A COMPARATIVE STUDY OF MACHINE-LEARNING-BASED SCORING FUNCTIONS IN PREDICTING PROTEIN-LIGAND BINDING AFFINITY

By

Hossam Mohamed Farg Ashtawy

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Electrical Engineering

2011

ABSTRACT

A COMPARATIVE STUDY OF MACHINE-LEARNING-BASED SCORING FUNCTIONS IN PREDICTING PROTEIN-LIGAND BINDING AFFINITY

By

Hossam Mohamed Farg Ashtawy

Accurately predicting the binding affinities of large sets of diverse protein-ligand complexes is a key challenge in computational biomolecular science, with applications in drug discovery, chemical biology, and structural biology. Since a scoring function (SF) is used to score, rank, and identify drug leads, the fidelity with which it predicts the affinity of a ligand candidate for a protein's binding site and its computational complexity have a significant bearing on the accuracy and throughput of virtual screening. Despite intense efforts in this area, so far there is no universal SF that consistently outperforms others. Therefore, in this work, we explore a range of novel SFs employing different machine-learning (ML) approaches in conjunction with a diverse feature set characterizing protein-ligand complexes. We assess the scoring and ranking accuracies and computational complexity of these new ML-based SFs as well as those of conventional SFs in the context of the 2007 and 2010 PDBbind benchmark datasets. We also investigate the influence of the size of the training dataset, the number of features, the protein family, and the novelty of the protein target on scoring accuracy. Furthermore, we examine the interpretive power of the best ML-based SFs. We find that the best performing ML-based SF has a Pearson correlation coefficient of 0.797 between predicted and measured binding affinities compared to 0.644 achieved by a state-of-the-art conventional SF. Finally, we show that there is potential for further improvement in our proposed ML-based SFs.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Protein-ligand binding affinity is the principal determinant of many vital processes, such as cellular signaling, gene regulation, metabolism, and immunity, which depend upon proteins binding to some substrate molecule. Accurately predicting the binding affinities of large sets of diverse protein-ligand complexes remains one of the most challenging unsolved problems in computational biomolecular science, with applications in drug discovery, chemical biology, and structural biology. Computational molecular docking involves docking of tens of thousands to millions of ligand candidates into a target protein receptor's binding site and using a suitable scoring function to evaluate the binding affinity of each candidate to identify the top candidates as leads or promising protein inhibitors. Since a scoring function is used to score, rank, and identify drug leads, the fidelity with which it predicts the affinity of a ligand candidate for a protein's binding site and its computational complexity have a significant bearing on the accuracy and throughput of virtual screening. Despite intense efforts in this area, so far there is no universal scoring function that consistently outperforms others. Therefore, in this work, we explore a range of novel scoring functions employing different machine-learning (ML) approaches in conjunction with a diverse feature set characterizing protein-ligand complexes with a view toward significantly improving scoring function accuracy compared to that achieved by existing conventional scoring functions.

In this chapter, we first briefly describe the process of designing new drugs and introduce techniques used to expedite such task. These techniques include high-throughput screening and computational methods such as virtual screening and molecular docking. Scoring functions, which are a key component of molecular docking, are then presented and their types and limitations are also briefly covered. The challenges associated with conventional scoring functions have motivated us and other researchers to build more accurate and robust scoring models. A more detailed motivation behind this work and a review of related studies are discussed at the end of the chapter.

1.1 Background

1.1.1 Drug Design

The first step in designing a new drug for many diseases is accurately identifying the enzyme, receptor, or other protein responsible for altering the biological functioning of the signaling pathway associated with the disease. This protein must be a critical component in the activity of the cell linked to the disease. Then finding chemical molecules that stimulate the protein to modulate the cell's function is the main idea of drug discovery. Such molecules (drugs) should be able to inhibit or activate the cell's cycle or specific function. Furthermore, proteins stimulated by such drugs should be unique and their function may not be substituted by other proteins in the cell.

In the early days, natural products derived from animals and plants (e.g. alkaloids and terpenoids) were the dominant sources of drugs. Over the years new synthetic and semi-synthetic methodologies have been developed to generate large numbers of competitive and effective drug-like compounds. The field under which these methodologies lie is known as *combinatorial chemistry* which is widely utilized by pharmaceutical companies to help to produce their in-house compound libraries. Automated devices and robots are typically used to accelerate the process of synthesizing and screening chemical compounds from these large databases. This process is generally known as High Throughput Screening (HTS), which in the context of drug design is used to rapidly synthesize and verify the activity of small molecules (Ligands) against the given protein. Although HTS technology has significantly shortened the time and reduced the costs of discovering new drugs, it is still not fast and cheap enough to screen the large and yet ever-growing compound databases.

