	6
			out	84	66	28	28	23
	True	False	in	4057	2700	239	216	162
		True	in	707	423	141	126	103
Tk	False	False	in	1125	717	51	45	30
			other	14	12	6	6	6
			out	486	356	42	40	28
		True	in	117	85	28	26	17
			other	7	4	3	3	3
			out	40	29	14	14	13
	True	False	in	876	606	44	38	25
		True	in	83	56	23	21	13

structures also varied with protein family: 22-25% for CAMK, CMGC, STE; 36% for TKL and TKs; 39% for AGC; 65% for NEK (Fig. 4.9a).

To summarize, the structural data, while abundant and fully labeled, did not yield a highly diverse collection of sequences with inactive domain structures. 1789 inactive TK domains produced only 172 sequence clusters at the 95% identity threshold. Furthermore, despite expanding the definition of apo structures to include ligands distant from the DFG-out motif, the proportions of structures and sequences they cover were low (16% of all structures, 39% of all sequences), and lower in the inactive subsets (6% and 20%). This limitation was addressed below by adding orthologous sequences.

4.3.2 Identifying Ser/Thr and Tyr kinase sequences with decision trees

As a test problem, we used ML to identify sequence determinants that distinguish Ser/Thr and Tyr kinases. We considered a dataset of 188 TK and 322 STK sequences. Supplementing them with orthologs led to 1892 TK and 2978 STK sequences. We trained decision tree ensembles using either `RF` and `XGB` models. They achieved perfect sequence partitioning with no errors in cross-

validation test sets. RF and XGB relied on varying sets of residues (features). XGB used 39 input variables (ProtFP PCA components), corresponding to 25 positions in the PK HMM profile.RF considered 752 out of 795 features (252 positions) to be relevant. Additionally, XGB conferred greater importance to a select few positions, in contrast to RF's more uniform distribution (Fig. 4.13). We inspected the top 10 profile positions identified by each model, represented in (Fig. 4.2c) as sequence logos (one each for TKs and STKs). A mapping of $\mathcal{P}_K$ profile numbering to sequences from eight representative PK domains is given in Table 4.10.

Most positions' importance coincided with a pronounced difference in conservation between TKs and STKs. The most prominent positions selected by both models featured residues with contrasting physicochemical properties, such as charge and size. For instance, position 115 contained His in almost all STKs, while TKs harbored Glu, Ser, and Ala. Similarly, Lys at position 125 was predominant in STKs, and Ala in TKs. Similar observations applied to other selected positions. Some of these positions were identified previously as determinants of TK and STK substrate specificity.

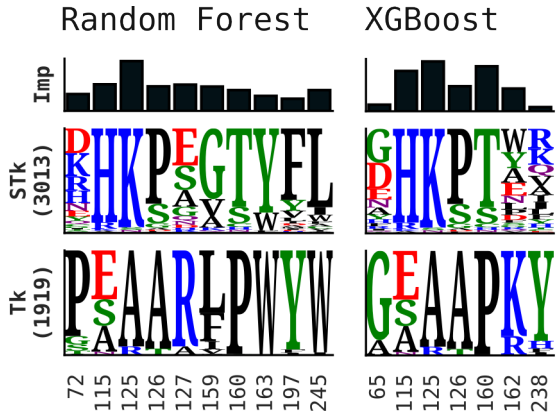

Figure 4.2: A sequence logo of the TkST sequences at top-10 selected positions selected by RF and XGB.

4.3.3 Phylogenetic analysis of the DFG conformation propensity

We now turn to the DFG in/out conformational preferences. To understand how this preference evolved, we selected domain sequences that: (1) were

labeled as inactive, (2) were labeled either DFG-in or DFG-out, and (3) had both DFG-Asp and DFG-Phe present in the structures. This resulted in 3729 structures, and 202 sequence clusters at the 95% threshold (compared to the initial 410 clusters). Clusters with at least one apo structure were labeled "apo," and all others were labeled "holo." We further labeled apo clusters as "in" or "out" if no label conflicts existed and "mixed" in the presence of conflicts (opposing labels within the same sequence cluster). Holo clusters were designated "out-exists" if they contained at least one DFG-out structure and "in-only" otherwise. We selected the most representative sequence from each cluster and constructed phylogenetic trees for each family separately (see Methods).

Within the holo subset, 40 clusters had at least one DFG-out structure, while 51 contained only DFG-in structures. The apo subset consisted of 77 DFG-in, 24 DFG-out, and 10 mixed (or conflicting) clusters. Additional details about these clusters, including discrepancies in labeling are in Supplementary Material (Table 4.5). Potential reasons for these discrepancies include: (1) a naturally low DFG-in/DFG-out free energy difference, (2) allosteric ligand effects, as seen in MK14, (3) interdomain allostery in BRAF, and (4) sequence mutations in proteins like MP2K7 and MKNK2.

Fig. 4.3 maps the cluster propensities onto a phylogenetic tree. Several observations stood out:

- There was no discernible pattern for the propagation of the DFG-out sequence propensity. DFG-out apo structures (within "pure" or mixed clusters) were sporadically distributed across the tree, occasionally neighboring the DFG-in ones. For instance, groups (MP2K7, MP2K6, MP2K4) and SNF1 from *S. Pombe* and *S. cerevisiae*.
- Sequences with a natural DFG-out preference often clustered together. In contrast, sequences such as TIE2, PKNE, VPS15 (DFG-out), and VGFR2 (DFG-in) deviated from this pattern. A manual examination revealed that VGFR2 was in the DFG-other conformation leaning towards DFG-in. Another misannotation concerned MP2K6, which displayed the DFG-other conformation with both DFG-Asp and DFG-Phe oriented "up." For other sequences, these deviations might arise due to limited coverage of the corresponding sequence space regions.
- Every PK family featured clusters with mixed propensities. Except for

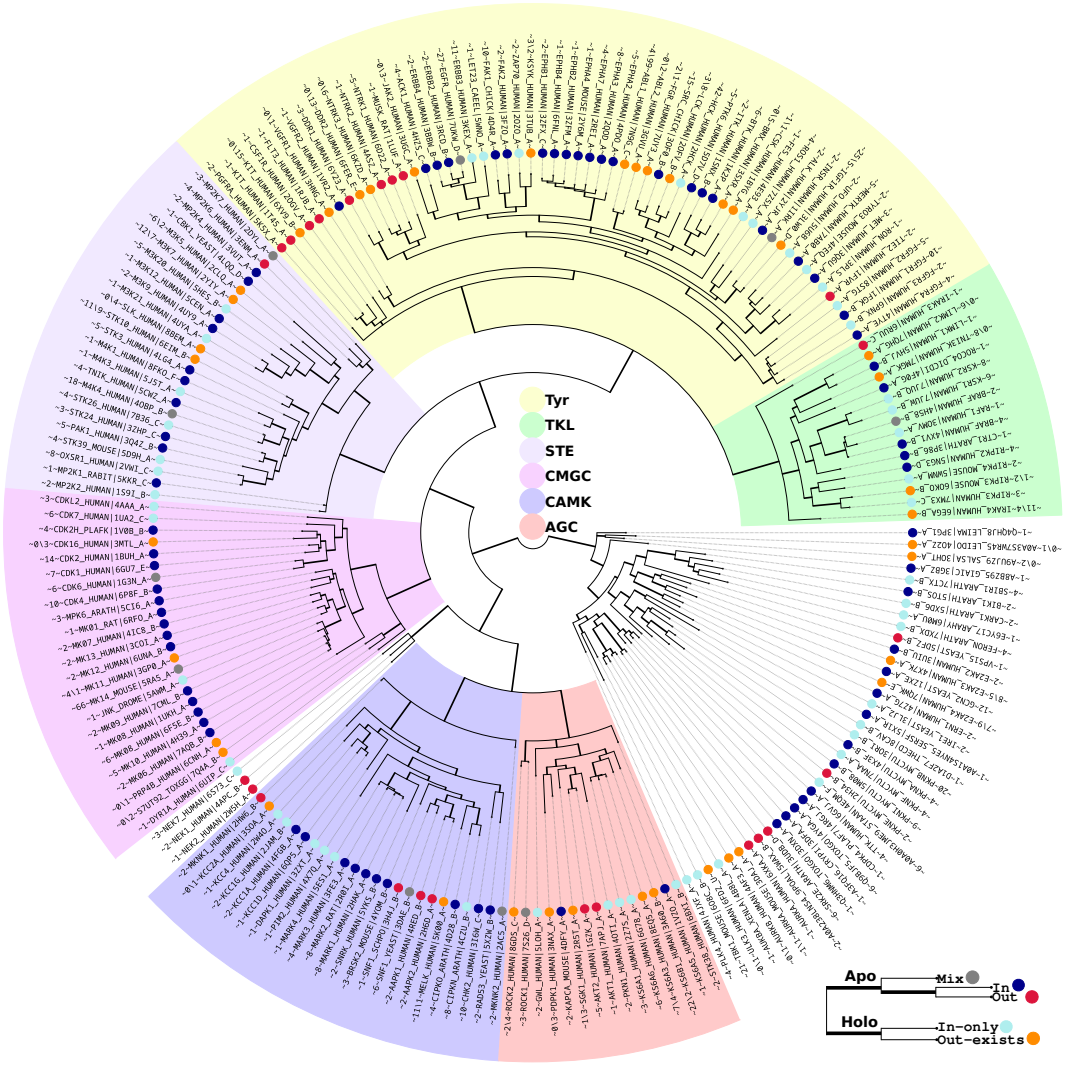

Figure 4.3: A phylogenetic tree of inactive PK domain sequences with mapped DFG conformational propensities. The leaves are annotated radially by the representative structure and sequence followed by the number of domains supporting the label (for "apo" and "in-only holo" clusters) or the DFG-in/DFG-out proportion for clusters with an existing DFG-out structure. Other details are in the main text.

CMGC, every family had at least one DFG-out apo structure. Remarkably, nine TK sequences from groups (1) and (2) displayed a consistent bias towards DFG-out, regardless of an inhibitor's presence. Additionally, closely-related sequences lacking apo structures (DDR2 and NTRK3) were solely linked with the DFG-out conformation.

4.3.4 ML models to identify DFG-in and DFG-out sequences

To identify the residues that correlate with each sequence's DFG in/out preference, we proceeded to train decision tree enesembles. We treated the TK and STK sequences separately, and we excluded any sequences that fell into the "mixed" clusters described above (clusters containing sequences with opposing in/out preferences). Each subset was enriched with orthologues, resulting in 840/490 in/out STKs and 588/412 in/out TKs.

The trained models were highly accurate classifiers, with F_1 scores approaching 1.00. `XGB` made 3 and 7 errors on the TKs and STKs, respectively, while `RF` made 9 and 6. Several misclassifications were shared by the two models. For instance, both algorithms misclassified the TK sequences TIE2 and DDR1 and the STK sequences MKNK1, VPS15, and PKNE. Other errors not common to both models included ALK, SNRK, and KAPCA, errors likely due to insufficient orthologous sequences.

We applied the same methods to the seed datasets (without orthologues). The limited data caused the models' performance to fluctuate during CV, rendering the results largely unreliable. Still, some of the TK positions (77 and 147, described below) were considered for further analysis.

4.3.5 Interpreting models on a sequence level

For each model(`XGB`, `RF`) and dataset (TKs, STKs), we examined the ten positions with the largest contribution to the classification. Fig. 4.4 shows sequence logos at these positions, separately for the DFG-in and DFG-out sequences, while Table 4.3 maps the $\mathcal{P}_\mathcal{K}$ numbering to the UniProt numbering of eight representative domains. For TKs, several positions displayed pronounced differences in conservation (Fig. 4.4a). Specifically, positions 11, 14, 84, 155, and 168 were highly conserved in the DFG-out sequences, but variable in the DFG-in ones. Six positions, 14, 46, 53, 155, 168, and 207, were selected by both `RF` and `XGB`, with a conserved Lys14 ranked as the top feature by both models. STK conservation patterns were less clearcut, with `RF` seemingly capturing the desired signal more effectively than `XGB` (Fig. 4.4b). Positions 69 (β_4-β_5 region), 168, and 177 (C-terminus of the AL) emerged as prominent discriminators. Position 69 featured Asp and Glu in most DFG-out sequences,

while DFG-in sequences mostly excluded such residues, without adhering to any particular preferences. Similarly, DFG-in sequences favored Val at position 168, while the DFG-out ones were heavily skewed towards Lys at position 177. Overall, the most prominent positions described above followed a common pattern: they were constrained to one or, at most, a few residues in one group, whereas the other group did not display any strong preferences and mostly excluded the dominant residues found in the opposing subset.

Table 4.3: Mapping of the $\mathcal{P}_{\mathcal{K}}$ ML-selected positions to representative DFG-in (ABL1–FGFR1) and DFG-out (PGFRA–MUSK) TKs.

Reg.	PK_p	ABL1	SRC	ALK	FGFR1	PGFRA	INSR	DDR1	MUSK
G-loop	11	Q_{252}	C_{280}	A_{1126}	C_{488}	A_{603}	S_{1033}	Q_{620}	A_{584}
β_2	14	E_{255}	E_{283}	E_{1129}	Q_{491}	K_{606}	M_{1036}	E_{623}	R_{587}
β_3–αC	35	D_{276}	G_{303}	V_{1155}	D_{519}	T_{632}	S_{1062}	D_{660}	E_{613}
	37	M_{278}	M_{305}	S_{1157}	T_{521}	R_{634}	S_{1064}	T_{662}	S_{615}
	39	V_{280}	P_{307}	Q_{1159}	K_{523}	S_{636}	R_{1066}	N_{664}	D_{617}
αC	46	L_{284}	L_{311}	L_{1165}	I_{529}	M_{642}	L_{1072}	L_{670}	Q_{623}
	53	K_{291}	K_{318}	S_{1172}	K_{536}	T_{649}	K_{1079}	S_{677}	A_{630}
Hinge	77	T_{315}	T_{341}	L_{1196}	V_{561}	T_{674}	M_{1103}	T_{701}	F_{654}
αD	84	N_{322}	S_{348}	D_{1203}	N_{568}	D_{681}	D_{1110}	D_{708}	D_{661}
AL_N	139	V_{379}	V_{405}	I_{1268}	I_{639}	I_{834}	I_{1175}	I_{782}	I_{740}
	147	L_{387}	L_{413}	D_{1276}	D_{647}	D_{842}	D_{1183}	N_{790}	N_{748}
	150	G_{390}	D_{416}	R_{1279}	H_{650}	H_{845}	E_{1186}	A_{793}	S_{751}
	154	Y_{393}	Y_{419}	G_{1286}	T_{657}	K_{852}	G_{1193}	Q_{800}	D_{758}
	155	T_{394}	T_{420}	G_{1287}	T_{658}	G_{853}	G_{1194}	G_{801}	G_{759}
AL_C	168	S_{410}	A_{436}	A_{1300}	A_{671}	S_{866}	S_{1207}	C_{814}	S_{772}
αG	207					Y_{906}	K_{1247}		Y_{812}
	226	R_{457}	R_{483}	R_{1347}	R_{718}	R_{914}	Y_{1254}	Y_{869}	

To gain further insights, we analyzed MSAs of the selected positions. Our goal was to elucidate some of the misclassifications discussed earlier. The STK sequences presented a challenge: they exhibited a blend of DFG-in/DFG-out characteristics, making them difficult to categorize clearly (Fig. 4.19). On the other hand, two DFG-out TK sequences, DDR1 and TIE2, displayed notable deviations at particular positions that were more conserved among their peers. For instance, DDR1 varied at positions 11, 14, 147, and 154 (QENQ vs. the AKDK), whereas TIE2 diverged at positions 11, 14, 84, 147, and 155 (NQNQT vs. AKDDG) (Fig. 4.20), leading to erroneous labeling of these sequences by the models.

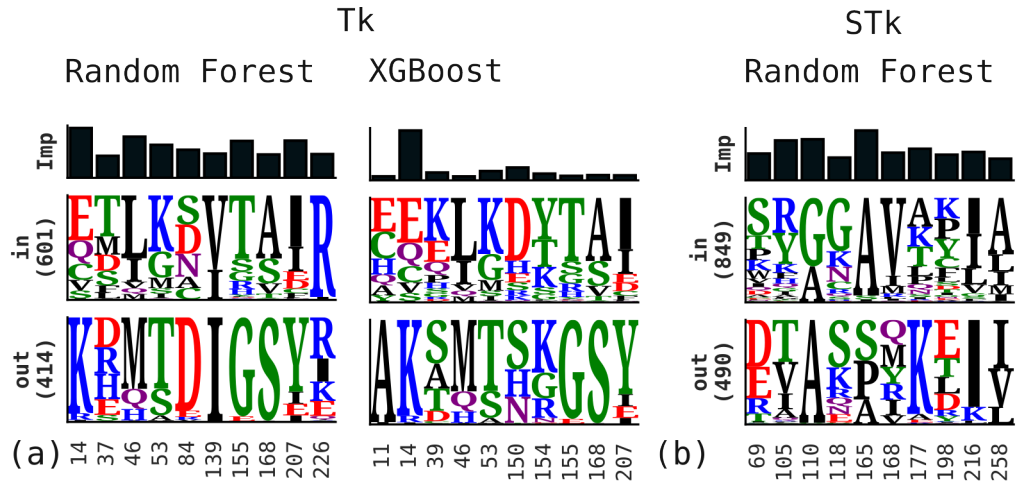

Figure 4.4: A sequence logo of DFG-in/DFG-out datasets' sequences at top-10 selected positions, with **(a)** depicting the Tk subset and **(b)** depicting the `RF` selection of the STk subset.

4.3.6 Structural analysis of the selected positions

To characterize structural changes between the DFG-in and -out conformations, we computed residue contacts in each conformation (see Methods). Contact frequencies were then averaged across DFG-in and DFG-out domain structures, to derive $N \times N$ Contact Frequency Matrices, where N is the number of residues in the kinase profile $\mathcal{P}_K$. The differences in contact frequencies between the in and out conformations are shown in Fig. 4.5. Variations related to AL residues are prominent, particularly the DFG motif (141–143) and nearby residues 144–149. Within TKs, the region 141-149 displayed DFG-out-specific contacts with the N-lobe G-loop and β_2 regions (11–17). These were less frequent or DFG-in-specific in STKs. A similar trend was observed in the (207–226, 199–217) and (207–226, 160–170) quadrants. Other variations included long-range contacts specific to either DFG-out in STKs (e.g., 156–202 and 157–215) or TKs (e.g., 9–147 and 170–215). Table 4.11 reports the top five DFG-in and -out-specific contacts for selected positions. The main contact effects did not overlap strongly with the main ML-derived discriminating positions.

Representative apo DFG-out TK structures are shown in Fig. 4.6. Unlike DFG-in apo structures (Fig. 4.17), the AL was fully resolved in all DFG-out structures. Its DFG-Phe residue was deeply nestled in the ATP-binding

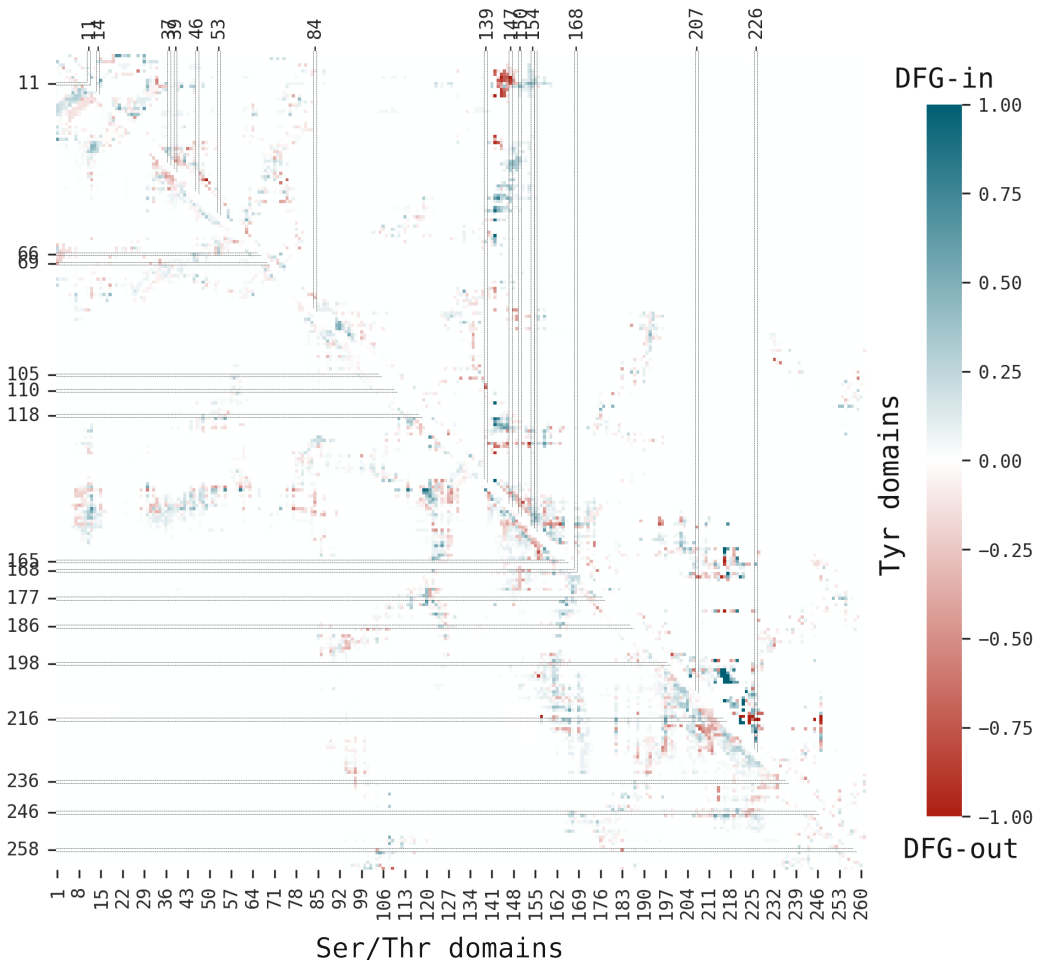

Figure 4.5: Contact frequency difference map (CFDM) for the apo inactive DFG-in and DFG-out TK domains. Missing distances were excluded on averaging the contact number per DFG-in/DFG-out groups. Selected positions are highlighted on the left and at the top.

pocket, while its distal residues were encompassed by the G-rich loop. This is consistent with the contact frequency differences above. AL–β_1 and AL–G-loop contacts were unique to DFG-out in TKs. Ala11 in the G-loop shields hydrophobic residues 144–145 from solvent. The gatekeeper residue 77 may act as a hydrophobic anchor for DFG-Phe, particularly in structures where Phe77 and DFG-Phe face each other (Fig. 4.6b). The top-ranked ML position, Lys14 in DFG-out sequences, was exposed on the surface and lacked specific contacts (Fig. 4.6a-b).

Residues 84, 147, and 150, highly ranked by the ML models, appear to stabilize the AL through polar interactions and assume distinct roles in var-

ious kinase groups. Within class III RTKs (including FLT3, KIT, CSF1R, and PGFRA), Asp147 interacts with Asn/His150 and Arg146 points towards Asn128 and DFG-Asp141 to form DFG-out-specific interactions. In the (NTRK1-2, DDR1, MUSK) group, an α-helix appears to be stabilized by the interaction between Tyr149 and Asp84 (Fig. 4.6b). More distant AL positions 154 and Gly155 (Fig. 4.6d) are in a β-turn present in all DFG-out apo TKs. Finally, residues 35, 37, 39, 47, and 53, within the αC helix (Fig. 4.6c), were found to have no specific contacts exclusive to the DFG-out conformation.

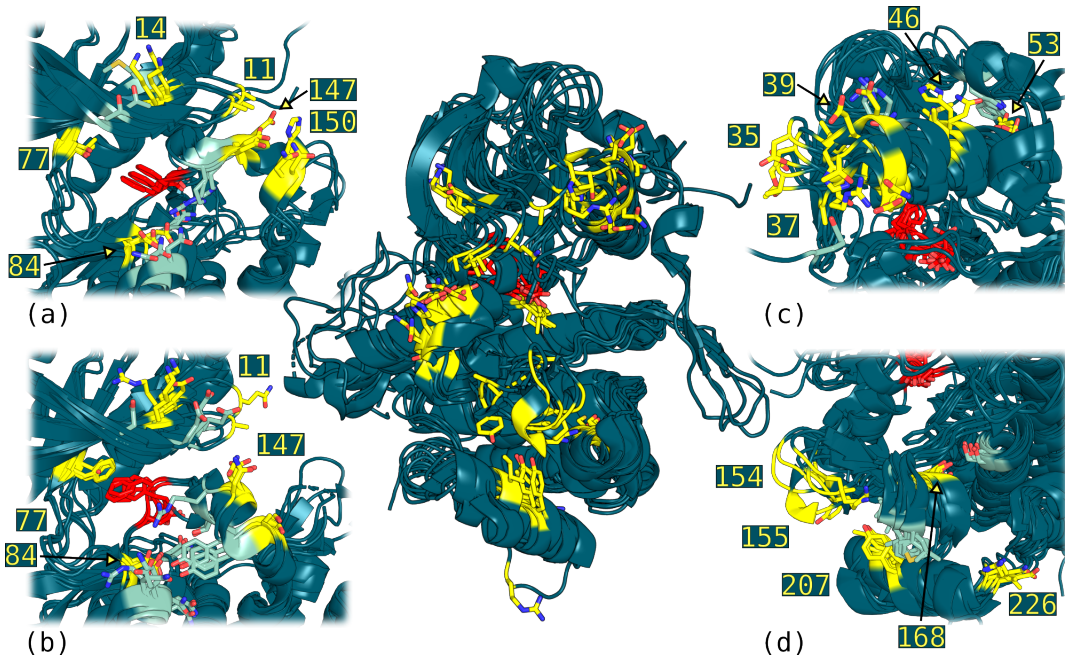

Figure 4.6: DFG-out apo Tk domains. The DFG-Phe and DFG-Asp were depicted in red. The selected positions were highlighted in yellow. The essential contacts were colored cyan. **(a)** The structures were rotated appox. 90° clockwise around the axis between the N and C lobes. PDB codes: 5K5X:A, 1T45:A, and 2OGV:A. **(b)** The structures were oriented as in (a). A subset with a different mechanism of the AL stabilization (see the main text). PDB codes: 6Y23:A, 4ASZ:A, 6D22:A, and 1LUF:A. **(c)** The positions within or proximal to the αC-helix. **(d)** The positions selected within the distal part of the AL and the C-lobe α-helices.

4.3.7 Exloring other labeling strategies

For completeness, we explored other strategies for assigning the DFG labels. To reiterate, when constructing the datasets above, we excluded sequences corresponding to both ligand-bound and "mixed" propensity domains. As

alternative hypotheses, we also made datasets where (1) sequences of ligand-bound structures were allowed, and (2) mixed-propensity entries were retained. In both of these scenarios, the DFG-out labels were assigned to all sequences within 95% identity clusters that contained at least one DFG-out-labeled domain. Although the models attained decent performance, they mostly failed to highlight any discernible conservation signal within these sequence data. This led us to conclude that ligand-bound structures do not possess a generalizable sequence signal susceptible to data mining, while retaining "mixed" clusters introduces noise that complicates sequence mining. Supplementary material provides further information on these additional datasets.

4.3.8 Predicting conformational propensities for SwissProt proteins

Most of the SwissProt [30] kinase sequences have undergone structural prediction with the AlphaFold2 tool [31]. For 3,784 of the Alphafold2 kinase models, active/inactive labels and DFG classifications were computed recently [16]. The character of the DFG conformation in the Alphafold2 models can thus be compared to the present, sequence-based predictions. Only the structures that Alphafold2 modeled in the inactive state were considered, namely 265 TKs and 781 STKs, for a total of 1046 Alphafold2 sequences and structures. 81% (847) were predicted by our ML models to prefer the DFG-in inactive conformation. In contrast, Alphafold2 modeled as many as 92% in the DFG-in conformation. For sequences predicted here to prefer DFG-in, agreement with Alphafold2 was almost perfect (only 16 discrepancies, or 1.9%). But for sequences predicted here to prefer DFG-out, 142 of 199 (71%) were predicted by Alphafold2 to prefer DFG-in. In effect, Alphafold2 behaved almost exactly like a Null model that assigned DFG-in (the predominant conformation in the PDB) to all inactive kinases. The data are summarized in Table 4.9.

AlphaFold2 favoring DFG-in was especially apparent for STK sequences. For 37 STK entries that were part of our initial ("seed") dataset, and thus had manually verified apo-state propensities, all the Alphafold2 structures were incorrectly predicted in the DFG-in orientation. These included two human (NEK1-2) and four non-human enties (SRK2E from *A. thaliana*; SNF1 and VPS15 from yeast, and PKNE from *M. tuberculosis*). Conversely, no mis-

Table 4.4: Numbers of in/out preferences predicted by `XGB` and Alphafold2 (AF2) (TKs/STKs in parentheses).

	AF2 DFG-in	AF2 DFG-out	Fractions
XGB DFG-in	831 (184/647)	16 (8/8)	82%
XGB DFG-out	142 (30/112)	57 (43/14)	18%
Fractions	92%	8%	

matches were found among 11 DFG-in and 4 DFG-out TK "seed" entries. This implies that our sequence-based models likely captured native DFG propensities better than AlphaFold2, which appears to have a strong bias towards the predominant conformation in the PDB, used for its training.

4.4 Discussion

DFG conformational selection involves rather subtle structural changes and modest free energy differences [24, 25] that can be modulated or inverted by ligand binding. In addition, positions whose conservation correlates with DFG preferences may have functional significance, but can also simply mirror broader evolutionary patterns such as ancestral sequences and branching timelines [58]. Thus, sequence determination of DFG-in/DFG-out propensities is not fully resolved and was revisited here using machine learning. Our recent PK collection was updated to include additional PK sequences and structures, of which 4,196 were labeled as inactive. However, only a fraction of sequences were represented by apo structures, where ligand effects are absent. These were augmented by orthologues lacking structural information. As a simpler, test problem, we first used ensembles of decision trees to distinguish TK and STK sequences. The cross-validated models perfectly partitioned the sequences and pinpointed positions with pronounced conservation differences. Top ranking positions 125, 127, 160, and 162 agreed with earlier approaches [58, 59]. This helped validate the approach.

We then analyzed how the DFG-out propensity evoloved throughout the domain phylogeny. Intriguingly, closely related sequences corresponding to apo domains often exhibited contrasting conformational tendencies, with varied potential explanations. Furthermore, phylogenetic placement of native propensities was often sporadic, implying no consistent evolutionary trajec-

tory. This suggested that DFG preferences should be investigated separately for TKs and STKs, so that evolutionary changes related directly to DFG structure are not obscured by other, broader changes.

Training the ML models to partition the DFG-in/DFG-out sequences led to near-perfect classification of both TK and STK sequences. The models trained to classify TK apo sequences detected a clear conservation signal. Two clusters of receptor TKs contrasted with more diverse DFG-in sequences. Most residues specific to DFG-out could be structurally rationalized as reinforcing the DFG-out AL, although some closely-related sequences appeared to use different stabilization mechanisms.

The ML-selected positions were corroborated by several studies. The class III RTKs and DDR1-2, which inherently favor the DFG-out state, were described by Hanson et al. as exceptionally receptive to a range of inhibitors, binding to over a hundred distinct molecules [60]. They proposed that increased AL stability in the DFG-out state might explain this adaptability. They associated this stability to contacts identified here, including a polar network formed by Arg146, Asn128, and DFG-Asp141. Mutations of these residues compromise the DFG-out conformation. Mutations at position 14 in ABL1 are associated with resistance to type II inhibitors in patients with chronic myeloid leukemia, suggesting DFG-out is less accessible [7, 61, 62].

The Asp84Asn mutation in NTRK1 (569 in NTRK1 numbering) abolishes phosphorylation [63, 64]. Beyond this, mutagenesis studies offer no further insights on position 84. In contrast, several DFG-out proteins manifest mutations at the conserved Asp147 that markedly influence their activity. In FLT3 and KIT these mutations produce constitutive activity and increased sensitivity to Gleevec [65]. They may be associated with AL destructuring [66–68]. The mutation Asp147Tyr in NTRK1 (residue 668) results in a less active protein. The same mutation in PGFRA (position 842) increases Gleevec sensitivity, while Asp147Val yields resistance in humans [69, 70]. These substitutions, as MD simulations imply [60, 71], destabilize the DFG-out AL conformation, making it less accessible.

Overall, several discriminating positions appear to the DFG-out conformations. However, despite clear conservation in DFG-out tyrosine kinases, the roles of some others positions remain unclear. Additionally, our results suggest that finding conformational determinants applicable across all kinases

is unlikely. Instead, these determinants appear to rely on sequence context and evolutionary history beyond the DFG motif and its neighborhood. It may be that the DFG propensity is a convergent trait, easily adopted when functionally necessary. Ancestral determinants may also have evolved differently within each PK subfamily, so that distinct positions serve analogous functions. Finally, if the DFG propensity is selectively neutral, its determinants could shift throughout the sequence, settling into consistent patterns only within specific homology-based subgroups.

As the volume of structural data expands and sequence diversity increases, machine learning will continue to help identify structural determinants, as demonstrated here with the DFG motif and TKs. Combined with carefully annotated datasets, our ML models and carefully crafted datasets offer a robust platform to assist in hypothesis testing, mutagenesis and engineering studies, and the development of PK inhibitors.

4.5 Supplementary Materials

4.5.1 Analysing clusters with "mixed" propensity

Table 4.5: Mixed-propensity clusters and their representative domain structures.

		DFG-in		DFG-out	
1	MK14_MOUSE	5RA5:A	62	2NPQ:A	4
2	EGFR_HUMAN	3GOP:A	13	4I21:A,B	14
3	M4K4_HUMAN	4U3Z:A	2	4U3Z:B	16
1	CDK6_HUMAN	1BLX:A	1	1G3N:A,E	5
5	SNF1_YEAST	3HYH:A,B	5	3MN3:A	1
6	ROCK1_HUMAN	7S26:D	1	3TWJ:B,D	2
7	MP2K7_HUMAN	2DYL:A	2	5Y8U:A	1
8	MKNK2_HUMAN	2AC5:A	1	2AC3:A	1
9	INSR_HUMAN	1P14:A	1	1IRK:A	1
10	BRAF_HUMAN	4H58:B	1	4H58:C	1

Table 4.5 lists clusters obtained at the 95% identity threshold and containing both DFG-in and DFG-out domains in the apo state. Table contains a representative structure and a total number of entries supporting the label. We will discuss each of the entries in the presented order.

(1) All four DFG-out MK14 structures (2NPQ:A, 2FSM:X, 4GEO:A, 5N64:A) have a ligand bound to an allosteric pocket within the C-lobe. Diskin et al. characterized this pocket as accommodating lipid molecules [72]. None of the publications accompanying these structures provide an insight into its influence on the DFG motif. Allosteric modulation and sequence mutations likely facilitate the DFG flip in this case.

(2) EGFR has structures supporting all DFG conformations. Perhaps due to a greater AL conformational plasticity [73], DFG transitions in EGFR have a lower energetic barrier, capturing some intermediate conformations.

(3) Two DFG-in M4K4 domains lack an accompanying publication, and no sequence differences could explain the observed discrepancy, leaving the possibility of some external variables influencing the DFG conformation.

(4) DFGin and DFGout structures bind natural protein inhibitors protruding inside the binding pocket and impacting AL conformation. We couldn't automatically account for natural (peptide/lipid) inhibitors in our data processing pipeline: otherwise, such cases should have been excluded.

(5) SNF1 DFG-out structure 3MN3:A contains a large G-loop insert protruding deep into the binding pocket and lacking in 3HYH:A,B domains. The G-loop's hydrophobic nature likely impacts "pulls" the DFG-Phe and reorients the structure accordingly.

(6) Both DFG-in and DFG-out domains come from multimeric structures, which interface via N-lobe regions. Neither accompanying papers nor sequence differences help to explain the observed discrepancies. The number of existing apo structures, unlike the EGFR case, doesn't allow positing natural DFG promiscuity.

(7) The DFG-in 2DYL:A domain of MP2K7 may be unique within available data. It has two AL mutations, including the T291D mutation of the gatekeeper, which, as the title suggests, is activating. No publications accompany the X-ray data. Based on this, we suspect the native sequence likely to have a weak DFG-out propensity.

(8) An Mnk2 DFG-out protein (2AC5:A) is a valuable example of a single mutation within the PK domain sufficient for the label's shift. As indicated by authors in [74], mutating Gly to Asp in the DFG motif pushed the mutated structure into the DFG-out conformation. Interestingly, among the sequences with confirmed preference, a single DFG-out sequence of the structure 2HW6:A

(Q9BUB5, Human MKNK1 protein) had the same non-canonical DFD motif.

(9) The sequence for INSR structure 1IRK:A contains mutation Y984F external to the PK domain and located at a highly conserved position implied in autoinhibition and catalytic activity [12].

(10) Both BRAF chains come from a trimeric complex with a ligand-bound chain 4H58:A. In a publication accompanying a similar structure [75], the authors suggested that it constitutes an example of an anomalous inhibitor-induced dimerization of a cancer-malformed protein. The inhibitor binds to a single protomer, inducing the conformational switch in another one.

4.5.2 Decision tree training example

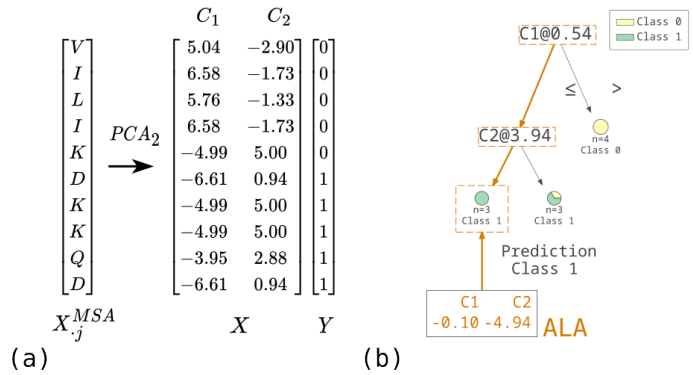

Figure 4.7: **(a)** Each character in an MSA column is embedded into numerical representation using two ProtFP PCA components. **(b)** A trained decision tree. Each DT node has a split threshold to distribute examples into child nodes. The decision path of ALA classification is highlighted in orange.

The Random Forest algorithm comprises an ensemble of decision trees, each trained separately on random subsamples of available instances and features. Each decision tree comprises a collection of internal and terminal (leaf) nodes. The former contains learned threshold values that guide training instances towards the latter. The tree is grown greedily by searching for input variable thresholds that decrease the impurity of the child nodes when compared to the parent node. In classification problems, the impurity is typically measured by the Gini importance or the Shannon entropy. Assuming that a node D comprises N instances, K of which have class 0, and the rest have class 1, its

Gini impurity is defined as

$$G(D) = 1 - \left(\frac{K}{N}\right)^2 - \left(\frac{N-K}{N}\right)^2 = 1 - \sum_i p_i^2 \tag{4.4}$$

where p_i is the fraction of observations that belong to the i-th class. To quantify the goodness of split of a parent node P with N observations into child nodes L and R of sizes N_1 and N_2, the following is used:

$$S = G(P) - \frac{N_1}{N}G(L) - \frac{N_2}{N}G(R) \tag{4.5}$$

To illustrate, consider a sequence corresponding to a single MSA column `VILIKDKKQD` labeled as `0000011111`. Using two ProfFP PCA components, we numerically encoded it into a 10×2 matrix with columns C_1 and C_2 (Fig. 4.7a). Fig. 4.7b shows a tree trained to partition these data and applied to infer a class of a single instance – an Ala residue. The derived thresholds correspond to average values between instances of different classes, e.g., $C_1 = \frac{-3.95+5.04}{2} \approx 0.54$. The tree growth stopped at a depth of two since it wasn't possible to separate Lys residues of classes 0 and 1 that were distributed into the right leaf on the second level.

4.5.3 Dataset naming conventions

The main text emphasizes datasets that yielded more interpretable results. These were constructed from domain sequences, labeled as apo, inactive, DFG-in, or DFG-out, and further divided into Tk and STk subsets. In the Supplementary Material, we refer to these datasets with the abbreviation AAIO (Apo All In or Out) to distinguish them from additional datasets.

In addition to AAIO, we explored two alternative labeling strategies:

- AHAO (Apo/Holo Any Out): This dataset includes sequences from ligand-bound entries. All sequences within 95% identity clusters are labeled as DFG-out if the cluster contains at least one sequence in this state. All others are labeled as DFG-in.
- AAO (Apo Any Out): This dataset excludes sequences corresponding to ligand-bound entries but includes those with conflicting conformational tendencies within 95% identity clusters.

Each of these datasets had two versions:

- A seed version, denoted by a "*" symbol.
- A version enriched with orthologous sequences (no special designation).

Together with the TkST datasets used for testing the methodology, a total of 14 datasets were used, and both `RF` and `XGB` models were applied to each, using the same initial settings, resulting in 28 different models. This additional information is provided for completeness.

4.5.4 Supplementary Tables

Table 4.6: Additional datasets and their performance in ML applications.

Dataset	C_{100} **	C_{95} **	C_{70} **	Pos.	Neg.	Prec/Rec/F_1 Random Forest	Feat/Pos Random Forest	Prec/Rec/F_1 XGBoost	Feat/Pos XGBoost
$AHAO_{STk}$	2978	739	116	1645	1333	0.97/0.97/0.97	724/246	0.94/0.98/0.96	237/133
$AHAO^{*}_{STk}$	322	149	116	181	141	0.85/0.70/0.77	46/39	0.64/0.91/0.75	4/4
$AHAO_{Tk}$	1892	355	39	1661	231	0.99/1.00/0.99	541/196	0.99/1.00/0.99	102/67
$AHAO^{*}_{Tk}$	188	58	39	162	26	-	-	0.86/1.00/0.93	2/2
AAO_{STk}	1401	390	63	562	840	0.98/0.99/0.98	736/251	0.95/0.98/0.97	109/71
AAO^{*}_{STk}	120	79	63	36	84	0.80/0.33/0.47	24/22	0.38/0.56/0.45	4/4
AAO_{Tk}	1115	204	23	527	588	1.00/1.00/1.00	616/218	0.99/1.00/0.99	27/21
AAO^{*}_{Tk}	68	34	23	23	45	0.64/0.30/0.41	41/32	0.73/0.96/0.83	2/2

* Seed dataset.
** C_X columns indicate the number of clusters obtained at the X% identity threshold.

Table 4.7: Final paramerters for the Random Forest models.

DS	Sub.	Seed	bootstrap	ccp	class weight	crit.	max d.	max f.	n est.
AAIO	STk	N	True	0.0	balanced subsample	log loss	12	log2	181
		Y	True	0.19	balanced subsample	entropy	4	log2	252
	Tk	N	True	0.14	None	gini	2	log2	382
		Y	True	0.03	balanced	log loss	11	log2	363
AAO	STk	N	True	0.0	balanced subsample	entropy	15	log2	470
		Y	True	0.05	None	entropy	16	sqrt	402
	Tk	N	True	0.08	balanced	entropy	10	log2	166
		Y	True	0.0	balanced	log loss	15	log2	241
AHAO	STk	N	False	0.01	balanced subsample	entropy	9	sqrt	102
		Y	True	0.03	balanced subsample	entropy	10	sqrt	162
	Tk	N	True	0.0	balanced subsample	entropy	10	None	136
TkST		N	False	0.01	balanced subsample	entropy	8	sqrt	219
		Y	True	0.6	balanced subsample	entropy	10	log2	195

Table 4.8: Final paramerters for the XGBoost models.

Dataset	Subset	Seed	ss. level	ss. tree	γ	lr	max d.	n est.	α	λ	scale pos	subs.
AAIO	STk	N	0.6	0.49	0.62	0.08	12	208	2.12	4.61	2.27	0.4
		Y	0.93	0.62	2.15	0.02	7	277	0.31	9.14	3.1	0.68
	Tk	N	0.49	0.9	9.91	0.19	7	30	6.23	1.51	3.83	0.47
		Y	0.52	0.97	8.36	0.72	16	18	7.33	7.28	9.25	0.66
AAO	STk	N	0.91	0.46	3.43	0.62	5	36	3.75	6.65	4.77	0.99
		Y	0.54	0.6	5.44	0.53	16	50	5.85	6.5	4.0	0.5
	Tk	N	0.97	0.52	4.01	0.56	13	48	4.78	2.42	9.16	0.75
		Y	0.5	0.56	1.98	0.7	7	28	9.44	4.1	8.25	0.73
AHAO	STk	N	0.67	0.97	0.61	0.41	11	27	0.37	3.33	6.94	0.49
		Y	0.54	0.64	5.58	0.43	9	11	4.61	4.81	8.67	0.86
	Tk	N	0.82	0.91	1.52	0.77	13	13	1.79	1.97	8.8	0.75
		Y	0.42	0.6	0.65	0.02	11	35	2.78	4.22	9.42	0.58
TkST		N	0.86	0.9	8.1	0.81	16	17	8.56	9.62	4.44	0.76
		Y	0.76	0.56	5.75	0.4	12	37	1.46	2.04	7.4	0.63

Table 4.9: `XGB` models' predictions (DFG-in/-out) for the SP proteins.

	AHAO		AAO		AAIO		
	in	out	in	out	in	out	Total
Ser/Thr	743	765	989	519	1176	332	1508
CMGC	147	399	490	56	546	0	546
Tyr	77	444	333	188	384	137	521
CAMK	245	218	379	84	408	55	463
AGC	102	265	162	205	263	104	367
STE	106	240	300	46	336	10	346
TKL	62	137	170	29	183	16	199
CK1	58	31	40	49	39	50	89
NEK	24	26	1	49	14	36	50

Table 4.10: Full mapping of the reference $\mathcal{P}_K$ positions to the selected human PK domains. Reg. denotes the begginging of a structural region

Reg.	$\mathcal{P}_K$	ABL1	SRC	PGFRA	INSR	AURKA	BRAF	KAPCA	MKNK1
	1						F_{133}		F_{44}
	2	T_{243}	R_{271}	V_{594}	T_{1024}		E_{134}	T_{458}	E_{45}
	3	M_{244}	L_{272}	L_{595}	L_{1025}	L_{29}	I_{135}	V_{459}	R_{46}
β_1	4	K_{245}	E_{273}	G_{596}	L_{1026}	G_{30}	G_{136}	G_{460}	I_{47}
	5	H_{246}	V_{274}	R_{597}	R_{1027}	D_{31}	R_{137}	Q_{461}	K_{48}
	6	K_{247}	K_{275}	V_{598}	E_{1028}	T_{32}	P_{138}	R_{462}	T_{49}
	7	L_{248}	L_{276}	L_{599}	L_{1029}	L_{33}	L_{139}	I_{463}	L_{50}
	8	G_{249}	G_{277}	G_{600}	G_{1030}	G_{34}	G_{140}	G_{464}	G_{51}
G-loop	9	G_{250}	Q_{278}	S_{601}	Q_{1031}	V_{35}	K_{141}	S_{465}	T_{52}
	10	G_{251}	G_{279}	G_{602}	G_{1032}	G_{36}	G_{142}	G_{466}	G_{53}
	11	Q_{252}	C_{280}	A_{603}	S_{1033}	T_{37}	K_{143}	S_{467}	S_{54}
	12	Y_{253}	F_{281}	F_{604}	F_{1034}	F_{38}	F_{144}	F_{468}	F_{55}
β_2	13	G_{254}	G_{282}	G_{605}	G_{1035}	G_{39}	G_{145}	G_{469}	G_{56}
	14	E_{255}	E_{283}	K_{606}	M_{1036}	K_{40}	N_{146}	T_{470}	R_{57}
	15	V_{256}	V_{284}	V_{607}	V_{1037}	V_{41}	V_{147}	V_{471}	V_{58}
	16	Y_{257}	W_{285}	V_{608}	Y_{1038}	K_{42}	Y_{148}	Y_{472}	M_{59}
	17	E_{258}	M_{286}	E_{609}	E_{1039}	V_{43}	L_{149}	K_{473}	L_{60}
	18	G_{259}	G_{287}	G_{610}	G_{1040}	G_{44}	A_{150}	G_{474}	V_{61}
	19	V_{260}	T_{288}	T_{611}	I_{1046}	K_{45}	R_{151}	K_{475}	K_{62}
	20	W_{261}		R_{617}	K_{1047}	H_{46}	E_{152}		H_{63}
	21	K_{262}	W_{289}	S_{618}	G_{1048}	E_{47}	K_{153}		K_{64}
	22	K_{263}	N_{290}	Q_{619}	E_{1049}	L_{48}	Q_{154}		E_{65}
β_3	23	Y_{264}	G_{291}	P_{620}	A_{1050}	T_{49}	S_{155}	W_{476}	T_{66}
	24	S_{265}	T_{292}	V_{621}	E_{1051}	G_{50}	K_{156}	H_{477}	G_{67}
	25	L_{266}	T_{293}	M_{622}	T_{1052}	H_{51}	F_{157}	G_{478}	N_{68}
	26	T_{267}	R_{294}	K_{623}	R_{1053}	K_{52}	I_{158}	D_{479}	H_{69}
	27	V_{268}	V_{295}	V_{624}	V_{1054}	V_{53}	L_{159}	V_{480}	Y_{70}
	28	A_{269}	A_{296}	A_{625}	A_{1055}	A_{54}	A_{160}	A_{481}	A_{71}
	29	V_{270}	I_{297}	V_{626}	V_{1056}	V_{55}	L_{161}	V_{482}	M_{72}
	30	K_{271}	K_{298}	K_{627}	K_{1057}	K_{56}	K_{162}	K_{483}	K_{73}
	31	T_{272}	T_{299}	M_{628}	T_{1058}	I_{57}	V_{163}	M_{484}	I_{74}
	32	L_{273}	L_{300}	L_{629}	V_{1059}	L_{58}	L_{164}	L_{485}	L_{75}
	33	K_{274}	K_{301}	K_{630}	N_{1060}	N_{59}	F_{165}	N_{486}	D_{76}
αC	34	E_{275}	P_{302}	P_{631}	E_{1061}	R_{60}	K_{166}	V_{487}	K_{77}
	35	D_{276}	G_{303}	T_{632}	S_{1062}	Q_{61}	A_{167}	T_{488}	Q_{78}
	36	T_{277}	T_{304}	A_{633}	A_{1063}	K_{62}	Q_{168}	A_{489}	K_{79}
	37	M_{278}	M_{305}	R_{634}	S_{1064}	I_{63}	L_{169}	P_{490}	V_{80}
	38	E_{279}	S_{306}	S_{635}	L_{1065}	R_{64}	E_{170}	T_{491}	V_{81}
	39	V_{280}	P_{307}	S_{636}	R_{1066}	S_{65}	K_{171}	P_{492}	K_{82}
	40	E_{281}		E_{637}	E_{1067}	L_{66}	A_{172}	Q_{493}	L_{83}
	41			K_{638}	R_{1068}	D_{67}	G_{173}	Q_{494}	K_{84}
	42					V_{68}	V_{174}	L_{495}	Q_{85}
	43		E_{308}	Q_{639}	I_{1069}	G_{70}	H_{176}	Q_{496}	E_{87}
	44	E_{282}	A_{309}	A_{640}	E_{1070}	K_{71}	Q_{177}	A_{497}	H_{88}
	45	F_{283}	F_{310}	L_{641}	F_{1071}	I_{72}	L_{178}	F_{498}	T_{89}
	46	L_{284}	L_{311}	M_{642}	L_{1072}	R_{73}	R_{179}	K_{499}	L_{90}

Continued on the next page...

Table 4.10: Mapping reference positions to the selected human PK domains.

Reg.	$\mathcal{P}_K$	ABL1	SRC	PGFRA	INSR	AURKA	BRAF	KAPCA	MKNK1
	47	K_{285}	Q_{312}	S_{643}	N_{1073}	R_{74}	R_{180}	N_{500}	N_{91}
	48	E_{286}	E_{313}	E_{644}	E_{1074}	E_{75}	E_{181}	E_{501}	E_{92}
	49	A_{287}	A_{314}	L_{645}	A_{1075}	I_{76}	V_{182}	V_{502}	K_{93}
	50	A_{288}	Q_{315}	K_{646}	S_{1076}	Q_{77}	E_{183}	G_{503}	R_{94}
	51	V_{289}	V_{316}	I_{647}	V_{1077}	N_{78}	I_{184}	V_{504}	I_{95}
	52	M_{290}	M_{317}	M_{648}	M_{1078}	L_{79}	Q_{185}	L_{505}	L_{96}
	53	K_{291}	K_{318}	T_{649}	K_{1079}	K_{80}	S_{186}	R_{506}	Q_{97}
	54	E_{292}	K_{319}	H_{650}	G_{1080}	L_{81}	H_{187}	K_{507}	A_{98}
	55	I_{293}	L_{320}	L_{651}	F_{1081}	F_{82}	L_{188}	T_{508}	V_{99}
	56	K_{294}	R_{321}	P_{653}	T_{1082}	R_{83}	R_{189}	R_{509}	N_{100}
	57	H_{295}	H_{322}	H_{654}	C_{1083}	H_{84}	H_{190}	H_{510}	F_{101}
	58	P_{296}	E_{323}	L_{655}	H_{1084}	P_{85}	P_{191}	V_{511}	P_{102}
	59	N_{297}	K_{324}	N_{656}	H_{1085}	H_{86}	N_{192}	N_{512}	F_{103}
	60	L_{298}	L_{325}	I_{657}	V_{1086}	I_{87}	I_{193}	I_{513}	L_{104}
	61	V_{299}	V_{326}	V_{658}	V_{1087}	I_{88}	L_{194}	L_{514}	V_{105}
β_4	62	Q_{300}	Q_{327}	N_{659}	R_{1088}	K_{89}	R_{195}	L_{515}	K_{106}
	63	L_{301}	L_{328}	L_{660}	L_{1089}	L_{90}	L_{196}	F_{516}	L_{107}
	64	L_{302}	Y_{329}	L_{661}	L_{1090}	Y_{91}	Y_{197}	M_{517}	E_{108}
	65	G_{303}	A_{330}	G_{662}	G_{1091}	Q_{92}	G_{198}	G_{518}	F_{109}
	66	V_{304}	V_{331}	A_{663}	V_{1092}	V_{93}	Y_{199}	Y_{519}	S_{110}
	67	C_{305}	V_{332}	C_{664}	V_{1093}	I_{94}	F_{200}	S_{520}	F_{111}
	68	T_{306}	S_{333}	T_{665}	S_{1094}	S_{95}	H_{201}	T_{521}	K_{112}
	69	R_{307}		K_{666}	K_{1095}	T_{96}	D_{202}		D_{113}
β_5	70	E_{308}	E_{334}	S_{667}	G_{1096}	P_{97}	A_{203}	K_{522}	N_{114}
	71	P_{309}	E_{335}	G_{668}	Q_{1097}	S_{98}	T_{204}	P_{523}	S_{115}
	72	P_{310}	P_{336}	P_{669}	P_{1098}	D_{99}	R_{205}	Q_{524}	N_{116}
	73	F_{311}	I_{337}	I_{670}	T_{1099}	I_{100}	V_{206}	L_{525}	L_{117}
	74	Y_{312}	Y_{338}	Y_{671}	L_{1100}	F_{101}	Y_{207}	A_{526}	Y_{118}
	75	I_{313}	I_{339}	I_{672}	V_{1101}	M_{102}	L_{208}	I_{527}	M_{119}
	76	I_{314}	V_{340}	I_{673}	V_{1102}	V_{103}	I_{209}	V_{528}	V_{120}
	77	T_{315}	T_{341}	T_{674}	M_{1103}	M_{104}	L_{210}	T_{529}	M_{121}
	78	E_{316}	E_{342}	E_{675}	E_{1104}	E_{105}	E_{211}	Q_{530}	E_{122}
	79	F_{317}	Y_{343}	Y_{676}	L_{1105}	Y_{106}	Y_{212}	W_{531}	Y_{123}
	80	M_{318}	M_{344}	C_{677}	M_{1106}	V_{107}	A_{213}	C_{532}	V_{124}
	81	T_{319}	S_{345}	F_{678}	A_{1107}	S_{108}	P_{214}	E_{533}	P_{125}
	82	Y_{320}	K_{346}	Y_{679}	H_{1108}	G_{109}	L_{215}	G_{534}	G_{126}
αD	83	G_{321}	G_{347}	G_{680}	G_{1109}	G_{110}	G_{216}	S_{535}	G_{127}
	84	N_{322}	S_{348}	D_{681}	D_{1110}	E_{111}	T_{217}	S_{536}	E_{128}
	85	L_{323}	L_{349}	V_{778}	L_{1111}	L_{112}	V_{218}	L_{537}	M_{129}
	86	L_{324}	L_{350}	K_{779}	K_{1112}	F_{113}	Y_{219}	Y_{538}	F_{130}
	87	D_{325}	D_{351}	N_{780}	S_{1113}	D_{114}	R_{220}	H_{539}	S_{131}
	88	Y_{326}	F_{352}	L_{781}	Y_{1114}	Y_{115}	E_{221}	H_{540}	H_{132}
	89	L_{327}	L_{353}	L_{782}	L_{1115}	I_{116}	L_{222}	L_{541}	L_{133}
	90	R_{328}	K_{354}	S_{783}	R_{1116}	C_{117}	Q_{223}	H_{542}	R_{134}
	91	N_{331}	T_{357}	N_{786}	G_{1127}	K_{118}	K_{224}	I_{544}	R_{135}
	92	R_{332}	G_{358}	S_{787}	R_{1128}	N_{119}	L_{225}	E_{545}	I_{136}
	93	Q_{333}	K_{359}	E_{788}	P_{1129}	G_{120}	S_{226}	T_{546}	G_{137}
	94	E_{334}	Y_{360}	G_{789}	P_{1130}	R_{121}	K_{227}	K_{547}	R_{138}

Continued on the next page...

Table 4.10: Mapping reference positions to the selected human PK domains.

Reg.	$\mathcal{P}_K$	ABL1	SRC	PGFRA	INSR	AURKA	BRAF	KAPCA	MKNK1
αE	95	V_{335}	L_{361}	L_{790}	P_{1131}	L_{122}	F_{228}	F_{548}	F_{139}
	96	N_{336}	R_{362}	T_{791}	T_{1132}	D_{123}	D_{229}	E_{549}	S_{140}
	97	A_{337}	L_{363}	L_{792}	L_{1133}	E_{124}	E_{230}	M_{550}	E_{141}
	98	V_{338}	P_{364}	L_{793}	Q_{1134}	K_{125}	Q_{231}	I_{551}	P_{142}
	99	V_{339}	Q_{365}	D_{794}	E_{1135}	E_{126}	R_{232}	K_{552}	H_{143}
	100	L_{340}	L_{366}	L_{795}	M_{1136}	S_{127}	T_{233}	L_{553}	A_{144}
	101	L_{341}	V_{367}	L_{796}	I_{1137}	R_{128}	A_{234}	I_{554}	R_{145}
	102	Y_{342}	D_{368}	S_{797}	Q_{1138}	R_{129}	T_{235}	D_{555}	F_{146}
	103	M_{343}	M_{369}	F_{798}	M_{1139}	L_{130}	Y_{236}	I_{556}	Y_{147}
	104	A_{344}	A_{370}	T_{799}	A_{1140}	F_{131}	I_{237}	A_{557}	A_{148}
	105	T_{345}	A_{371}	Y_{800}	A_{1141}	Q_{132}	T_{238}	R_{558}	A_{149}
	106	Q_{346}	Q_{372}	Q_{801}	E_{1142}	Q_{133}	E_{239}	Q_{559}	Q_{150}
	107	I_{347}	I_{373}	V_{802}	I_{1143}	I_{134}	L_{240}	T_{560}	I_{151}
	108	S_{348}	A_{374}	A_{803}	A_{1144}	L_{135}	A_{241}	A_{561}	V_{152}
	109	S_{349}	S_{375}	R_{804}	D_{1145}	S_{136}	N_{242}	Q_{562}	L_{153}
	110	A_{350}	G_{376}	G_{805}	G_{1146}	G_{137}	A_{243}	G_{563}	T_{154}
	111	M_{351}	M_{377}	M_{806}	M_{1147}	V_{138}	L_{244}	M_{564}	F_{155}
	112	E_{352}	A_{378}	E_{807}	A_{1148}	D_{139}	S_{245}	D_{565}	E_{156}
	113	Y_{353}	Y_{379}	F_{808}	Y_{1149}	Y_{140}	Y_{246}	Y_{566}	Y_{157}
	114	L_{354}	V_{380}	L_{809}	L_{1150}	C_{141}	C_{247}	L_{567}	L_{158}
	115	E_{355}	E_{381}	A_{810}	N_{1151}	H_{142}	H_{248}	H_{568}	H_{159}
	116	K_{356}	R_{382}	S_{811}	A_{1152}	R_{143}	S_{249}	A_{569}	S_{160}
	117	K_{357}	M_{383}	K_{812}	K_{1153}	H_{144}	K_{250}	K_{570}	L_{161}
	118	N_{358}	N_{384}	N_{813}	K_{1154}	M_{145}	R_{251}	S_{571}	D_{162}
	119	F_{359}	Y_{385}	C_{814}	F_{1155}	V_{146}	V_{252}	I_{572}	L_{163}
	120	I_{360}	V_{386}	V_{815}	V_{1156}	V_{147}	I_{253}	I_{573}	I_{164}
	121	H_{361}	H_{387}	H_{816}	H_{1157}	H_{148}	H_{254}	H_{574}	Y_{165}
	122	R_{362}	R_{388}	R_{817}	R_{1158}	R_{149}	R_{255}	R_{575}	R_{166}
	123	D_{363}	D_{389}	D_{818}	D_{1159}	D_{150}	D_{256}	D_{576}	D_{167}
CL	124	L_{364}	L_{390}	L_{819}	L_{1160}	L_{151}	I_{257}	L_{577}	L_{168}
	125	A_{365}	R_{391}	A_{820}	A_{1161}	K_{152}	K_{258}	K_{578}	K_{169}
	126	A_{366}	A_{392}	A_{821}	A_{1162}	P_{153}	P_{259}	S_{579}	P_{170}
	127	R_{367}	A_{393}	R_{822}	R_{1163}	E_{154}	E_{260}	N_{580}	E_{171}
	128	N_{368}	N_{394}	N_{823}	N_{1164}	N_{155}	N_{261}	N_{581}	N_{172}
	129	C_{369}	I_{395}	V_{824}	C_{1165}	V_{156}	L_{262}	I_{582}	L_{173}
	130	L_{370}	L_{396}	L_{825}	M_{1166}	L_{157}	L_{263}	F_{583}	L_{174}
	131	V_{371}	V_{397}	L_{826}	V_{1167}	L_{158}	L_{264}	L_{584}	I_{175}
	132	G_{372}	G_{398}	A_{827}	A_{1168}	D_{159}	G_{265}	H_{585}	D_{176}
	133	E_{373}	E_{399}	Q_{828}	H_{1169}	A_{160}	S_{266}	E_{586}	Q_{177}
	134	N_{374}	N_{400}	G_{829}	D_{1170}	H_{161}	A_{267}	D_{587}	Q_{178}
	135	H_{375}	L_{401}	K_{830}	F_{1171}	M_{162}	G_{268}	L_{588}	G_{179}
	136	L_{376}	V_{402}	I_{831}	T_{1172}	N_{163}	E_{269}	T_{589}	Y_{180}
	137	V_{377}	C_{403}	V_{832}	V_{1173}	A_{164}	L_{270}	V_{590}	I_{181}
	138	K_{378}	K_{404}	K_{833}	K_{1174}	K_{165}	K_{271}	K_{591}	Q_{182}
AL_N	139	V_{379}	V_{405}	I_{834}	I_{1175}	I_{166}	I_{272}	I_{592}	V_{183}
	140	A_{380}	A_{406}	C_{835}	G_{1176}	A_{167}	A_{273}	G_{593}	T_{184}
	141	D_{381}	D_{407}	D_{836}	D_{1177}	D_{168}	D_{274}	D_{594}	D_{185}
	142	F_{382}	F_{408}	F_{837}	F_{1178}	F_{169}	F_{275}	F_{595}	F_{186}

Continued on the next page...

Table 4.10: Mapping reference positions to the selected human PK domains.

Reg.	$\mathcal{P}_K$	ABL1	SRC	PGFRA	INSR	AURKA	BRAF	KAPCA	MKNK1
	143	G_{383}	G_{409}	G_{838}	G_{1179}	G_{170}	G_{276}	G_{596}	G_{187}
	144	L_{384}	L_{410}	L_{839}	M_{1180}	L_{171}	W_{277}	L_{597}	F_{188}
	145	S_{385}	A_{411}	A_{840}	T_{1181}	S_{172}	S_{278}	A_{598}	A_{189}
	146	R_{386}	R_{412}	R_{841}	R_{1182}	N_{173}	V_{279}	T_{599}	K_{190}
	147	L_{387}	L_{413}	D_{842}	D_{1183}	M_{174}	H_{280}	R_{603}	R_{191}
	148	M_{388}	I_{414}	I_{843}	I_{1184}	M_{175}	A_{281}	W_{604}	V_{192}
	149	T_{389}	E_{415}	M_{844}	Y_{1185}	S_{176}	P_{282}	S_{605}	K_{193}
	150	G_{390}	D_{416}	H_{845}	E_{1186}	D_{177}	S_{283}	G_{606}	G_{194}
	151	D_{391}	N_{417}	D_{846}	T_{1187}	G_{178}	S_{284}	S_{607}	R_{195}
	152	T_{392}	E_{418}	S_{847}	D_{1188}	E_{179}	R_{285}	H_{608}	T_{196}
	153			N_{848}	Y_{1189}	F_{180}		Q_{609}	
	154	Y_{393}	Y_{419}	K_{852}	G_{1193}	L_{181}	R_{286}	F_{610}	
	155	T_{394}	T_{420}	G_{853}	G_{1194}	R_{182}	T_{287}	E_{611}	W_{197}
	156	A_{395}	A_{421}	S_{854}	K_{1195}	T_{183}	T_{288}	Q_{612}	T_{198}
AL_C	157	H_{396}	R_{422}	T_{855}	G_{1196}	S_{184}	L_{289}	L_{613}	L_{199}
	158	A_{397}	Q_{423}	F_{856}	L_{1197}	C_{185}	C_{290}	S_{614}	C_{200}
	159	F_{401}	F_{427}	L_{857}	L_{1198}	G_{186}	G_{291}	G_{615}	G_{201}
	160	P_{402}	P_{428}	P_{858}	P_{1199}	S_{187}	T_{292}	S_{616}	T_{202}
	161	I_{403}	I_{429}	V_{859}	V_{1200}	P_{188}	L_{293}	I_{617}	P_{203}
	162	K_{404}	K_{430}	K_{860}	R_{1201}	N_{189}	D_{294}	L_{618}	E_{204}
	163	W_{405}	W_{431}	W_{861}	W_{1202}	Y_{190}	Y_{295}	W_{619}	Y_{205}
	164	T_{406}	T_{432}	M_{862}	M_{1203}	A_{191}	L_{296}	M_{620}	L_{206}
	165	A_{407}	A_{433}	A_{863}	A_{1204}	A_{192}	P_{297}	A_{621}	A_{207}
	166	P_{408}	P_{434}	P_{864}	P_{1205}	P_{193}	P_{298}	P_{622}	P_{208}
	167	E_{409}	E_{435}	E_{865}	E_{1206}	E_{194}	E_{299}	E_{623}	E_{209}
	168	S_{410}	A_{436}	S_{866}	S_{1207}	V_{195}	M_{300}	V_{624}	I_{210}
	169	L_{411}	A_{437}	I_{867}	L_{1208}	I_{196}	I_{301}	I_{625}	I_{211}
	170	A_{412}	L_{438}	F_{868}	K_{1209}	S_{197}	E_{302}	D_{629}	L_{212}
	171	Y_{413}	Y_{439}	D_{869}	D_{1210}	G_{198}	G_{303}	K_{630}	S_{213}
	172	N_{414}	G_{440}	N_{870}	G_{1211}	R_{199}	R_{304}	N_{631}	K_{214}
	173	K_{415}	R_{441}	L_{871}	V_{1212}	L_{200}	M_{305}	P_{632}	G_{215}
αF	174	F_{416}	F_{442}	Y_{872}	F_{1213}	Y_{201}	H_{306}	Y_{633}	Y_{216}
	175	S_{417}	T_{443}	T_{873}	T_{1214}	G_{203}	D_{307}	S_{634}	N_{217}
	176	I_{418}	I_{444}	T_{874}	T_{1215}	P_{204}	E_{308}	F_{635}	K_{218}
	177	K_{419}	K_{445}	L_{875}	S_{1216}	E_{205}	K_{309}	Q_{636}	A_{219}
	178	S_{420}	S_{446}	S_{876}	S_{1217}	V_{206}	V_{310}	S_{637}	V_{220}
	179	D_{421}	D_{447}	D_{877}	D_{1218}	D_{207}	D_{311}	D_{638}	D_{221}
	180	V_{422}	V_{448}	V_{878}	M_{1219}	I_{208}	L_{312}	V_{639}	W_{222}
	181	W_{423}	W_{449}	W_{879}	W_{1220}	W_{209}	W_{313}	Y_{640}	W_{223}
	182	A_{424}	S_{450}	S_{880}	S_{1221}	S_{210}	S_{314}	A_{641}	A_{224}
	183	F_{425}	F_{451}	Y_{881}	F_{1222}	S_{211}	L_{315}	F_{642}	L_{225}
	184	G_{426}	G_{452}	G_{882}	G_{1223}	G_{212}	G_{316}	G_{643}	G_{226}
	185	V_{427}	I_{453}	I_{883}	V_{1224}	V_{213}	V_{317}	I_{644}	V_{227}
	186	L_{428}	L_{454}	L_{884}	V_{1225}	I_{214}	L_{318}	V_{645}	L_{228}
	187	L_{429}	L_{455}	L_{885}	L_{1226}	L_{215}	C_{319}	L_{646}	I_{229}
	188	W_{430}	T_{456}	W_{886}	W_{1227}	Y_{216}	Y_{320}	Y_{647}	Y_{230}
	189	E_{431}	E_{457}	E_{887}	E_{1228}	A_{217}	E_{321}	E_{648}	E_{231}
	190	I_{432}	L_{458}	I_{888}	I_{1229}	L_{218}	F_{322}	L_{649}	M_{232}

Continued on the next page...

Table 4.10: Mapping reference positions to the selected human PK domains.

Reg.	$\mathcal{P}_K$	ABL1	SRC	PGFRA	INSR	AURKA	BRAF	KAPCA	MKNK1
	191	A_{433}	T_{459}	F_{889}	T_{1230}	L_{219}	L_{323}	M_{650}	A_{233}
	192	Y_{435}	K_{461}	L_{891}	L_{1232}	C_{220}	V_{324}	T_{651}	A_{234}
	193	G_{436}	G_{462}	G_{892}	A_{1233}	G_{221}	G_{325}	G_{652}	G_{235}
FL	194	M_{437}	R_{463}	G_{893}	E_{1234}	T_{222}	K_{326}	Q_{653}	Y_{236}
	195	S_{438}	V_{464}	T_{894}	Q_{1235}	L_{223}	P_{327}	L_{654}	P_{237}
	196	P_{439}	P_{465}	P_{895}	P_{1236}	P_{224}	P_{328}	P_{655}	P_{238}
	197	Y_{440}	Y_{466}	Y_{896}	Y_{1237}	F_{225}	F_{329}	Y_{656}	F_{239}
	198	P_{441}	P_{467}	P_{897}	Q_{1238}	D_{226}	E_{330}	S_{657}	F_{240}
	199	G_{442}	G_{468}	G_{898}	G_{1239}	D_{227}	A_{331}	N_{658}	A_{241}
	200	I_{443}	M_{469}	M_{899}	L_{1240}	D_{228}	N_{332}	I_{659}	D_{242}
	201		V_{470}	M_{900}	S_{1241}	H_{229}	T_{333}	N_{660}	Q_{243}
αG	202		N_{471}	V_{901}	N_{1242}	V_{230}	Y_{334}	N_{661}	P_{244}
	203		R_{472}	D_{902}	E_{1243}	P_{231}	Q_{335}	R_{662}	I_{245}
	204		E_{473}	S_{903}	Q_{1244}	T_{232}	E_{336}	D_{663}	Q_{246}
	205			T_{904}	V_{1245}	L_{233}		Q_{664}	I_{247}
	206			F_{905}	L_{1246}	F_{234}		I_{665}	Y_{248}
	207			Y_{906}	K_{1247}	K_{235}		I_{666}	E_{249}
	208			N_{907}	F_{1248}	K_{236}		F_{667}	K_{250}
	209	D_{444}	V_{474}	K_{908}	V_{1249}	I_{237}	T_{337}	M_{668}	I_{251}
	210	L_{445}	L_{475}	I_{909}	M_{1250}	C_{238}	Y_{338}	V_{669}	V_{252}
	211	S_{446}	D_{476}	K_{910}	D_{1251}	D_{239}	K_{339}	G_{670}	S_{253}
	212	Q_{447}	Q_{477}	S_{911}	G_{1252}	G_{240}	R_{340}	R_{671}	G_{254}
	213		V_{478}	G_{912}				G_{672}	
	214		E_{479}	Y_{913}				Y_{673}	
	215							L_{674}	
	216	V_{448}						S_{675}	
	217	Y_{449}						P_{676}	
	218	E_{450}						D_{677}	
	219	L_{451}						L_{678}	
	220	L_{452}							
	221	E_{453}							
	222						I_{341}		
	223	K_{454}	R_{480}				S_{342}		
	224	D_{455}	G_{481}				R_{343}		
	225	Y_{456}	Y_{482}		G_{1253}		V_{344}		
	226	R_{457}	R_{483}	R_{914}	Y_{1254}	I_{241}	E_{345}		K_{255}
	227	M_{458}	M_{484}	M_{915}	L_{1255}	F_{242}	F_{346}	S_{679}	V_{256}
	228	E_{459}	P_{485}	A_{916}	D_{1256}	Y_{243}	T_{347}	K_{680}	R_{257}
	229	R_{460}	C_{486}	K_{917}	Q_{1257}	T_{244}	F_{348}	V_{681}	F_{258}
	230	P_{461}	P_{487}	P_{918}	P_{1258}	P_{245}	P_{349}	R_{682}	P_{259}
	231	E_{462}	P_{488}	D_{919}	D_{1259}	Q_{246}	D_{350}	S_{683}	S_{260}
	232	G_{463}	E_{489}	H_{920}	N_{1260}	Y_{247}	F_{351}	N_{684}	H_{261}
αH	233	C_{464}	C_{490}	A_{921}	C_{1261}	L_{248}	V_{352}	C_{685}	F_{262}
	234	P_{465}	P_{491}	T_{922}	P_{1262}	N_{249}	T_{353}	P_{686}	S_{263}
	235	E_{466}	E_{492}	S_{923}	E_{1263}	P_{250}	E_{354}	K_{687}	S_{264}
	236	K_{467}	S_{493}	E_{924}	R_{1264}	S_{251}	G_{355}	A_{688}	D_{265}
	237	V_{468}	L_{494}	V_{925}	V_{1265}	V_{252}	A_{356}	M_{689}	L_{266}
	238	Y_{469}	H_{495}	Y_{926}	T_{1266}	I_{253}	R_{357}	K_{690}	K_{267}

Continued on the next page...

Table 4.10: Mapping reference positions to the selected human PK domains.

Reg.	$\mathcal{P}_K$	ABL1	SRC	PGFRA	INSR	AURKA	BRAF	KAPCA	MKNK1
	239	E_{470}	D_{496}	E_{927}	D_{1267}	S_{254}	D_{358}	R_{691}	D_{268}
	240	L_{471}	L_{497}	I_{928}	L_{1268}	L_{255}	L_{359}	L_{692}	L_{269}
	241	M_{472}	M_{498}	M_{929}	M_{1269}	L_{256}	I_{360}	M_{693}	L_{270}
	242	R_{473}	C_{499}	V_{930}	R_{1270}	K_{257}	S_{361}	A_{694}	R_{271}
	243	A_{474}	Q_{500}	K_{931}	M_{1271}	H_{258}	R_{362}	E_{695}	N_{272}
	244	C_{475}	C_{501}	C_{932}	C_{1272}	M_{259}	L_{363}	C_{696}	L_{273}
	245	W_{476}	W_{502}	W_{933}	W_{1273}	L_{260}	L_{364}	L_{697}	L_{274}
	246	Q_{477}	R_{503}	N_{934}	Q_{1274}	Q_{261}	K_{365}	K_{698}	Q_{275}
	247	W_{478}	K_{504}	S_{935}	F_{1275}	V_{262}	H_{366}	K_{699}	V_{276}
	248	N_{479}	E_{505}	E_{936}	N_{1276}	D_{263}	N_{367}	K_{700}	D_{277}
	249	P_{480}	P_{506}	P_{937}	P_{1277}	P_{264}	P_{368}	R_{701}	L_{278}
	250	S_{481}	E_{507}	E_{938}	K_{1278}	M_{265}	S_{369}	D_{702}	T_{279}
	251	D_{482}	E_{508}	K_{939}	M_{1279}	K_{266}	Q_{370}	E_{703}	K_{280}
αI	252	R_{483}	R_{509}	R_{940}	R_{1280}	R_{267}	R_{371}	R_{704}	R_{281}
	253	P_{484}	P_{510}	P_{941}	P_{1281}	A_{268}	P_{372}	P_{705}	F_{282}
	254	S_{485}	T_{511}	S_{942}	T_{1282}	T_{269}	M_{373}	L_{706}	G_{283}
	255	F_{486}	F_{512}	F_{943}	F_{1283}	I_{270}	L_{374}	F_{707}	V_{289}
	256	A_{487}	E_{513}	Y_{944}	L_{1284}	K_{271}	R_{375}	P_{708}	N_{290}
	257	E_{488}	Y_{514}	H_{945}	E_{1285}	D_{272}	E_{376}	Q_{709}	D_{291}
	258	I_{489}	L_{515}	L_{946}	I_{1286}	I_{273}	V_{377}	I_{710}	I_{292}
	259	H_{490}	Q_{516}	S_{947}	V_{1287}	R_{274}	L_{378}	L_{711}	K_{293}
	260	Q_{491}	A_{517}	E_{948}	N_{1288}	E_{275}	E_{379}	A_{712}	N_{294}
	261	A_{492}	F_{518}		L_{1289}	H_{276}	H_{380}		H_{295}
	262					E_{277}	P_{381}		K_{296}
	263					W_{278}	W_{382}		W_{297}
	264					F_{279}	I_{383}		F_{298}

Table 4.11: Top-5 interacting position pairs per model.

Dataset	Subset	Model	P1	P2	Int.
TkST		RF	115	125	0.59
			125	162	0.65
				178	-0.56
			159	163	-0.65
			160	163	-0.7
		XGB	65	160	64.89
			115	125	-39.59
			125	160	-73.41
				162	27.84
			126	238	31.89
AHAO	STk	RF	55	180	-0.21
			69	202	-0.19
			132	180	0.17
			199	202	0.16
			201	232	-0.19
		XGB	35	232	-16.92
			37	132	19.91
			44	73	15.82
			55	201	-12.18
			73	132	19.15
	Tk	RF	64	98	1.27
			70	77	8.93
			87	95	1.21
			95	104	1.08
				149	2.5
		XGB	22	192	36.67
			23	77	-25.05
			25	46	-21.03
			155	256	17.08
			156	261	-26.33
AAO	STk	RF	69	165	-0.13
			105	110	-0.1
			108	251	-0.09
			110	165	-0.11
			150	165	-0.1
		XGB	66	165	55.15
			81	165	-12.37
			165	192	-20.47
			168	173	-14.6
				177	19.62
	Tk	RF	14	84	0.2
			35	139	-0.2
			53	155	0.31
			82	168	0.32
			153	155	-0.2
		XGB	35	150	32.48
				168	25.82

Continued on the next page...

Table 4.11: Top-5 interacting position pairs per model.

Dataset	Subset	Model	P1	P2	Int.
			139	150	35.18
				168	40.08
			150	168	22.45
AAIO	STk	RF	69	110	-0.21
				165	-0.2
			110	186	-0.24
			177	255	-0.2
			208	220	-0.2
		XGB	11	110	7.27
			105	118	6.03
				177	5.69
				251	5.26
			110	118	5.86
	Tk	RF	14	113	0.09
				207	0.1
			35	53	-0.1
			46	84	-0.14
			53	259	-0.09
		XGB	14	77	12.13
				207	-11.25
			39	101	14.54
				150	7.19
			150	154	7.37

Table 4.12: Top CFDM conctacts for manually* and algorithmically selected positions. Only pairs with ≥ 0.3 frequence difference are displayed.

Subset	P.	P. cont.	Freq. diff.
STk	11	151	0.37
		150	0.32
		149	0.31
	66	45	0.33
	69	1	-0.38
	105	263	-0.36
		259	0.54
	115*	120	0.33
	139*	125	-0.5
		123	-0.44
		142	0.66
		108	0.44
		106	0.32
	165	219	-0.37
		179	0.56
		216	0.31
	168	153	-0.41
		151	-0.4
		122	0.37
	198	218	-0.57
	202*	156	-0.85
		218	-0.55
		211	0.35
		207	0.31
	215*	157	-1.0
		199	-0.36
		161	-0.35
		221	-0.33
		206	0.4
		197	0.33
	216	197	0.55
		181	0.52
		247	0.48
		221	0.44
		164	0.43
	246	220	0.3
	258	183	-0.57
		104	-0.35
Tk	11	147	-0.79
		144	-0.52
		33	-0.5
		143	-0.45
		146	-0.39
		153	0.5
		154	0.43
		36	0.33

Continued on the next page...

Table 4.12: Top CFDM contacts.

Subset	P.	P. cont.	Freq. diff.
		152	0.3
	14	144	-0.88
		145	-0.75
	37	147	0.44
		148	0.41
		149	0.31
	39	36	-0.43
		147	0.3
	46	74	0.33
	84	149	-0.45
		148	-0.45
		142	-0.36
		80	-0.31
	139	109	-0.69
		108	-0.44
		142	0.98
	142*	15	-1.0
		30	-0.94
		130	-0.91
		28	-0.9
		7	-0.75
		114	1.0
		139	0.98
		60	0.98
		119	0.91
		121	0.91
	144*	9	-1.0
		10	-0.88
		13	-0.88
		14	-0.88
		15	-0.88
		121	0.59
		122	0.46
		52	0.37
		75	0.35
		148	0.32
	145*	9	-0.75
		10	-0.75
		14	-0.75
		15	-0.75
		8	-0.67
		122	0.61
		12	0.38
		47	0.37
		121	0.35
	147	9	-1.0
		10	-0.88
		11	-0.79

Continued on the next page...

Table 4.12: Top CFDM contacts.

Subset	P.	P. cont.	Freq. diff.
		143	-0.5
		150	-0.34
		37	0.44
		45	0.39
		36	0.35
		48	0.35
		44	0.3
	150	147	-0.34
		145	-0.31
		12	0.44
		154	0.37
		155	0.33
		36	0.31
	154	207	-0.5
		161	-0.32
		160	0.58
		11	0.43
		150	0.37
		152	0.32
	155	158	0.62
		151	0.46
		150	0.33
		152	0.31
	168	175	-0.51
		153	0.42
	198*	188	-0.31
	207	170	-0.65
		154	-0.5
		223	1.0
	215*	225	-1.0
		224	-1.0
		222	-1.0
		221	-1.0
		247	-1.0
		200	1.0
		202	1.0
		201	1.0
	216*	181	-1.0
		226	-1.0
		224	-1.0
		165	-1.0
		227	-1.0
		200	1.0
		221	1.0
		161	1.0
		162	1.0
		210	1.0
	226	216	-1.0

Continued on the next page...

Table 4.12: Top CFDM contacts.

Subset	P.	P. cont.	Freq. diff.
		220	1.0
		222	0.8
		219	0.67
		223	0.57
		166	0.49

4.5.5 Supplementary figures

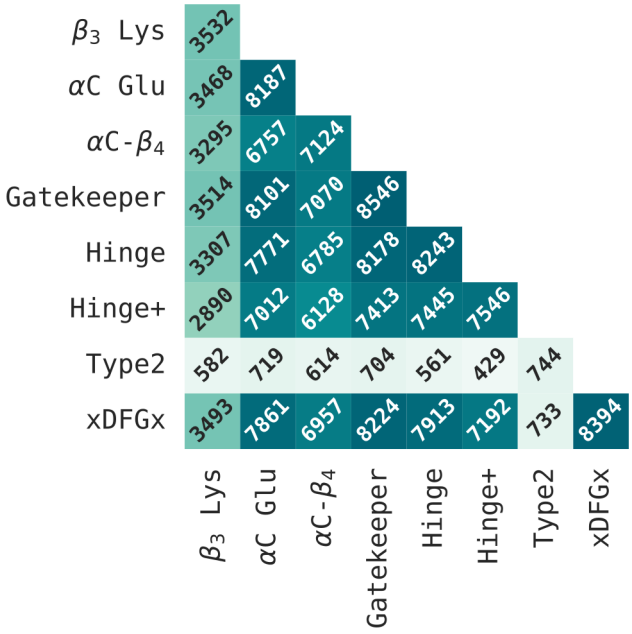

Figure 4.8: Subpockets' coverage in ligand-bound structures.

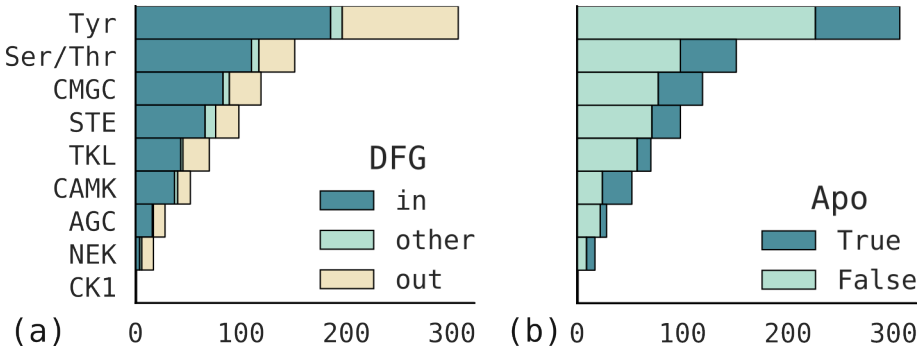

Figure 4.9: Counts of unique inactive sequences grouped by the PK family annotation. (a) Unique sequences' counts colored by the predicted DFG label. (b) Unique sequences' counts colored by being apo or holo concerning the DFG motif (see the main text for details).

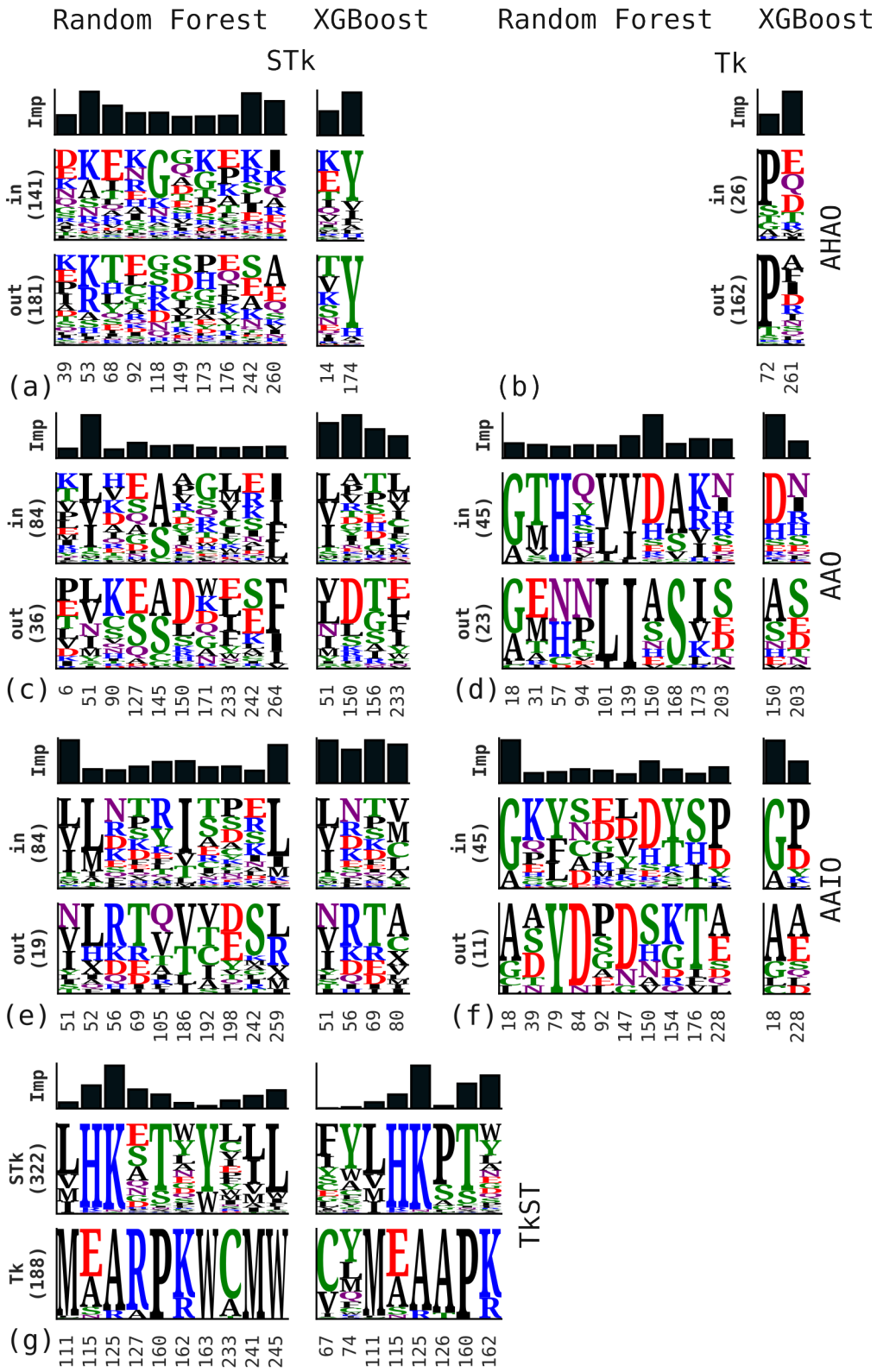

Figure 4.10: Feature selection results for "seed" datasets, i.e., constructed from sequences from our data collection and missing orthologs.

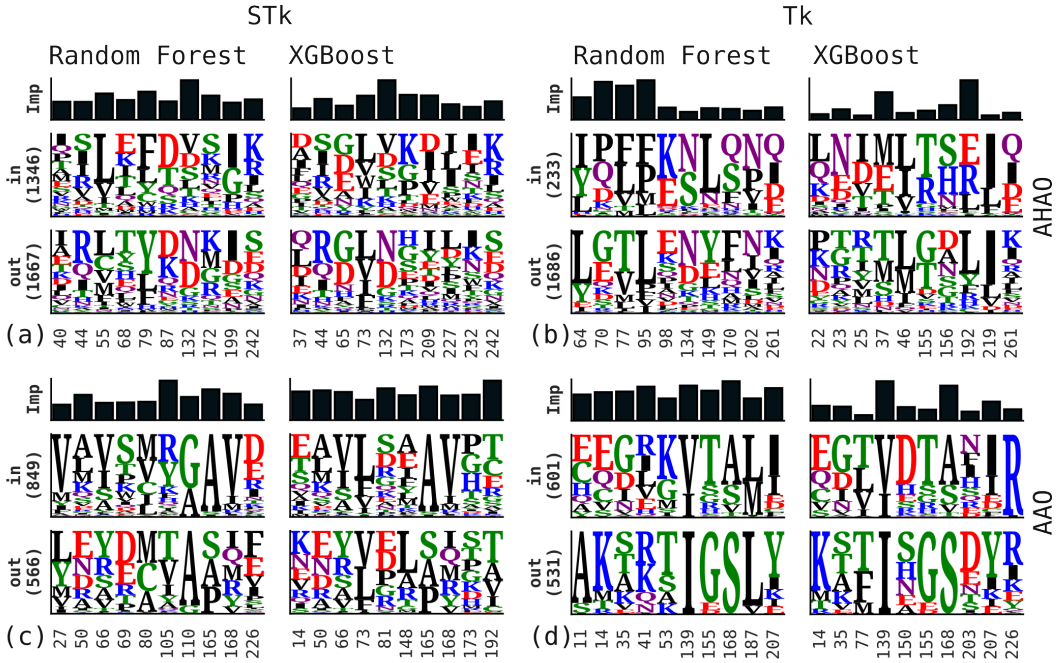

Figure 4.11: Feature selection results for additional datasets: **(a-b)** Apo/Holo Any Out (AHAO) and **(c-d)** Apo Any Out (AAO).

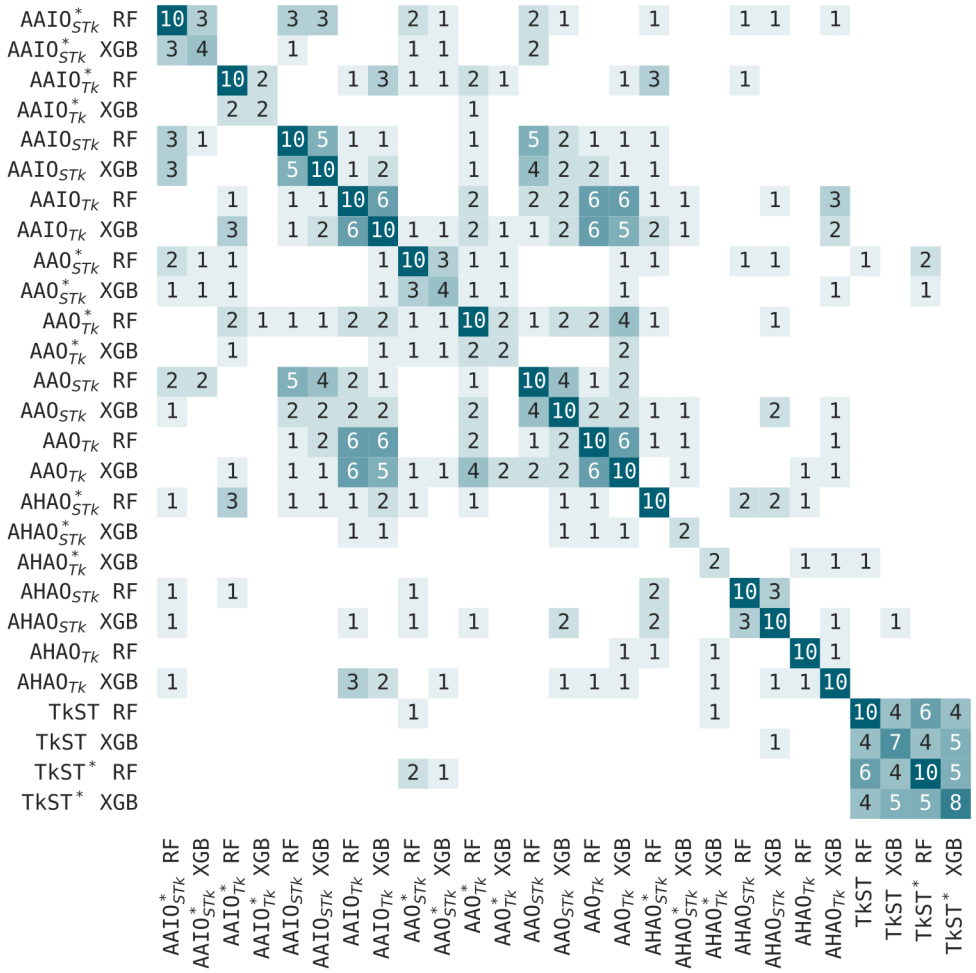

Figure 4.12: Overlap sizes of top-10 selected positions per model.

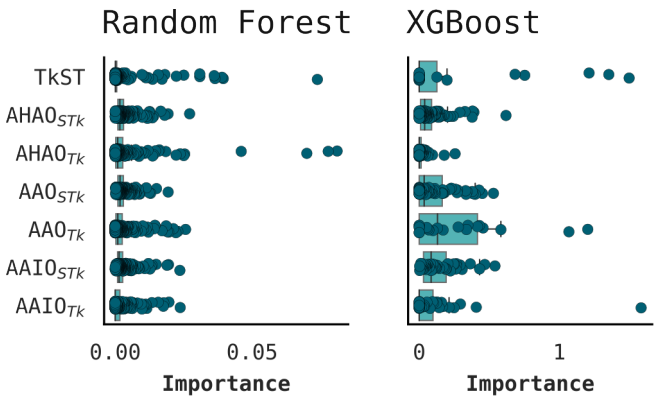

Figure 4.13: Importance values' distributions across ML model features.

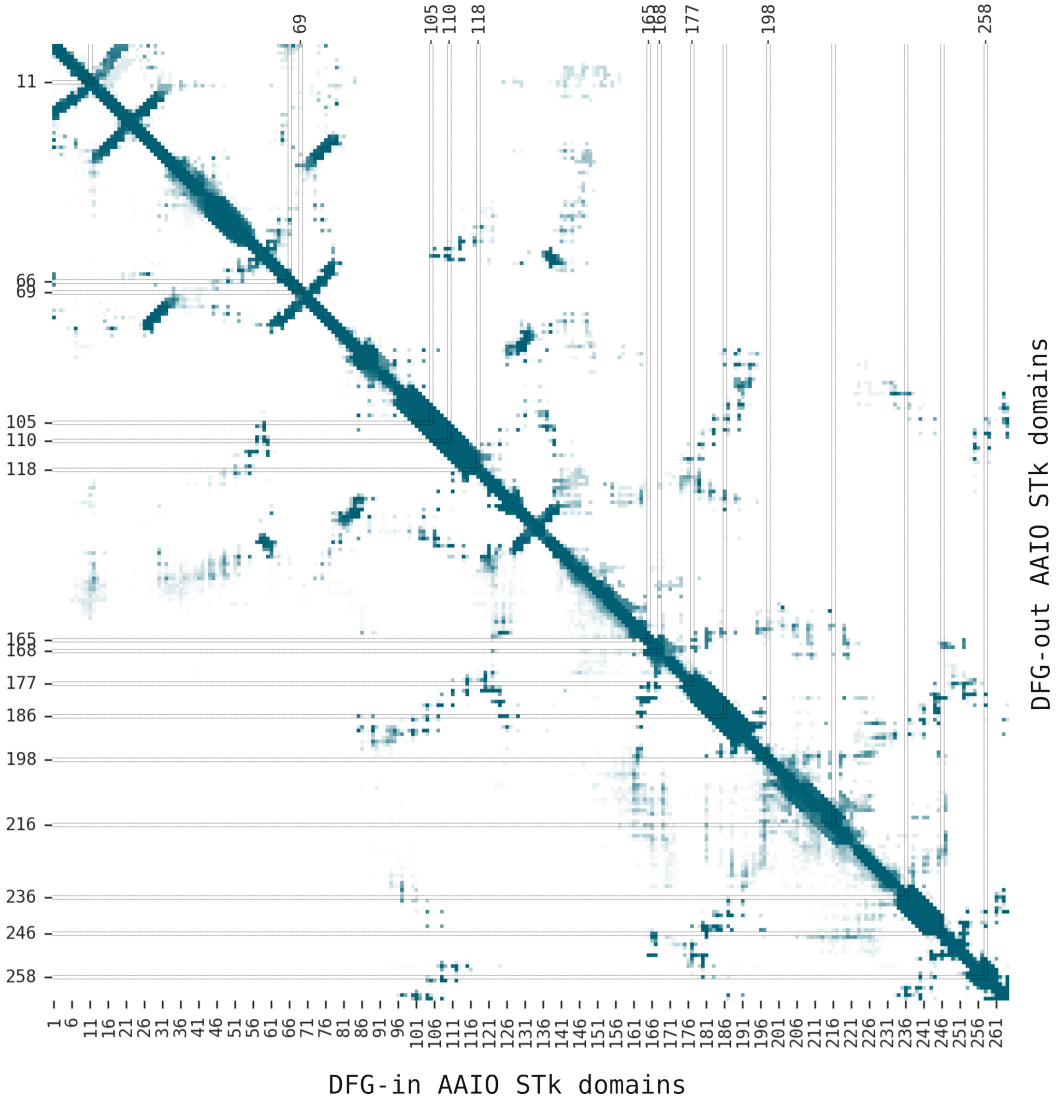

Figure 4.14: Contact frequency map for the AAIO$_{STk}$ structures. Missing residue pairs were excluded during averaging across the structures. Selected positions are highlighted on the left and at the top. CFM for DFG-in and DFG-out structures occupy the lower and upper triangles, respectively.

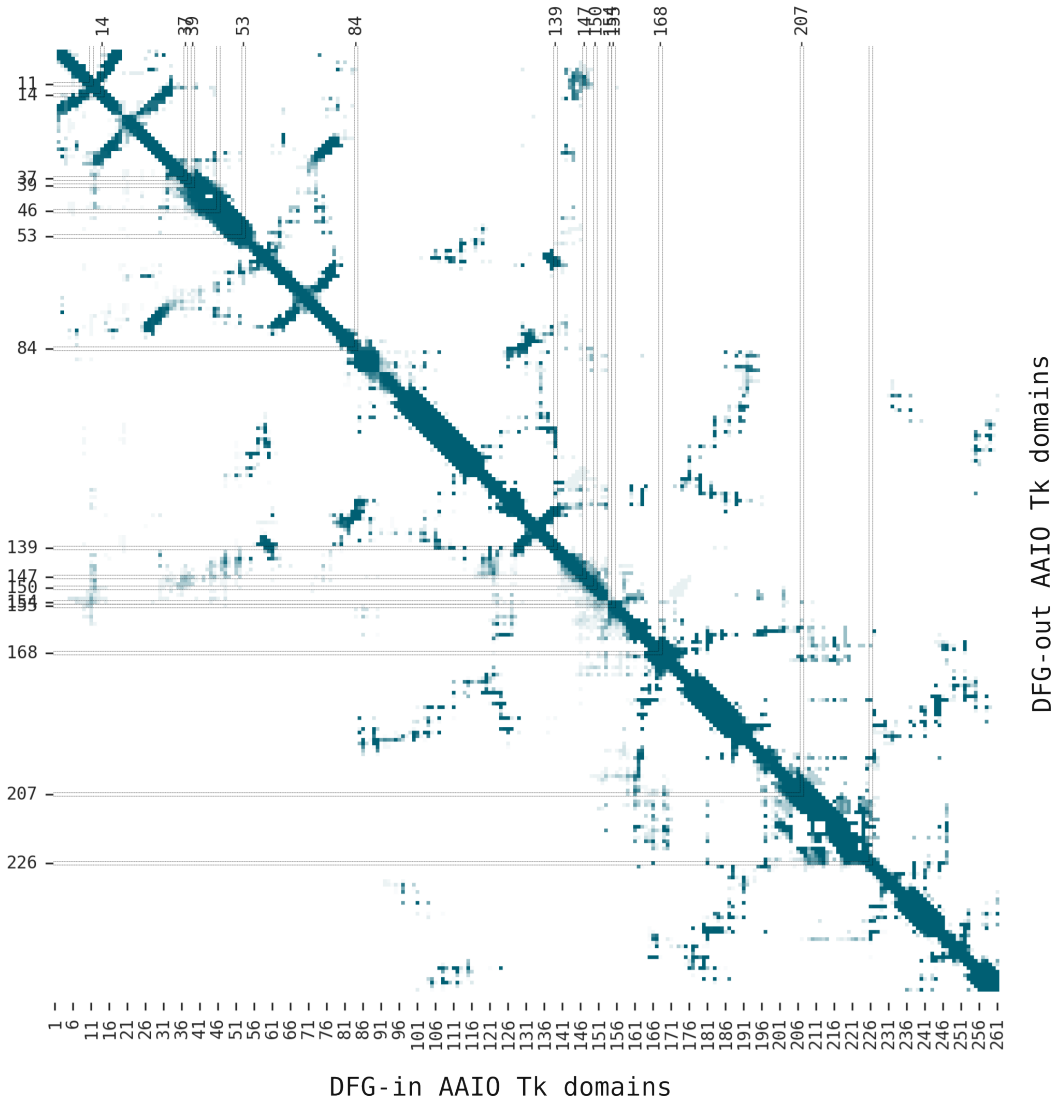

Figure 4.15: Contact frequency map for the AAIO$_{Tk}$ structures. See Fig. 4.14 for further details.

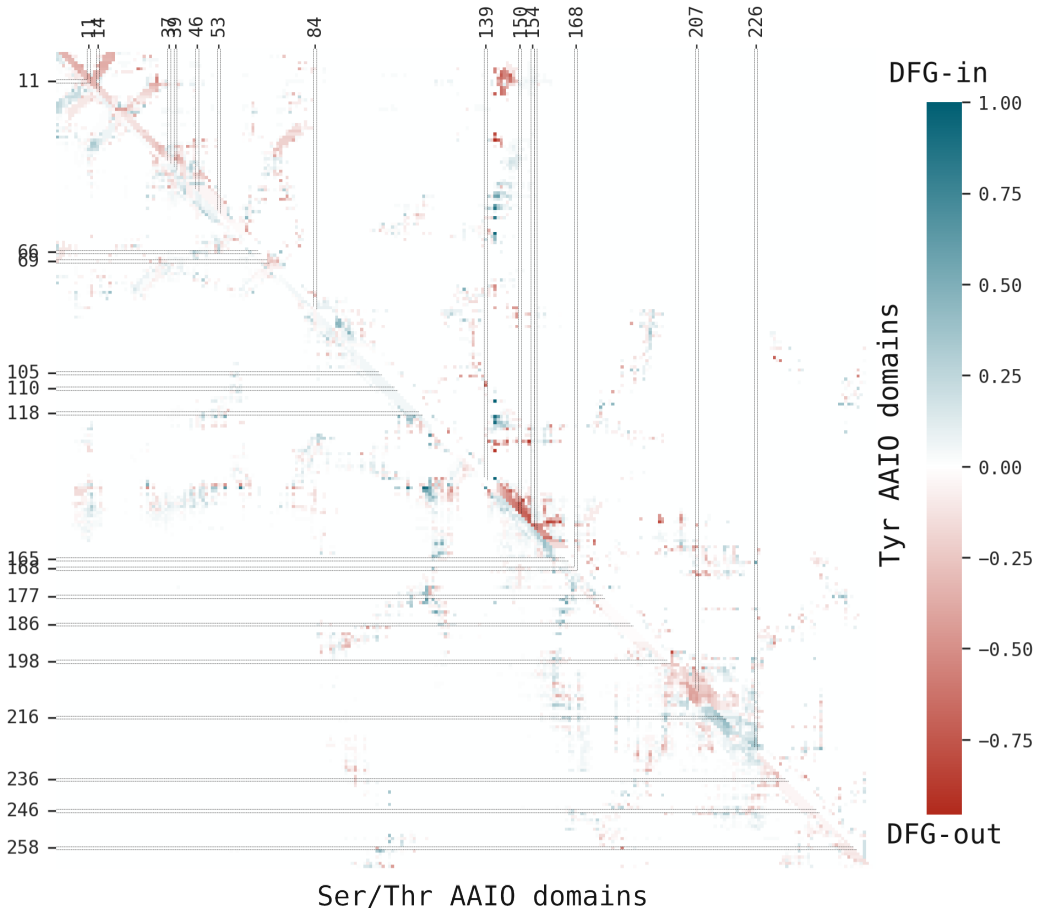

Figure 4.16: Contact frequency difference map (CFDM) for the AAIO structures. The absence of contacting residues was treated as the absence of contact when averaging across identically labeled domains. Selected positions are highlighted on the left and at the top.

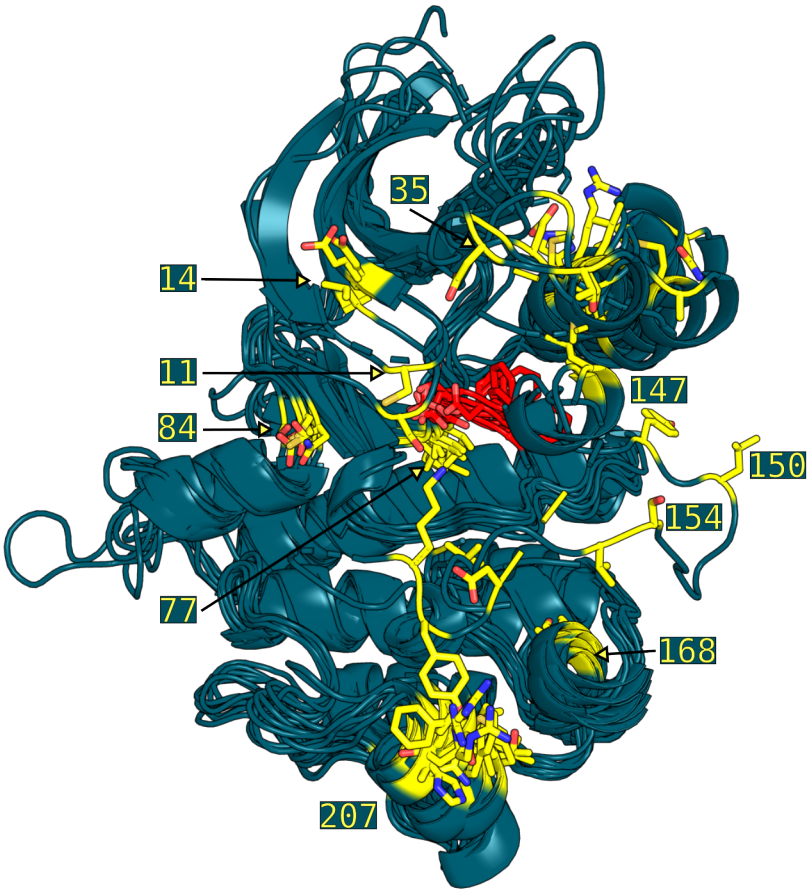

Figure 4.17: Representative DFG-in apo TK domains with marked selected positions. AL is disordered in most structures, including positions distal to DFG-Phe 147, 150, 154, and 155. In contrast to the DFG-out domains (Fig. 4.6) these residues are placed "below" the αC-helix. PDB codes: 4HZS:C, 3BBW:B, 3RCD:B, 4PDO:A, 3OF0:B, 1K2P:A, 2YJR:A, 1FGK:B, and 3Q6U:A

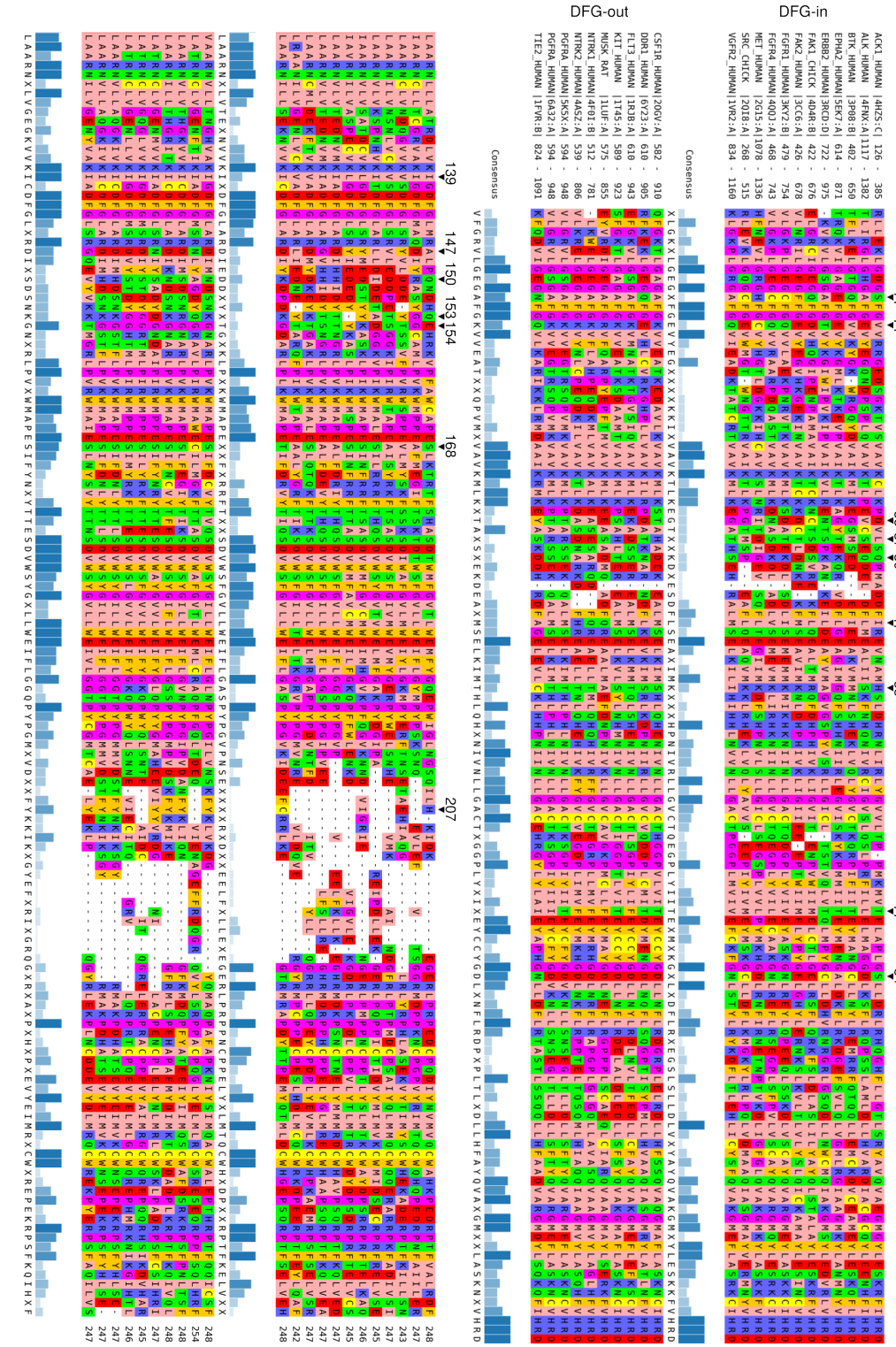

Figure 4.18: MSA of TK DFG-in (top) and DFG-out (bottom) domain sequences, with $\mathcal{P}_{\mathcal{K}}$ positions spanning 2–261 region. Selected positions are marked by black triangles.

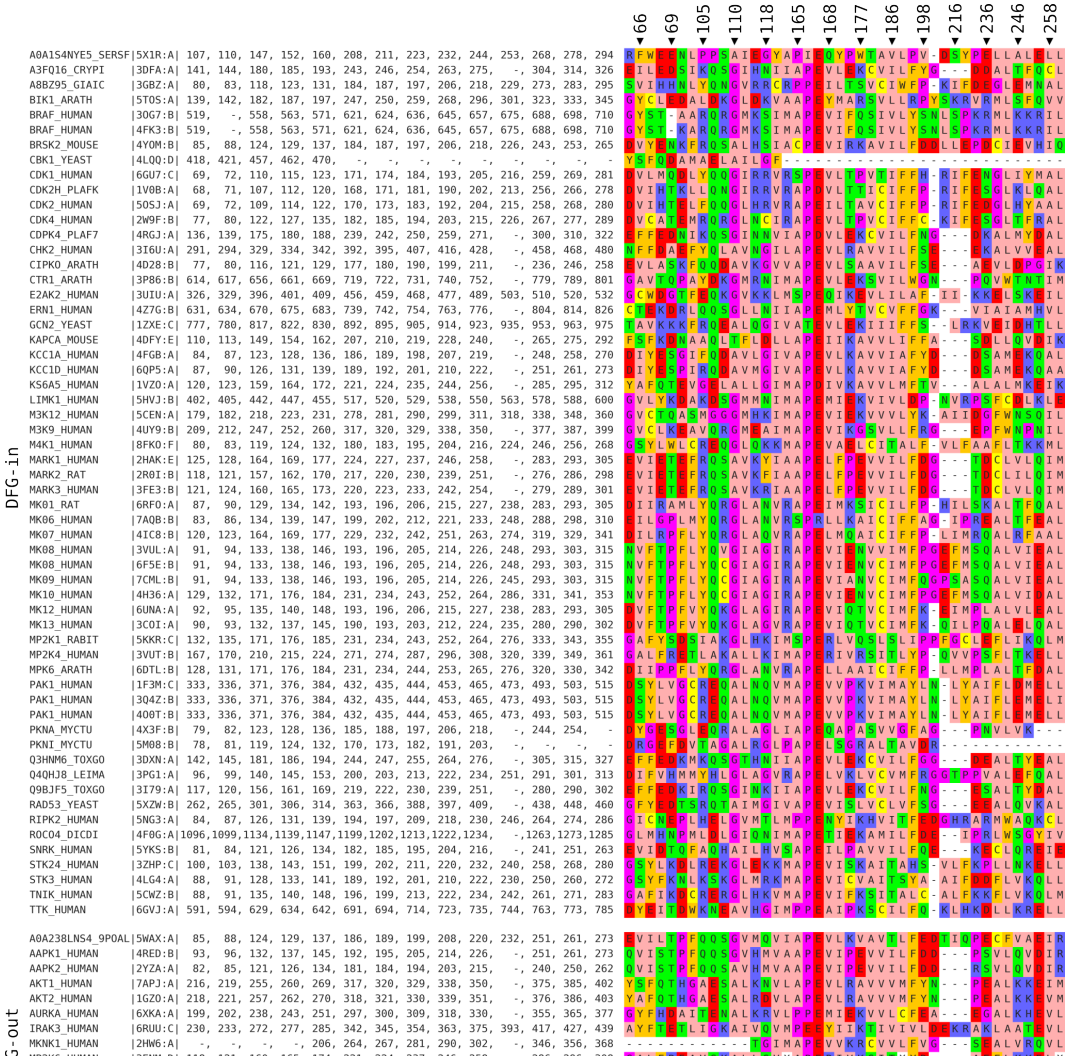

Figure 4.19: AAIO$_{STk}$ partial MSA depicting selected positions (±1). Only entries with unique subsequences corresponding to the selection are displayed. Sequence headers contain UniProt numbering of the selected positions.

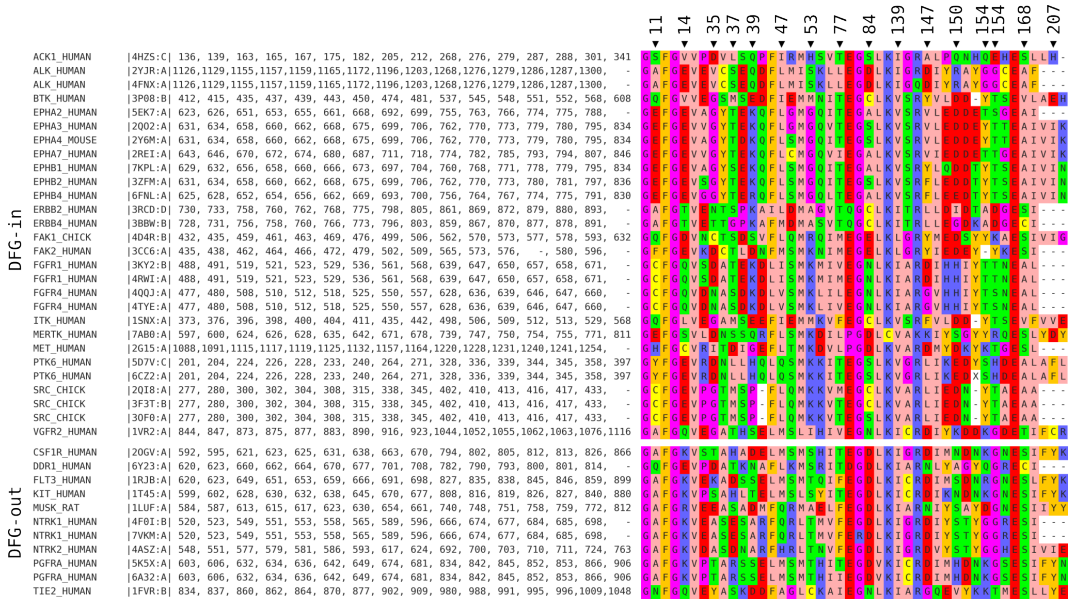

Figure 4.20: AAIO$_{Tk}$ partial MSA depicting selected positions (± 1). Only entries with unique subsequences corresponding to the selection are displayed. Sequence headers contain UniProt numbering of the selected positions.

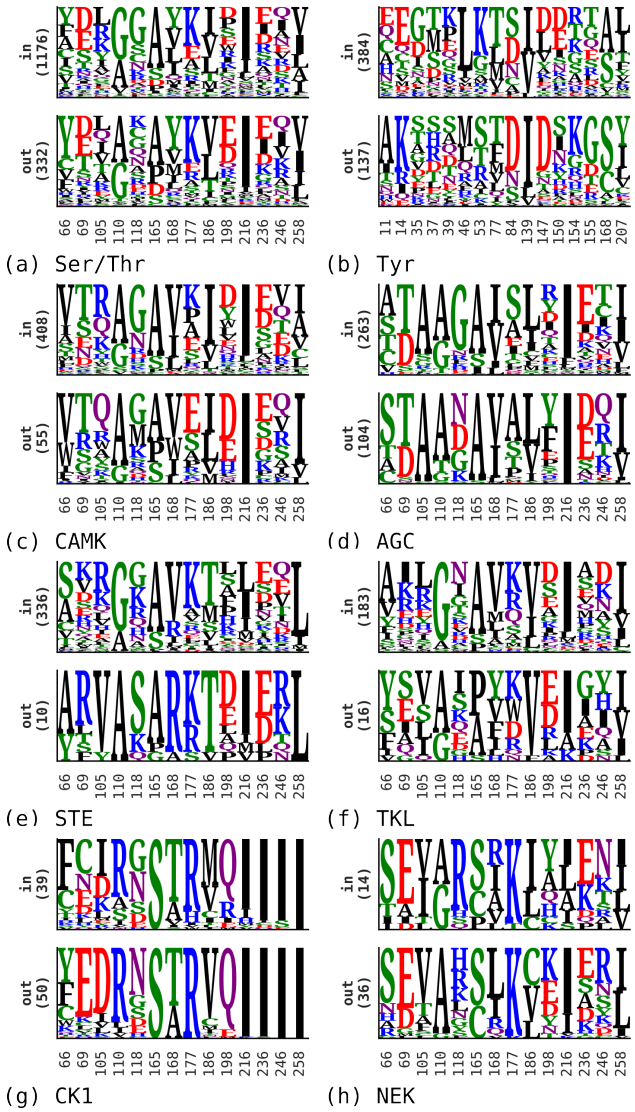

Figure 4.21: Sequence logos displaying selected positions of Swiss-Prot PK domains predicted as DFG-in (top) or DFG-out (bottom) using AAIO orthologs-based models – Tk for Tyr and STk for the rest.

References

1. Hanks, S. K., Quinn, A. M. & Hunter, T. The Protein Kinase Family: Conserved Features and Deduced Phylogeny of the Catalytic Domains. *Science* **241,** 42–52. ISSN: 0036-8075 (1988).
2. Leonard, C. J., Aravind, L. & Koonin, E. V. Novel Families of Putative Protein Kinases in Bacteria and Archaea: Evolution of the "Eukaryotic" Protein Kinase Superfamily. *Genome Res.* **8,** 1038–1047. ISSN: 1088-9051 (Oct. 1998).
3. Stancik, I. A. *et al.* Serine/Threonine protein kinases from bacteria, archaea and eukarya share a common evolutionary origin deeply rooted in the tree of life. en. *J. Mol. Biol.* **430,** 27–32. ISSN: 00222836 (2018).
4. Bradley, D. & Beltrao, P. Evolution of protein kinase substrate recognition at the active site. en. *PLoS Biol.* **17** (ed Turk, B. E.) e3000341. ISSN: 1545-7885 (June 2019).
5. Irby, R. B. & Yeatman, T. J. Role of Src expression and activation in human cancer. en. *Oncogene* **19,** 5636–5642. ISSN: 0950-9232, 1476-5594 (2000).
6. Wang, J. & Zhuang, S. Src family kinases in chronic kidney disease. eng. *Amer. J. Physio.: Renal Physiol.* **313.** Edition: 2017/06/14 Publisher: American Physiological Society, F721–F728. ISSN: 1522-1466 (2017).
7. Soverini, S., Mancini, M., Bavaro, L., Cavo, M. & Martinelli, G. Chronic myeloid leukemia: the paradigm of targeting oncogenic tyrosine kinase signaling and counteracting resistance for successful cancer therapy. *Molec. Canc.* **17,** 49 (2018).
8. Liu, Y. & Gray, N. S. Rational design of inhibitors that bind to inactive kinase conformations. *Nat. Chem. Biol.* **2,** 358–364. ISSN: 1552-4450 (2006).
9. Johnson, L. N. Protein kinase inhibitors: contributions from structure to clinical compounds. *Quart. Rev. Biophys.* **42,** 1–40. ISSN: 0033-5835 (2009).
10. Roskoski, R. Properties of FDA-approved small molecule protein kinase inhibitors: a 2020 update. en. *Pharm. Res.* **152,** 104609. ISSN: 10436618 (Feb. 2020).
11. Castelo-Soccio, L. *et al.* Protein kinases: drug targets for immunological disorders. *Nat. Rev. Immun.,* 1–20. ISSN: 1474-1733 (May 2023).

12. Li, S., Covino, N. D., Stein, E. G., Till, J. H. & Hubbard, S. R. Structural and Biochemical Evidence for an Autoinhibitory Role for Tyrosine 984 in the Juxtamembrane Region of the Insulin Receptor. *J. Biol. Chem.* **278,** 26007–26014. ISSN: 0021-9258 (2003).

13. McClendon, C. L., Kornev, A. P., Gilson, M. K. & Taylor, S. S. Dynamic architecture of a protein kinase. en. *Proc. Natl. Acad. Sci. USA* **111,** E4623–E4631. ISSN: 0027-8424, 1091-6490 (Oct. 2014).

14. Kanev, G. K. *et al.* The landscape of atypical and eukaryotic protein kinases. *Trends Pharm. Sci.* **40,** 818–832. ISSN: 0165-6147 (2019).

15. Modi, V. & Dunbrack, R. L. Defining a new nomenclature for the structures of active and inactive kinases. *Proc. Natl. Acad. Sci. USA* **116,** 6818 (2019).

16. Reveguk, I. & Simonson, T. *Classifying protein kinase conformations with machine learning* 2023.

17. Bhullar, K. S. *et al.* Kinase-targeted cancer therapies: progress, challenges and future directions. *Molec. Canc.* **17,** 48 (Feb. 2018).

18. Kooistra, A. J. *et al.* KLIFS: a structural kinase-ligand interaction database. *Nucl. Acids Res.* **44,** D365–D371. ISSN: 0305-1048 (2016).

19. Kanev, G. K., de Graaf, C., Westerman, B. A., de Esch, I. J. P. & Kooistra, A. J. KLIFS: an overhaul after the first 5 years of supporting kinase research. *Nucl. Acids Res.* **49,** gkaa895–. ISSN: 0305-1048 (2020).

20. Rahman, R., Ung, P. M.-U. & Schlessinger, A. KinaMetrix: a web resource to investigate kinase conformations and inhibitor space. *Nucl. Acids Res.* **47,** D361–D366. ISSN: 0305-1048 (2019).

21. Möbitz, H. The ABC of protein kinase conformations. *Biochem. Biophys. Acta* **1854,** 1555–1566. ISSN: 1570-9639 (2015).

22. Young, M. A. *et al.* Structure of the kinase domain of an Imatinib-resistant Abl mutant in complex with the Aurora kinase inhibitor VX-680. *Canc. Res.* **66,** 1007–1014. ISSN: 0008-5472 (2006).

23. Seeliger, M. A. *et al.* c-Src binds to the cancer drug Imatinib with an inactive Abl/c-Kit conformation and a distributed thermodynamic penalty. en. *Structure* **15,** 299–311. ISSN: 09692126 (Mar. 2007).

24. Aleksandrov, A. & Simonson, T. Molecular dynamics simulations show that conformational selection governs the binding preferences of imatinib for several tyrosine kinases. en. *J. Biol. Chem.* **285,** 13807–13815. ISSN: 0021-9258, 1083-351X. (2020) (2010).

25. Meng, Y., Lin, Y.-l. & Roux, B. Computational study of the "DFG-flip" conformational transition in c-Abl and c-Src tyrosine kinases. *J. Phys. Chem. B* **119,** 1443–1456. ISSN: 1520-6106 (2015).

26. Meng, Y. *et al.* Predicting the Conformational Variability of Abl Tyrosine Kinase using Molecular Dynamics Simulations and Markov State Models. *Journal of Chemical Theory and Computation* **14,** 2721–2732. ISSN: 1549-9618 (May 2018).

27. Hari, S. B., Merritt, E. A. & Maly, D. J. Sequence determinants of a specific inactive protein kinase conformation. en. *Chem. & Biol.* **20,** 806–815. ISSN: 10745521 (June 2013).

28. Azam, M., Seeliger, M. A., Gray, N. S., Kuriyan, J. & Daley, G. Q. Activation of tyrosine kinases by mutation of the gatekeeper threonine. *Nat. Struct. Mol. Biol.* **15,** 1109–1118. ISSN: 1545-9993 (2008).

29. Haldane, A., Flynn, W. F., He, P., Vijayan, R. & Levy, R. M. Structural propensities of kinase family proteins from a Potts model of residue co-variation. *Prot. Sci.* **25,** 1378–1384. ISSN: 1469-896X (2016).

30. Boutet, E., Lieberherr, D., Tognolli, M., Schneider, M. & Bairoch, A. UniProtKB/Swiss-Prot. *Meth. Mol. Biol.* **406,** 89–112. ISSN: 1064-3745 (2007).

31. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature,* 1–11. ISSN: 0028-0836 (2021).

32. Varadi, M. *et al.* AlphaFold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucl. Acids Res.* **50,** D439–D444. ISSN: 0305-1048 (2021).

33. Velankar, S. *et al.* SIFTS: structure integration with function, taxonomy and sequences resource. eng. *Nucl. Acids Res.* **41.** Edition: 2012/11/29 Publisher: Oxford University Press, D483–D489. ISSN: 1362-4962 (2013).

34. Li, W. & Godzik, A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* **22,** 1658–9. ISSN: 1367-4803 (May 2006).

35. Katoh, K. & Standley, D. M. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molec. Biol. & Evol.* **30,** 772–780. ISSN: 0737-4038 (2013).

36. Kunzmann, P. & Hamacher, K. Biotite: a unifying open source computational biology framework in Python. *BMC Bioinf.* **19,** 346. ISSN: 1471-2105 (2018).

37. Smith, T. & Waterman, M. Identification of common molecular subsequences. *J. Mol. Biol.* **147,** 195–197. ISSN: 0022-2836 (Mar. 1981).

38. Larralde, M. & Zeller, G. PyHMMER: a Python library binding to HMMER for efficient sequence analysis. *Bioinformatics.* ISSN: 1367-4803 (2023).

39. Eddy, S. R. Accelerated Profile HMM Searches. *PLoS Computational Biology* **7,** e1002195. ISSN: 1553-734X (2011).

40. Thomas, P. D. *et al.* PANTHER: Making genome-scale phylogenetics accessible to all. *Prot. Sci.* **31,** 8–22. ISSN: 0961-8368 (Jan. 2022).

41. Suzek, B. E., Huang, H., McGarvey, P., Mazumder, R. & Wu, C. H. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics* **23,** 1282–1288. ISSN: 1367-4803 (Mar. 2007).

42. Suzek, B. E. *et al.* UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **31,** 926–932. ISSN: 1367-4803 (Mar. 2015).

43. Cokelaer, T., Pultz, D., Harder, L. M., Serra-Musach, J. & Saez-Rodriguez, J. BioServices: a common Python package to access biological web services programmatically. *Bioinformatics* **29,** 3241–3242. ISSN: 1367-4803 (Dec. 2013).

44. Kawashima, S. & Kanehisa, M. AAindex: amino acid index database. eng. *Nucl. Acids Res.* **28.** Publisher: Oxford University Press, 374–374. ISSN: 0305-1048 (2000).

45. Van Westen, G. J. *et al.* Benchmarking of protein descriptor sets in proteochemometric modeling (part 1): comparative study of 13 amino acid descriptor sets. eng. *J. Chemoinf.* **5.** Publisher: BioMed Central, 41–41. ISSN: 1758-2946 (2013).

46. Westen, G. J. v. *et al.* Benchmarking of protein descriptor sets in proteochemometric modeling (part 2): modeling performance of 13 amino acid descriptor sets. *J. Chemoinf.* **5,** 42. ISSN: 1758-2946 (Sept. 2013).

47. Chen, T. & Guestrin, C. XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 785–794 (2016).

48. Breiman, L. Random forests. *Mach. Learn.* **45,** 5–32. ISSN: 0885-6125 (2001).

49. Friedman, J. H. Greedy function approximation: a gradient boosting machine. *The Annals Stat.* **29.** ISSN: 0090-5364 (2001).

50. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. *Optuna: a next-generation hyperparameter optimization framework* in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019).

51. Kursa, M. B. & Rudnicki, W. R. Feature selection with the Boruta package. *J. Stat. Soft.* **36,** 1–13 (2010).

52. Shapley, L. S. in *Contributions to the Theory of Games II* (eds Kuhn, H. W. & Tucker, A. W.) 307–317 (Princeton University Press, Princeton, 1953).

53. Lundberg, S. & Lee, S.-I. A Unified Approach to Interpreting Model Predictions. *arXiv.* eprint: 1705.07874 (May 2017).

54. Lundberg, S. M., Erion, G. G. & Lee, S.-I. Consistent individualized feature attribution for tree ensembles. *arXiv.* eprint: 1802.03888 (2018).

55. Minh, B. Q. *et al.* IQ-TREE 2: New Models and Efficient Methods for Phylogenetic Inference in the Genomic Era. *Molec. Biol. & Evol.* **37,** 1530–1534. ISSN: 0737-4038 (Feb. 2020).

56. Huerta-Cepas, J., Serra, F. & Bork, P. ETE 3: reconstruction, analysis, and visualization of phylogenomic data. *Molec. Biol. & Evol.* **33,** 1635–1638. ISSN: 0737-4038 (2016).

57. Iyer, M., Li, Z., Jaroszewski, L., Sedova, M. & Godzik, A. Difference contact maps: From what to why in the analysis of the conformational flexibility of proteins. *PLOS One* **15,** e0226702 (2020).

58. Singh, J. Comparison of conservation within and between the Ser/Thr and Tyr protein kinase family: proposed model for the catalytic domain of the epidermal growth factor receptor. *Prot. Engin. Des. Sel.* **7,** 849–858. ISSN: 1741-0126 (1994).

59. Taylor, S. S., Radzio-Andzelm, E. & Hunter, T. How do protein kinases discriminate between serine/threonine and tyrosine? Structural insights from the insulin receptor protein-tyrosine kinase. *The FASEB J.* **9,** 1255–1266. ISSN: 0892-6638 (1995).

60. Hanson, S. M. *et al.* What Makes a Kinase Promiscuous for Inhibitors? *Cell Chem. Bio.* **26,** 390–399.e5. ISSN: 2451-9456 (2019).

61. Zhao, Z. *et al.* Exploration of Type II Binding Mode: A Privileged Approach for Kinase Inhibitor Focused Drug Discovery? *ACS Chemic. Biol.* **9,** 1230–1241. ISSN: 1554-8929 (2014).

62. Yaghmaie, M. & Yeung, C. C. Molecular Mechanisms of Resistance to Tyrosine Kinase Inhibitors. *Curr. Hematol. Malig. Rep.* **14,** 395–404. ISSN: 1558-8211 (2019).

63. Rogers, C., Morrissette, J. J. & Sussman, R. NTRK Point Mutations and Their Functional Consequences. **262-263,** 5–15. ISSN: 2210-7762 (2021).

64. Fuse, M. J. *et al.* Mechanisms of Resistance to NTRK Inhibitors and Therapeutic Strategies in NTRK1-Rearranged Cancers. *Molecular Cancer Therapeutics* **16,** 2130–2143. ISSN: 1535-7163 (2017).

65. Kiyoi, H., Kawashima, N. & Ishikawa, Y. FLT3 mutations in acute myeloid leukemia: Therapeutic paradigm beyond inhibitor development. *Cancer Science* **111,** 312–322. ISSN: 1347-9032 (2020).

66. Griffith, J. *et al.* The Structural Basis for Autoinhibition of FLT3 by the Juxtamembrane Domain. *Molec. Cell* **13,** 169–178. ISSN: 1097-2765 (2004).

67. Mol, C. D. *et al.* Structural Basis for the Autoinhibition and STI-571 Inhibition of c-Kit Tyrosine Kinase. *J. Biol. Chem.* **279,** 31655–31663. ISSN: 0021-9258 (2004).

68. Walter, M. *et al.* The 2.7 Å Crystal Structure of the Autoinhibited Human c-Fms Kinase Domain. *J. Mol. Biol.* **367,** 839–847. ISSN: 0022-2836 (2007).

69. Heinrich, M. C. *et al.* PDGFRA Activating Mutations in Gastrointestinal Stromal Tumors. *Science* **299,** 708–710. ISSN: 0036-8075 (2003).

70. Rizzo, A., Pantaleo, M. A., Astolfi, A., Indio, V. & Nannini, M. The Identity of PDGFRA D842V-Mutant Gastrointestinal Stromal Tumors (GIST). *Cancers* **13,** 705. ISSN: 2072-6694 (2021).

71. Laine, E., Beauchêne, I. C. d., Perahia, D., Auclair, C. & Tchertanov, L. Mutation D816V Alters the Internal Structure and Dynamics of c-KIT Receptor Cytoplasmic Region: Implications for Dimerization and Activation Mechanisms. *PLoS Computational Biology* **7,** e1002068. ISSN: 1553-734X (2011).

72. Diskin, R., Engelberg, D. & Livnah, O. A novel lipid binding site formed by the MAP kinase insert in p38 alpha. *J. Mol. Biol.* **375,** 70–9 (June 2007).

73. Shan, Y., Arkhipov, A., Kim, E. T., Pan, A. C. & Shaw, D. E. Transitions to catalytically inactive conformations in EGFR kinase. *Proc. Natl. Acad. Sci. USA* **110,** 7270–7275. ISSN: 0027-8424 (2013).

74. Jauch, R. *et al.* Crystal Structures of the Mnk2 Kinase Domain Reveal an Inhibitory Conformation and a Zinc Binding Site. *Structure* **13,** 1559–1568. ISSN: 0969-2126 (2005).

75. Bollag, G. *et al.* Clinical efficacy of a RAF inhibitor needs broad target blockade in BRAF-mutant melanoma. *Nature* **467,** 596–599. ISSN: 0028-0836 (2010).

Chapter 5

lXtractor: Data Mining from Macromolecules

In the fast-evolving domains of structural biology, machine learning, and data mining, the meticulous and precise preparation of data stands as a critical yet frequently neglected phase. While existing tools offer a diverse repertoire of data description techniques, they inadequately address the foundational steps of data collection and management. This oversight fosters fragmented initiatives and repeated efforts in the scientific community, hindering the seamless integration and analysis that are vital for the development of robust machine learning models and data mining applications. Addressing this gap, we introduce `lXtractor` – an open-source Python library that synergizes data preparation and feature extraction, fostering transparent and complex data preparation workflows. This tool not only enables the creation of customizable workflows for data mining from macromolecular sequences and structures but also promotes transparency, reproducibility, and accessibility in data analysis. Through `lXtractor`, users can intuitively design and analyze sequence/structure data collections, with the added capability to scale up to vast data volumes and interface with various external resources. This chapter delineates the key principles, functionalities, and future directions of `lXtractor`, showcasing its potential as a robust backbone for data analysis and a catalyst for collaborative advancements in structural biology.

5.1 Introduction

The burgeoning repository of experimental [1, 2] and predicted [3, 4] structural data has propelled the rise of structure-based machine learning (ML) applications [5]. These applications hinge critically on data preparation and representation, factors that significantly influence their accuracy and reliability. Consequently, there is a pressing need for programmatic solutions, aiding in these steps, that are not only accessible but also adept at handling large-scale scenarios.

While several tools, including `BioPython` [6], `biotite` [7], `Bio3D` [8], and `ProDy` [9], facilitate the initial steps of structural data preparation, they often fall short in addressing more advanced needs. Designed to cater to a wide audience, these tools primarily focus on basic functionalities, making the task of adapting them for large-scale feature extraction and engineering a laborious process.

Thus, specialized tools for feature extraction from proteins have emerged. However, they predominantly focus on protein sequences rather than their structures [10–14]. Even among those that do consider tertiary structures, there is a noticeable deficiency in functionalities for pre-processing and correlating extracted features [15, 16], with Caretta being a notable exception [17]. This gap becomes particularly glaring when researchers aim to scrutinize groups of related molecules to answer complex questions such as, "What characteristics of the binding site enable family X proteins to accommodate Y ligands?" or "How does the conformation of region X vary across Y domain-containing proteins?" The existing methods, primarily serving as an intermediary layer between input data and their numerical representation, necessitate substantial additional effort and the integration of other tools to construct the final dataset suitable for data mining.

Therefore, we devised a comprehensive toolbox, `lXtractor` (/lɛˈkstraktər/), that seamlessly integrates data preparation and feature extraction. This tool facilitates the development of intricate yet fully customizable workflows for data mining, encompassing macromolecular sequences, structures, and their combinations. It promotes an interactive and adaptable approach to designing and analyzing sequence/structure data collections, with a strong emphasis on transparency, reproducibility, and accessibility.

`lXtractor` adeptly handles related protein structures and sequences by maintaining a clear connection to a selected reference entity, be it a sequence, Multiple Sequence Alignment (MSA), or an HMM model. The data collections curated by `lXtractor` are intuitively structured, allowing for convenient storage and retrieval, thus fostering iterative refinement and effortless distribution. Moreover, these data can be characterized by an array of sequence and structure descriptors, designed to preserve their linkage to the input data, thereby aiding in the extraction of significant insights during subsequent data mining phases.

For advanced users, the tool provides the flexibility to develop their own descriptors and employ `lXtractor` for their computation. Its capability to parallelize essential operations ensures scalability to handle extensive datasets. Furthermore, it offers basic interfaces to a variety of external resources, including SIFTS [18], Pfam [19], UniProt [20, 21], PDB [1], and AlphaFold2 [4], along with features to analyze ligands and their respective binding pockets.

This manuscript is structured as follows: Initially, we elucidate the core principles underlying `lXtractor` and outline the general workflow of its data mining methodology. Subsequently, we delve into the implementation specifics, highlighting the key data structures that constitute the backbone of the tool. Following this, we introduce a versatile "database" protocol to assemble a data collection of domain sequences and associated structures. Lastly, we illustrate the capabilities of `lXtractor` in facilitating variable computations.

5.2 Background

`lXtractor` is a Python library devised to facilitate the exploration of sequence and structure data in the realm of bioinformatics. It's designed to be a one-stop solution to carry a researcher from the initial hypothesis to the datasets ready for machine learning and data mining applications. Beyond its utility in data preparation, it may serve as a tool for the interactive analysis of macromolecular sequences and structures, fostering a deeper understanding and engagement with the data at hand.

The fundamental workflow of an `lXtractor` project can be delineated into the following stages, as illustrated in Fig. 5.1:

- **Data Gathering**: This initial stage involves acquiring the necessary

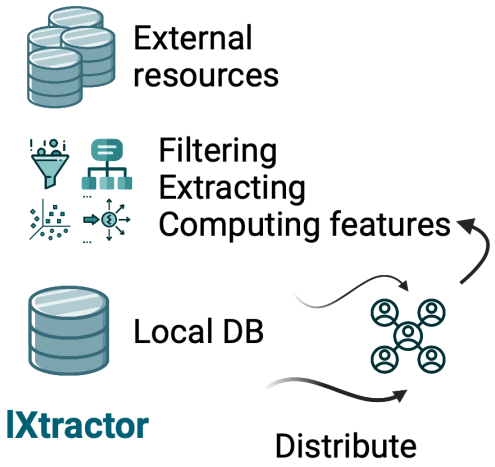

Figure 5.1: The core `lXtractor` workflow. The initial data are obtained from external resources, parsed to attain an internal representation, then processed and described by collections of variables. The data collections can then be saved, shared with the community, or loaded back into the Python interpreter for further refinement and analysis.

data for the project. The library offers minimalist interfaces to several databases including UniProt [20, 21], PDB [1], PANTHER [22], Pfam [19], and the AlphaFold2 database [3, 4] to facilitate this. It creates local interfaces to the Pfam database to access any desired HMM model, and the SIFTS database [18] to swiftly access mappings between UniProt and PDB identifiers, as well as segment-wise mappings between UniProt and PDB protein numbering schemes.

- **Data Parsing**: In this stage, the gathered sequence and structure data are parsed into an internal representation, structured into classes following an object-oriented approach. These objects facilitate the subsequent tasks:
  - **Mapping**: This involves relating sequences, alignments, and HMM models. Users are encouraged to establish a single reference object to map all pertinent sequences to it for variable calculations (see below).
  - **Filtering**: This task allows users to filter sequence/structure collections based on various criteria, such as the coverage of a structure sequence by a UniProt sequence or a reference object.
  - **Extracting**: This task enables the extraction of the regions of

interest from sequences and structures – continuous segments with boundaries defined either manually or by a reference object.

The tool offers the flexibility to perform these tasks in any order, fostering the development of a detailed and transparent workflow suited to individual use-cases.

- **Variables Calculation**: This stage involves computing variables for sequences and structures, categorized based on the input data required for their calculation:
  - **Sequence Variables**: Variables in this category require an arbitrary sequence as input.
  - **Structure Variables**: These variables necessitate an array of atoms for computation.
  - **Ligand Variables**: Variables in this category require an internal ligand representation as input.

  Each variable yields an atomistic output, either a single string or number. While users can define their own variables, these definitions must comply with specific standards.
- **Saving and Summarizing**: The final stage involves saving the data collections as sets of binary and text files, organised and named specifically. `lXtractor` also generates a summary in the form of a CSV table to offer a quick overview of the gathered data. These collections can be shared within the community and easily loaded into a Python interpreter for further refinement and analysis.

5.3 Implementation

5.3.1 Core Data Structures

At the core of the `lXtractor` library are a series of data structures engineered to facilitate the streamlined analysis of sequence and structure data. These structures form the backbone of the tool, enabling a seamless and efficient workflow.

The `Segment` object serves as a foundational unit, delineating sequences as named continuous segments. Each segment houses an array of indices along with a collection of arbitrary sequences that span its length. This object

operates as a flexible container, utilizing dictionary-like and list-like properties to expedite the inspection and addition of new sequences and metadata entries.

Expanding upon the `Segment` is the `ChainSequence`, a critical data structure within `lXtractor`. It accommodates a continuous sequence corresponding to a single polymeric chain, requiring a non-empty primary sequence element for each segment index. It permits the input of actual sequence numbering if it deviates from the default segment numbering. Beyond housing the primary sequence and its numbering, it can retain numerous additional arrays (such as secondary structure), which can be categorized and accessed with ease. Furthermore, the chain sequence can be flexibly mapped to other chain sequences, multiple sequence alignments (MSAs), or HMM models.

Working in tandem with the `ChainSequence` is the `GenericStructure`, capable of storing an arbitrary structure encompassing any number of chains, internally depicted as an array of atoms. During the parsing process, each atom is classified based on its function as a ligand, solvent, or polymer, with additional categorizations for polymeric atoms into protein, nucleotide, or carbohydrate groups. Ligands are documented as separate entities, as we elaborate below. These categories are maintained as binary masks, facilitating rapid access to specific segments within the atom array.

Complementing the `GenericStructure` is the `ChainStructure`, which integrates both the `GenericStructure` and `ChainSequence` corresponding to a single structural chain. In instances where the structural data presents multiple variants, as indicated by `altloc` PDB annotations, the `ChainStructure` restricts itself to a single `altloc`. These limitations ensure it embodies a singular polymeric entity, defined by fixed atom positions and a primary sequence outlined by polymeric atoms, thus preserving a transparent and unequivocal data representation.

In practice, it is not uncommon to encounter deviations in PDB structure sequences from the canonical UniProt sequences, which may be attributed to mutations or varying numbering conventions. Moreover, a multitude of PDB chains often correspond to a singular reference sequence. To address these complexities, the `Chain` object has been developed. This object encapsulates a single canonical `ChainSequence` and a series of associated `ChainStructures`, wherein the chain structure sequences are mapped to the canonical chain sequence through pairwise sequence alignment.

Within the `lXtractor` framework, the `ChainSequence`, `ChainStructure`, and `Chain` objects cater to diverse use-cases in data analysis and mining. Whether the focus is on analyzing pure sequence data, structure data, or a combination of both, `lXtractor` recognizes these "chain-type" objects as primary entities. Consequently, they offer similar interfaces, fostering a uniform approach in other segments of the codebase that are responsible for maintaining and manipulating these objects.

5.3.2 Chain Identifiers and Tree

The `lXtractor` library is designed to facilitate workflows that predominantly operate on interrelated chain-type objects, such as domains or smaller motifs. Consequently, a significant emphasis has been placed on the efficient extraction of such regions from sequences and structures.

Each chain-type object is equipped with a `spawn_child()` method, which allows for the subsetting of the chain based on specified boundaries. These boundaries are typically defined according to a certain numbering system, such as that of a domain or a reference sequence, stored within a `ChainSequence`. Figure 5.2 illustrates the process of creating a child segment from a `Chain`.

To streamline the management of object relationships, each chain-type object maintains a record of its "parent" and any "children" derived from it, thereby fostering the creation and inspection of segment trees. Moreover, each object is assigned a dynamic unique identifier, formulated from its name, segment boundaries, and a recursively appended list of parent identifiers. While these identifiers are not employed in object comparisons, adhering to a convention where matching identifiers indicate object identity is strongly recommended.

5.3.3 A List of Chains

Within the `lXtractor` framework, collections of chain-type objects are aggregated into a specialized container termed `ChainList`. This container serves as a direct extension of the Python `list` class, exclusively housing objects of a uniform type, namely, `ChainSequence`, `ChainStructure`, or `Chain`. Beyond the conventional list functionalities such as slicing, the `ChainList` offers enhanced capabilities including flexible filtering options, the application of ar-

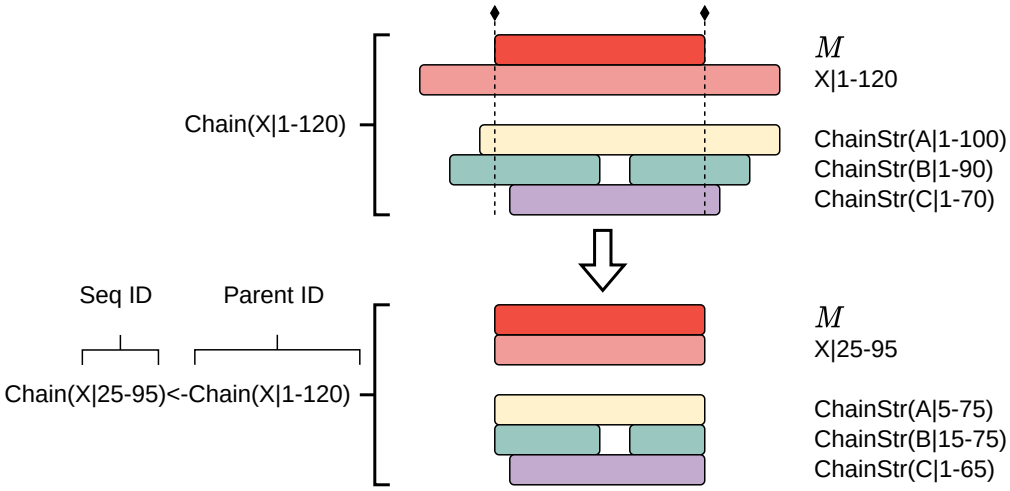

Figure 5.2: Illustration of a child segment creation from a `Chain`. The chain houses a canonical sequence X comprised of 120 residues, along with three structures - A, B, and C - whose sequences are mapped to X. This canonical sequence is further mapped to a reference M, delineating a specific region of interest. The extraction of this region initiates with the subsetting of the canonical sequence, followed by the utilization of sequence-to-sequence mappings to extract the child region from the stored structures. The newly formed child `Chain` is assigned a unique identifier that mirrors its lineage.

bitrary functions to its constituents, and the retrieval of all child objects at a specified topological level within a segment tree. These functionalities facilitate the construction of complex workflows, similar to `Pandas` data frames [23].

5.3.4 I/O and Storage Layout

The `lXtractor` library ensures a systematic approach to data storage and retrieval through the individual methods defined within each chain-type object. These methods facilitate the reading and writing of data to the disk, storing them as a series of text files that can be examined both manually and programmatically. Essential to each object are the obligatory sequence and metadata files, preserved as TSV tables. The metadata file functions as an object's "passport," encapsulating vital details such as the name and segment boundaries, alongside other pertinent information. Moreover, structure data can be accommodated in various formats including `pdb`, `cif`, and `mmtf`.

To streamline I/O operations, the `ChainIO` module has been developed,

capable of handling bulk storage and retrieval of chain-type objects. This module orchestrates a hierarchical organization of the objects, wherein child segments are nested under their respective parents, as illustrated in Fig. 5.3.

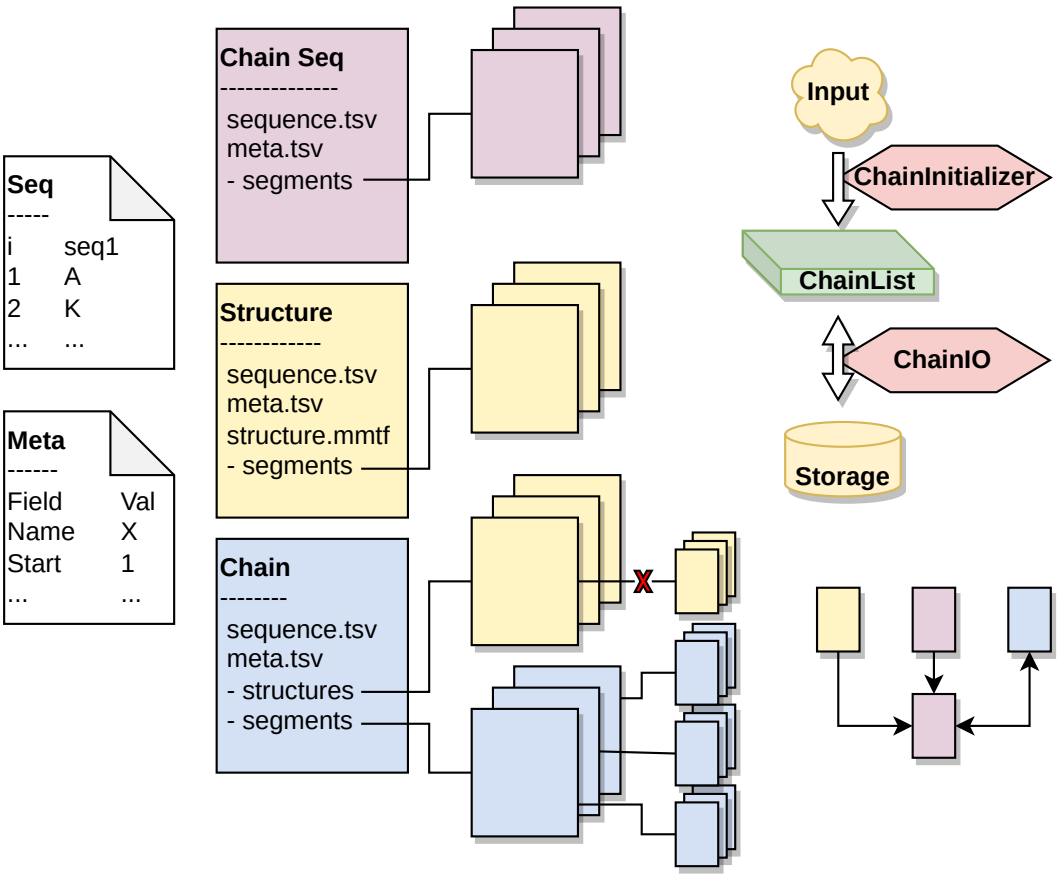

Figure 5.3: The data storage architecture of `lXtractor` segregates each object into distinct folders, each housing two compulsory files: `meta.tsv` and `sequence.tsv`. In the absence of supplementary files, an object is identified as a `ChainSequence`. The inclusion of a `structure.fmt` file, where `fmt` denotes the chosen format, elevates the object to a `ChainStructure`. Conversely, the addition of a `structures` folder classifies it as a `Chain`. This setup allows for flexible parsing, with objects being recognized as `ChainSequence` if structural data is not required. Furthermore, each object can recursively store child segments. In cases where a `Chain` accommodates child segments, these segments should be housed under the `Chain` rather than individual `ChainStructure`s. The `ChainInitializer` and `ChainIO` modules streamline the transformation of initial data into chain-type objects and their subsequent storage or retrieval from the disk, respectively.

To expedite the initial data parsing process, the `ChainInitializer` has been introduced. This class adeptly identifies the nature of the input data, initializing the appropriate chain-type objects accordingly. Furthermore, its

callback system offers the flexibility to tailor the initialization process to suit specific use-cases.

5.3.5 Ligands and Binding Pockets

In the process of parsing structure files, the `GenericStructure` categorizes atoms into one of three distinct groups: solvent, ligand, or polymeric. The categorization of solvents is facilitated through a carefully curated list of PDB residue codes, while polymeric atoms are identified based on the guidelines set by the Chemical Component Dictionary (CCD), accessible at https://www.wwpdb.org/data/ccd. Atoms that do not fall under the solvent or polymer categories are considered potential ligands.

The classification of atoms as a ligand is determined based on a set of criteria encompassing the number of ligand atoms, the number of contacts involving polymeric atoms and residues. These parameters are modifiable within the configuration settings of each `GenericStructure`. If a residue meets these criteria, it is encapsulated within a separate `Ligand` object, which retains the connection to its parent `GenericStructure`. The `Ligand` object stores various pieces of information, including the indices of its atoms within the parent structure, the contact atoms of the parent structure, distances to these atoms, and pertinent metadata. When generating subsets of a `GenericStructure`, such as during the division into chains, the ligand configuration guides the allocation of ligands into these subsets.

Complementing the `Ligand` object is the `Pocket` data structure, designed to efficiently differentiate ligands into "binders" and "non-binders". It utilizes a flexible string-based definition that translates into specific statements, which, when applied to a parent structure of a particular ligand, yields a binary response (True or False). Each definition comprises statements that can be algebraically combined, allowing for the definition of pockets with varying levels of complexity. For example, the statement below classifies a ligand as a binder if it establishes a minimum of two contacts with any atoms of a parent structure residue at position 2, or if both C_α atoms at structure positions 2 and 3 are within a distance of less than 6 Å from any ligand atoms:

```
c:1:any >= 2 | daa:2,3:CA < 6
```

To standardize pocket definitions, reference positions can be utilized, accompanied by a mapping that correlates these positions with the structure numbering. The computational efficiency of this approach facilitates the rapid filtering of thousands of ligands across a multitude of structures.

5.3.6 Variables Computation

`lXtractor` offers a flexible and consistent way to compute comprehensive sets of descriptor variables for sequences, structures, and ligands. To unify variable definitions, it is recommended to anchor them to a common reference numbering.

Variables are defined as classes adhering to a standard abstract interface. An important feature of a variable is its string definition, automatically created for each initialized variable from its name and attribute values. For instance, a variable

```
Dist(p1=1,p2=2,a1=`CA',a2=`CB')
```

specifies a distance between C_α and C_β atoms of protein positions 1 and 2, and executing a `eval("Dist(p1=1,p2=2,a1=`CA',a2=`CB')")` statement will create this variable within the Python interpreter, enabling seamless interconversion between textual and programmatic representation. This facilitates the creation of short and meaningful variable definitions, enhancing readability and ease of use.

Each variable specifies its own computation in the `calculate` method. As input, it accepts a sequence, atom array, or a ligand (subject to the variable type) and an optional mapping from reference to the object numbering, and returns a value, typically a number or a string, representing a specific atomistic property.

`lXtractor` defines sets of variables that cover basic use cases, such as secondary structure elements and ProtFP PCA embeddings [24, 25] for sequences, and basic geometric variables like distances and dihedral angles for structural data. Users may define their own variable types tailored for specific use cases, by inheriting the core interface from an abstract class.

Variables computation is orchestrated by the special `Manager` class. It accepts a series of chain-type objects, a reference mapping name, and a series of variables to compute. It automatically determines (object, variable) pairs

based on their types and retrieves necessary mappings. Once staged, it schedules variable calculations that can be parallelized. Finally, it dynamically aggregates the results as they become available and outputs them as a table.

5.3.7 Protocols

The `protocols` module is designed to encapsulate complex workflows tailored for specific analytical purposes. One such workflow, the `superpose` protocol, facilitates the analysis of a series of `ChainStructure` objects by orchestrating the superposition of each unique pair of structures. Initially, it aligns the sequences of each pair, followed by subsetting the aligned residues to common atoms. Subsequently, it executes the structural superposition and optionally computes a specified distance metric on the superposed structures. This computed distance metric can then be utilized for clustering analyses, aiding in the categorization and study of structural similarities and variations.

5.4 Demonstration

5.4.1 The Database Protocol

We'll demonstrate how `lXtractor` can be used to create sequence/structure data collections. Namely, we'll compile a collection of PDZ domain-containing proteins and extract these domains from sequences and structures. PDZ is a short (80-90 residues) domain serving as a protein interaction module in various cellular processes [26]. This protocol can be easily extrapolated towards other domains by simply changing the `PFAM_ID`.

Listing 5.1 demonstrates the setup of the protocol with imports and the configuration of relevant paths and constants like the number of CPUs for parallel processing.

Listing 5.1: Setup for the database protocol

```
from io import StringIO
from itertools import chain
from pathlib import Path

# Import chain and interfaces to external resources
import lXtractor.core.chain as lxc
from lXtractor.ext import (
    PyHMMer, PDB, SIFTS, Pfam, fetch_uniprot, filter_by_method
```

```
)
# Import utility functions
from lXtractor.util import read_fasta, valgroup
from toolz import keymap, valmap, valfilter

PFAM_ID = 'PF00595'  # Define the target domain reference model
NUM_PROC = 20  # A maximum number of CPUs to allocate
NUM_THREADS = 20  # A maximum number of threads to allocate
# Initialize paths to store data
PDB_DIR = Path('./tmp/pdb')
DB_DIR = Path(f'./tmp/{PFAM_ID}/db')
PDB_DIR.mkdir(exist_ok=True, parents=True)
DB_DIR.mkdir(exist_ok=True, parents=True)
# Initialize external resources
pdb = PDB(verbose=1, num_threads=NUM_THREADS)
sifts = SIFTS(load_id_mapping=True)
pfam = Pfam()
```

Listing 5.2 encapsulates the code to obtain the initial sequences from the SIFTS resource and parses them into `ChainSequence` objects for internal representation. Each sequence will be named by a UniProt identifier.

Listing 5.2: Obtaining and parsing initial sequences

```
# Fetch all SIFTS sequences from UniProt in fasta format
fetched = fetch_uniprot(
    sifts.uniprot_ids, num_threads=NUM_THREADS, chunk_size=500, verbose=True
)

# Parse the loaded sequences from the fetched data
fasta = read_fasta(StringIO(fetched))
# Convert the parsed fasta sequences into lXtractor's chain sequences
chains = lxc.ChainList(
    # `name.split('|')[1]` parses the full UniProt name into an ID
    lxc.ChainSequence.from_string(seq, name=name.split('|')[1])
    for name, seq in fasta
)
```

Next, the PDZ domain HMM model is applied to each parsed sequence (Listing 5.3). This model can be accessed from the `Pfam` interface, which automatically integrates it into the `PyHMMer` class. The latter serves as a simplified wrapper around the eponymous Python tool, which is an HMMer redeveloped in Cython [27, 28]. The `annotate` method of this class processes a collection of chain-type objects, extracting child segments corresponding to successful domain hits and generating results sequentially. These child segments are stored internally within each chain, allowing us to tally the domain hits by consuming the iterator.

Listing 5.3: Getting initial domain hits

```
# Load the HMM model from the Pfam interface using the specified PFAM_ID
hmm = pfam[PFAM_ID]

# Apply the HMM model to annotate the chain sequences, creating child sequences
# based on the domain boundaries identified for each sequence
num_domains = sum(
        1 for _ in hmm.annotate(chains, min_score=30, min_cov_hmm=0.7)
)

# Filter out sequences that did not have any domain hits
chains = chains.filter(lambda x: len(x.children) > 0)
```

After securing the initial domain hits, the next step is to consult SIFTS for corresponding structural chains in the PDB. Since the PDB only allows downloading complete structure files, it is necessary to first compile a list of query identifiers; the specific chains are extracted during the subsequent parsing. This list is further refined to include only entries associated with X-ray structures. The `PDB` interface efficiently manages the retrieval process, avoiding redundant downloads by checking the existing files in the designated directories, thereby preventing unnecessary strain on the PDB servers. Listing 5.4 showcases how `lXtractor` accomplishes these tasks with just a few lines of code.

Listing 5.4: Fetching relevant structures

```
# Generate UniProt–PDB mappings for valid hits using SIFTS,
# to be used later for initializing Chain objects
uniprot2pdb = {c.name: sifts[c.name] for c in chains}

# Compile an initial set of PDB IDs
pdb_ids = {x.split(':')[0] for x in chain.from_iterable(uniprot2pdb.values())}

# Refine the set to only include X-ray structures
pdb_ids = filter_by_method(
    pdb_ids, pdb=pdb, dir_=PDB_DIR / 'info', method="X-ray"
)
# Retrieve the structure files in mmtf format
pdb.fetch_structures(pdb_ids, PDB_DIR / 'mmtf', 'mmtf.gz')
```

With the necessary data retrieved, the next step is to construct an `lXtractor` data collection comprising `Chain` objects. This task is facilitated by the `ChainInitializer`, which requires a mapping between sequences and their corresponding structures. The sequences, already loaded and filtered to include only those with domain hits, are hashable objects in `lXtractor` and can be utilized directly as dictionary keys. To optimize memory usage, each

group of associated structural chains is represented as pairs consisting of the structure's path and a list of chain IDs. The `ChainInitializer` processes this mapping, initializing sequences and structure chains based on the input types provided. Additionally, it aligns each structural chain sequence with the respective canonical sequence through pairwise sequence alignment. Listing 5.5 illustrates the preparation of the sequence-structure mapping in the format expected by `ChainInitializer`, and the utilization of the latter to assemble the collection of `Chain` objects.

Listing 5.5: Initializing the data collection

```
# Clean up extracted segments to avoid duplicating child segments
# as Chain objects will be utilized in subsequent steps
chains = chains.apply(lambda c: c.filter_children(lambda _: False))

# Format the sequence-to-structure mapping into a format:
# SeqID => [(PDB, [Chain, ...]), ...]
seq2str = valgroup(uniprot2pdb)
# Replace PDB IDs with the appropriate paths to the retrieved structures
# `keymap` and `valmap` map a function to keys and values of a dictionary
seq2str = valmap(
    lambda xs: [
        PDB_DIR / 'mmtf' / f'{pdb_id}.mmtf.gz'
        for pdb_id, _ in xs if pdb_id in pdb_ids
    ],
    seq2str
)
# Replace UniProt IDs with the corresponding chain sequences
seq2str = keymap(
    lambda x: chains[x][0],
    seq2str
)
# Initialize the database using the prepared mappings
init = lxc.ChainInitializer(verbose=1, tolerate_failures=True)
chains = init.from_mapping(
    seq2str, num_proc_read_str=NUM_PROC, num_proc_map_numbering=NUM_PROC
)
```

Finally, invoking the `annotate` method on the chains initiates a process where the target HMM model aligns with each canonical sequence, extracting the domains as illustrated in Fig. 5.2[1]. Following domain extraction, several filtering steps are undertaken to retain only entries with valid and sufficiently large structural domains. The data collection is then saved to disk for future

[1]It should be noted that this step repeats the domain extraction for canonical sequences already verified to contain valid domains. Although less efficient than directly applying the previously identified domain boundaries to each structural chain, this approach serves demonstrative purposes better.

reference, with all objects summarized in a `pandas` dataframe, which can be stored as a CSV file for subsequent analysis. Listing 5.6 illustrates these steps.

Listing 5.6: Annotating domains and saving the collection

```
# Reapply domain mapping; now, domain boundaries influence
# both the canonical sequence and all associated chain structures
num_domains = sum(1 for _ in hmm.annotate(
    chains, min_score=30, min_cov_hmm=0.7, min_cov_seq=0.7,
    tolerate_failure=True, silent=True, str_map_from='map_canonical'
))

# Retain domain structures comprising at least 50 residues
for child in chains.collapse_children():
    child.structures = child.structures.filter(lambda x: len(x) > 50)
# Keep domain chains that have at least one extracted structural domain
for c in chains:
    c.children = c.children.filter(lambda child: len(child.structures) > 0)
# Maintain chains with a minimum of one extracted domain
chains = chains.filter(lambda c: len(c.children) > 0)

# Store the data collection on disk
io = lxc.ChainIO(verbose=1, num_proc=NUM_PROC)
sum(1 for _ in io.write(chains, DB_DIR))

# Compile all meta-information into a single table
df = chains.summary()
df.to_csv(DB_DIR / 'summary.csv')
```

Executing the outlined database protocol required approximately three minutes on a moderately powerful laptop (excluding the time for fetching the structural data). The resulting data collection of PDZ domains encompassed 146 complete canonical sequences and 227 domain sequences, correlating to 2337 structural domains.

5.4.2 Calculating Descriptors

In this section, we illustrate how `lXtractor` facilitates the computation of variables. Assuming the database protocol delineated in the preceding section has been executed, the objective now shifts to computing numerical descriptors for the extracted sequence and structure domains. Listing 5.7 delineates the setup procedure for this computation. After importing the necessary modules and defining constants, the domain segments are loaded. Given that their parent structures are not required for the computation, they are omitted; however, a specific callback is utilized to restore ancestral relationships from metadata,

necessary for the dynamic construction and display of identifiers. Moreover, since variables are anchored to the reference HMM numbering, it must be present in all objects. Hence, the last loop in Listing 5.7 transfers the target domain numbering from UniProt to PDB sequences.

Listing 5.7: Setup for variables calculation

```
from collections import Counter
from itertools import combinations, chain
from pathlib import Path

import numpy as np
# Import necessary modules for chain and variable management
import lXtractor.core.chain as lxc
from lXtractor.variables import Manager, GenericCalculator
from lXtractor.variables.structural import *
from lXtractor.variables.sequential import *
from toolz import keyfilter, valmap

PFAM_ID = 'PF00595'  # Reference model for the target domain
NUM_PROC = 20  # Maximum number of CPUs allocated for calculations
DB_DIR = Path(f'./tmp/{PFAM_ID}/db')  # Path to the stored database

# Initialize the database
io = lxc.ChainIO(verbose=1)
chains = lxc.ChainList(io.read_chain(
    DB_DIR.glob(f'*/segments/*{PFAM_ID}*'), callbacks=[lxc.recover])
)
# Transfer domain mappings from canonical sequences to
# the associated structures
for c in chains:
    c.transfer_seq_mapping(PFAM_ID)
```

With the chains loaded, Listing 5.8 illustrates the process of defining sequence and structure variables. Initially, positions covered in at least 80% of domains are identified to optimize the outcome. These positions then guide the definition of sequence variables, utilizing "Protein FingerPrints" (PFP) descriptors that encapsulate the physicochemical properties of amino acids, condensed through PCA analysis [24, 25]. Concurrently, structural variables are defined, initializing Φ and Ψ dihedral angles for each position, along with the minimum residue-residue distance for each position pair.

Listing 5.8: Defining variables for calculation

```
# Identify HMM positions covered in at least 80\% of domain sequences
counts = Counter(chain.from_iterable(s[PFAM_ID] for s in chains.sequences))
pos_coverage = valmap(lambda x: x / len(chains.sequences), counts)
positions = {int(k) for k, v in pos_coverage.items() if v > 0.8}

```

```
# Define variables for calculation
seq_variables = list(chain.from_iterable(
    # ProtFP embeddings (3 PCA components) for each position
    (PFP(p, i) for i in range(3)) for p in positions)
)
str_variables = list(chain(
    (Phi(p) for p in positions),  # Phi dihedral angle
    (Psi(p) for p in positions),  # Psi dihedral angle
    # Minimum residue-residue distance for each position pair
    (AggDist(p1, p2, 'min') for p1, p2 in combinations(positions, 2))
))
```

The calculation of variables is streamlined by the `Calculator` and `Manager` objects. The `Calculator` facilitates the calculation process, distributing it across multiple CPUs, while the `Manager` organizes the (object, variable) pairs for calculation, oversees execution, and compiles the results. It's vital to specify the common numbering name (`map_name`) in the `Manager` to align the reference HMM with the actual residue numbering, as shown in Listing 5.9.

Listing 5.9: Calculating variables and saving the results

```
# Initialize calculator and manager objects
calculator = GenericCalculator(verbose=True, num_proc=NUM_PROC)
manager = Manager()

# Stage and calculate sequence variables
staged = manager.calculate(
    chains.sequences, seq_variables, calculator, map_name=PFAM_ID
)
df_seq = manager.aggregate_from_it(staged, num_vs=len(seq_variables))
# Stage and calculate structure variables
staged = manager.calculate(
    chains.structures, str_variables, calculator, map_name=PFAM_ID
)
df_str = manager.aggregate_from_it(staged, num_vs=len(str_variables))

# Save the calculation results
df_seq.to_csv(DB_DIR / 'sequence_variables.csv')
df_str.to_csv(DB_DIR / 'structure_variables.csv')
```

Leveraging parallelization for critical operations significantly accelerates the calculation of variables. Consequently, the outlined protocol takes approximately two minutes to execute, generating 211 variables for each of the 227 domain sequences and 2556 variables for each of the 2337 domain structures, totaling 6,021,269 variables.

5.5 Methods

5.5.1 Implementation Details

`lXtractor` was developed in Python v3.10, featuring a modular structure organized into subpackages that house thematically distinct modules (Fig. 5.4). The source code adheres to the PEP 484 standard for type annotations, enhancing code reliability through the external `mypy` type checker used during development.

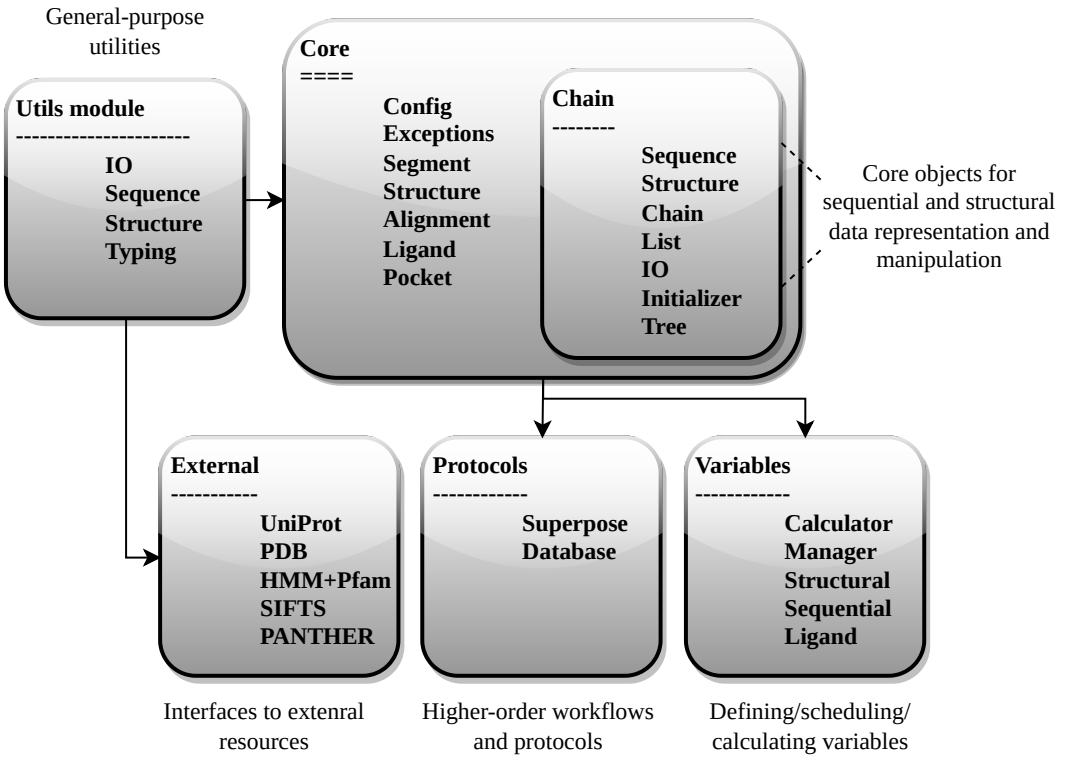

Figure 5.4: `lXtractor` codebase architecture.

Time-sensitive operations, including the management of chain-type objects and variables' calculations, can distribute computational load across available CPUs, utilizing either the built-in Python backend or `joblib`. Additionally, calls to external resources can be parallelized flexibly using multithreading.

5.5.2 Testing, Documentation, and Deployment

The package encompasses a comprehensive test suite, implemented with the `pytest` library, covering over 82% of the source code. Comprehensive docu-

mentation is available for all modules, classes, and functions, with API documentation generated from source code using `sphinx`.

Hosted on `GitHub`, the project employs webhooks to automate testing and documentation. A specific `GitHub` workflow triggers the test suite upon each commit, facilitating early error detection, while another assembles and deploys documentation to `readthedocs.org`. New releases are distributed through `PyPI` and can be installed using the command `pip install lxtractor`.

5.5.3 Executing the Protocols

The database and variable calculation protocols were executed on a laptop equipped with a 32-core 13th Gen Intel i9-13900HX processor and 32GB of RAM.

5.5.4 Package Dependencies

Table 5.1: `lXtractor` dependencies

Name	Usage	Ref.
`biotite`	Parsing and representing structure data.	[7]
`joblib`	Parallelizing jobs.	
`more-itertools`	Extending Python iterators.	
`networkx`	Analyzing chain trees.	[29]
`numpy`	Handling array computations.	[30]
`pandas`	Representing tabular data.	[23]
`pyhmmer`	Aligning sequence-to-HMM.	[28]
`toolz`	Processing dictionaries and composing functions.	
`tqdm`	Displaying progress bars.	
`requests`	Managing web requests.	

5.6 Discussion and Future Directions

In the rapidly evolving field of structural biology, machine learning and data mining have emerged as indispensable tools, unlocking the full potential of the expanding volume of structural data. The cornerstone of these applications lies in the establishment of reproducible, transparent, and accurate data preparation routines. Despite the plethora of tools available for describing these data,

a significant gap exists in addressing the initial steps of preparing and managing data collections. This gap often leads to different groups "reinventing the wheel" whenever such needs arise, hindering the progress of community-wide efforts.

To illustrate, numerous structural databases are dedicated to Protein Kinases (PKs) [31–35]. However, with the notable exception of Kincore [33], these databases do not disclose their exact data preparation workflows, presenting only the end results. This lack of transparency undermines the trust in these data and complicates collaborative efforts to enhance the quality of these datasets incrementally.

These observations spurred the development of `lXtractor` – an open-source Python library that seamlessly integrates data preparation and feature extraction. This tool facilitates the creation of transparent yet sophisticated data preparation workflows, providing a robust and flexible foundation for data analysis and description. In previous work [36], we utilized `lXtractor` to analyze a collection of PK domains. In this study, we introduced protocols for database and feature extraction that enable the assembly and numerical description of a collection of arbitrary domains without requiring any input data. For instance, executing these protocols for the PDZ domain allowed us to assemble 2237 structural domains corresponding to 227 sequences and compute over 6 million descriptors in approximately five minutes on a standard laptop.

Looking ahead, `lXtractor` is slated for active development, with plans to enhance basic functionalities, introduce more protocols, and expand the variety of available descriptors. Moreover, in a bid to democratize data mining in structural biology and alleviate the need of reassembling the data for each domain, we intend to create the "default" data collections for all domains available in the PDB. This initiative will enable users to easily access existing collections or contribute their own, fostering a collaborative and dynamic research environment.

References

1. Berman, H. M. *et al.* The protein data bank. eng. *Nucl. Acids Res.* **28.** Publisher: Oxford University Press, 235–242. ISSN: 0305-1048. http://www.rcsb.org/pdb/ (2000).
2. Burley, S. K. *et al.* Protein Data Bank: A Comprehensive Review of 3D Structure Holdings and Worldwide Utilization by Researchers, Educators, and Students. *Biomolecules* **12,** 1425 (Aug. 2022).
3. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature,* 1–11. ISSN: 0028-0836 (2021).
4. Varadi, M. *et al.* AlphaFold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucl. Acids Res.* **50,** D439–D444. ISSN: 0305-1048 (2021).
5. Durairaj, J., Ridder, D. d. & Dijk, A. D. v. Beyond sequence: Structure-based machine learning. *Comp. & Struc. Biotech. J.* **21,** 630–643. ISSN: 2001-0370 (2023).
6. Cock, P. J. A. *et al.* Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25,** 1422–3. ISSN: 1367-4803 (Mar. 2009).
7. Kunzmann, P. & Hamacher, K. Biotite: a unifying open source computational biology framework in Python. *BMC Bioinf.* **19,** 346. ISSN: 1471-2105 (2018).
8. Grant, B. J., Rodrigues, A. P. C., ElSawy, K. M., McCammon, J. A. & Caves, L. S. D. Bio3d: an R package for the comparative analysis of protein structures. *Bioinformatics* **22,** 2695–2696. ISSN: 1367-4803 (Aug. 2006).
9. Bakan, A., Meireles, L. M. & Bahar, I. ProDy: Protein Dynamics Inferred from Theory and Experiments. *Bioinformatics* **27,** 1575–1577. ISSN: 1367-4803 (June 2011).
10. Xiao, N., Cao, D.-S., Zhu, M.-F. & Xu, Q.-S. protr/ProtrWeb: R package and web server for generating various numerical representation schemes of protein sequences. *Bioinformatics* **31,** 1857–1859. ISSN: 1367-4803 (June 2015).
11. Bonidia, R. P., Domingues, D. S., Sanches, D. S. & Carvalho, A. C. P. L. F. d. MathFeature: feature extraction package for DNA, RNA and protein sequences based on mathematical descriptors. *Briefings Bioinf.* **23,** bbab434. ISSN: 1467-5463 (Nov. 2021).

12. Guevara-Barrientos, D. & Kaundal, R. ProFeatX: A parallelized protein feature extraction suite for machine learning. *Comp. & Struc. Biotech. J.* **21,** 796–801. ISSN: 2001-0370 (Dec. 2022).

13. Mu, Z. *et al.* FEGS: a novel feature extraction model for protein sequences and its applications. *BMC Bioinf.* **22,** 297 (June 2021).

14. Ismail, H., White, C., Al-Barakati, H., Newman, R. H. & Kc, D. B. FEPS: a Tool for Feature Extraction from Protein Sequence. *Meth. Mol. Biol.* **2499,** 65–104. ISSN: 1064-3745 (Jan. 2022).

15. Pande, A. *et al.* Pfeature: A Tool for Computing Wide Range of Protein Features and Building Prediction Models. *J. Comp. Biol.* **30,** 204–222 (Feb. 2023).

16. Chen, Z. *et al.* iFeatureOmega: an integrative platform for engineering, visualization and analysis of features from molecular sequences, structural and ligand data sets. *Nucl. Acids Res.* **50,** W434–W447. ISSN: 0305-1048 (2022).

17. Akdel, M., Durairaj, J., Ridder, D. d. & Dijk, A. D. v. Caretta – A multiple protein structure alignment and feature extraction suite. *Comp. & Struc. Biotech. J.* **18,** 981–992. ISSN: 2001-0370 (Apr. 2020).

18. Velankar, S. *et al.* SIFTS: structure integration with function, taxonomy and sequences resource. eng. *Nucl. Acids Res.* **41.** Edition: 2012/11/29 Publisher: Oxford University Press, D483–D489. ISSN: 1362-4962 (2013).

19. Mistry, J. *et al.* Pfam: the protein families database in 2021. *Nucl. Acids Res.* **49,** gkaa913–. ISSN: 0305-1048 (2020).

20. The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucl. Acids Res.* **45,** D158–D169. ISSN: 0305-1048 (2016).

21. The UniProt Consortium. UniProt: a worldwide hub of protein knowledge. *Nucl. Acids Res.* **47,** D506–D515. ISSN: 0305-1048 (2018).

22. Thomas, P. D. *et al.* PANTHER: Making genome-scale phylogenetics accessible to all. *Prot. Sci.* **31,** 8–22. ISSN: 0961-8368 (Jan. 2022).

23. Pandas development team, T. *pandas-dev/pandas: Pandas* version latest (Zenodo, 2020). https://doi.org/10.5281/zenodo.3509134.

24. Van Westen, G. J. *et al.* Benchmarking of protein descriptor sets in proteochemometric modeling (part 1): comparative study of 13 amino acid descriptor sets. eng. *J. Chemoinf.* **5.** Publisher: BioMed Central, 41–41. ISSN: 1758-2946 (2013).

25. Westen, G. J. v. *et al.* Benchmarking of protein descriptor sets in proteochemometric modeling (part 2): modeling performance of 13 amino acid descriptor sets. *J. Chemoinf.* **5,** 42. ISSN: 1758-2946 (Sept. 2013).

26. Lee, H.-J. & Zheng, J. J. PDZ domains and their binding partners: structure, specificity, and modification. *Cell Comm. & Sign.* **8,** 8 (May 2010).

27. Eddy, S. R. Accelerated Profile HMM Searches. *PLoS Computational Biology* **7,** e1002195. ISSN: 1553-734X (2011).

28. Larralde, M. & Zeller, G. PyHMMER: a Python library binding to HMMER for efficient sequence analysis. *Bioinformatics.* ISSN: 1367-4803 (2023).

29. Hagberg, A. A., Schult, D. A. & Swart, P. J. *Exploring Network Structure, Dynamics, and Function using NetworkX* in *Proc. 7 Python Sci. Conf.* (eds Varoquaux, G., Vaught, T. & Millman, J.) (Pasadena, CA USA, 2008), 11–15.

30. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585,** 357–362 (Sept. 2020).

31. Kooistra, A. J. *et al.* KLIFS: a structural kinase-ligand interaction database. *Nucl. Acids Res.* **44,** D365–D371. ISSN: 0305-1048 (2016).

32. Kanev, G. K., de Graaf, C., Westerman, B. A., de Esch, I. J. P. & Kooistra, A. J. KLIFS: an overhaul after the first 5 years of supporting kinase research. *Nucl. Acids Res.* **49,** gkaa895–. ISSN: 0305-1048 (2020).

33. Modi, V. & Dunbrack, R. L. Defining a new nomenclature for the structures of active and inactive kinases. *Proc. Natl. Acad. Sci. USA* **116,** 6818 (2019).

34. Rahman, R., Ung, P. M.-U. & Schlessinger, A. KinaMetrix: a web resource to investigate kinase conformations and inhibitor space. *Nucl. Acids Res.* **47,** D361–D366. ISSN: 0305-1048 (2019).

35. Möbitz, H. The ABC of protein kinase conformations. *Biochem. Biophys. Acta* **1854,** 1555–1566. ISSN: 1570-9639 (2015).

36. Reveguk, I. & Simonson, T. *Classifying protein kinase conformations with machine learning* 2023.

Corrections for the final version

This section introduces corrections requested by the jury members. Apart from typographical errors, which I won't detail here, the following changes/additions were implemented:

- Clarified the description of the clustering procedure in Section 3.4.2.
- Added Supp. Fig. 3.8 that complements Fig. 3.4 and demonstrates the plane usage to manually annotate DFG-in, DFG-out, and DFG-other conformations.
- Added Supp. Fig. 3.14 depicting a sequence logo of an entire PK domain to complement Fig. 3.6d.
- Added Supp. Fig. 3.17 demonstrating how selected structural variables differ between domains captured in active/inactive and DFG-in/out/other states.
- Fixed formatting issues in Table 3.6.
- Edited Section 4.3.8 to clarify the comparison of the sequence and structure-based DFG labels of the AlphaFold2 models.

List of Figures

List of Tables

Listings

ECOLE DOCTORALE

Titre: Machine learning pour comprendre et concevoir le kinome structurel

Mots clés: Machine learning, Protéines kinases, Bioinformatique structurale, Exploration de données, Ingénierie des protéines

Résumé: Les protéines kinases (PK) constituent l'un des groupes d'enzymes les plus anciens et les plus omniprésents profondément intégrés dans la machinerie moléculaire d'une cellule. En modifiant la conformation de leurs cibles via le transfert de phosphate, les PK eux-mêmes passent d'un état actif à un état inactif. Tout déséquilibre entre eux peut entraîner des maladies nocives, notamment des cancers. Le motif DFG, situé dans la boucle d'activation (AL), présente une variabilité conformationnelle dans l'état inactif moins contraint : une propriété que les inhibiteurs à petites molécules exploitent souvent. À savoir, il existe deux orientations principales du motif DFG, connues sous le nom de DFG-in et DFG-out. Ce dernier empêche la liaison au substrat et est généralement associé à une plus grande sélectivité de l'inhibiteur. Bien qu'ils soient essentiels aux efforts de conception de médicaments, on sait peu de choses sur les caractéristiques qui façonnent le paysage conformationnel de l'AL. Ce travail constitue une tentative systématique de les découvrir via une conservation et une extraction minutieuses des données. Au cours de son parcours, nous avons créé le plus grand assemblage de kinomes structurels à ce jour, englobant près de dix mille domaines PK annotés. Le regroupement de ces domaines a permis un marquage semi-supervisé des conformations du motif DFG. Ces étiquettes ont servi d'entrée à notre pipeline d'apprentissage automatique interprétable (ML) incorporant des ensembles basés sur des arbres de décision et un algorithme de sélection de caractéristiques indépendant du modèle. Les classificateurs obtenus ont prédit avec précision les conformations DFG et les états actifs/inactifs et se sont appuyés sur des caractéristiques structurelles facilement interprétables. Nous avons utilisé les annotations obtenues et les prédictions des modèles ML pour caractériser les éléments de séquence probablement responsables du déplacement de l'équilibre conformationnel de l'état inactif de l'AL, ou des positions "discriminatives". Pour les découvrir, nous avons créé plusieurs ensembles de données basés sur des séquences, chacun ayant un niveau différent de propension conformationnelle attribuée à une séquence. Nous avons utilisé le même pipeline ML et la même analyse phylogénétique pour montrer qu'une propension conformationnelle DFG claire est probablement privilégiée à un groupe de récepteurs tyrosine protéine kinases étroitement apparentés. Les positions discriminantes découvertes chevauchent la littérature existante et les études de mutagenèse et peuvent fournir une base pour de futurs efforts expérimentaux, y compris des applications de conception de protéines informatiques. Enfin, la méthodologie développée permet d'automatiser l'annotation du kinome structural. Généralisable à des problèmes de même nature, il peut accroître l'efficacité et la transparence de la fouille de données en biologie structurale.

Title: Machine learning to understand and engineer the structural kinome)

Keywords: Machine learning, Protein kinases, Structural bioinformatics, Data mining, Protein engineering

Abstract: Protein kinases (PKs) comprise one of the most ancient and ubiquitous enzyme groups deeply embedded in a cell's molecular machinery. Changing their targets' conformation via phosphate transfer, PKs themselves cycle between active and inactive states. Any misbalance between them can lead to harmful diseases, including cancers. The DFG motif, situated within the activation loop (AL), displays conformational variability in the less constrained inactive state: a property that small molecule inhibitors often exploit. Namely, there are two major DFG motif orientations, known as the DFG-in and DFG-out. The latter precludes substrate binding and is typically associated with higher inhibitor selectivity. Despite being pivotal to drug design efforts, little is known regarding which features shape AL conformational landscape. This work constitutes a systematic attempt to uncover them via careful data curation and mining. Over its course, we created the largest structural kinome assembly to date, encompassing nearly ten thousand annotated PK domains. Clustering these domains enabled semi-supervised labeling of the DFG motif conformations. These labels served as input to our interpretable machine-learning (ML) pipeline incorporating decision tree-based ensembles and a model-agnostic feature selection algorithm. The obtained classifiers accurately predicted the DFG conformations and active/inactive states and relied on readily interpretable structural hallmarks. We used the obtained annotations and ML models' predictions to characterize sequence elements likely responsible for shifting the AL inactive state's conformational balance, or "discriminative" positions. To uncover these, we created several sequence-based datasets, each having a different level of conformational propensity attributed to a sequence. We used the same ML pipeline and phylogenetic analysis to show that a clear DFG conformational propensity is likely privileged to a group of closely related receptor tyrosine protein kinases. The discovered discriminating positions overlapped with the existing literature and mutagenesis studies and may provide a foundation for future experimental efforts, including computational protein design applications. Finally, the developed methodology enables automating the annotation of the structural kinome. Generalizable towards problems of similar nature, it may increase the efficiency and transparency of data mining in structural biology.

Institut Polytechnique de Paris
91120 Palaiseau, France