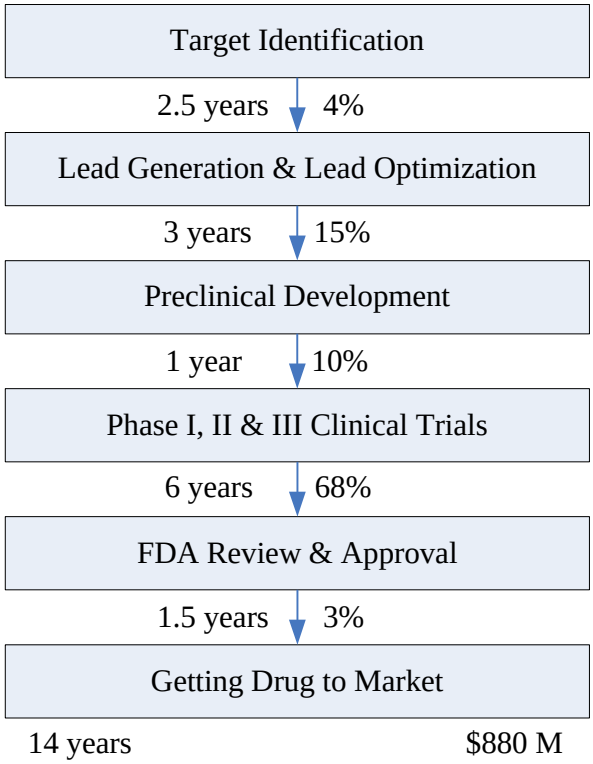


Figure 1.1 Time and cost involved in drug discovery [2]. *For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis.*

Discovery and development of a drug requires more than 10 years of research and development on an average and a budget that varies from hundreds of millions to tens of billions of dollars [1], [2]. Even with this prohibitive cost in time and money, its clinical success rate does not exceed 25% [3]. Such high expenses are mainly due to screening large numbers of drug-like compounds against the given target, follow up assays for investigating toxicological and pharmacokinetic properties, and clinical trials. Figure 1.1 shows the cost and time associated with the main stages of drug discovery based on data collected by PARAXEL several years ago [2].

1.1.2 Virtual Screening

In order to reduce laborious and costly hit-identification (or lead optimization) search campaigns, computational methods are widely employed to filter large in-house compound libraries into smaller and more focused ones. Such *virtual screening* methods are used as integral component of HTS. Virtual screening narrows down the number of ligands to be tested experimentally by orders of magnitude, which in turn dramatically could shorten drug discovery timeline and reduce associated costs.

In virtual screening, information, such as structure and physicochemical properties, about ligand, protein, or both is used to infer the activity of interest that would likely to take place in real physical settings. The essential biological interaction between a ligand and a target protein in the early phases of drug design is normally represented by binding affinity (or binding free energy). Binding affinity determines whether a ligand could bind to the given protein, and if so, with how much efficacy and potency. The most popular approach to predicting binding energy in virtual screening is structure-based in which physicochemical interactions between a ligand and target protein are extracted from the 3D structure of both molecules. Three-dimensional structures of proteins are typically obtained using X-ray crystallography [4], NMR spectroscopy [5] or homology models [6], [7]. This *in silico* method is also known as *protein-based* as opposed to the alternative approach, *ligand-based*, in which only ligands similar to the ones known to bind to the target are screened. Such ligand-based screening ignores valuable information about the target and exclusively considers compounds exhibiting common biochemical features. This limited and biased (toward the known ligand) search may not identify novel chemicals as hits. However, it is the method of choice when the 3D structure for the target is not available.

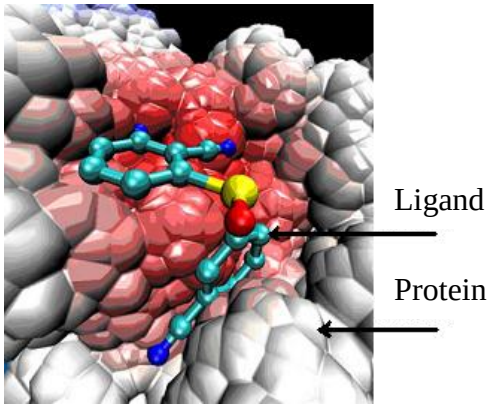


Figure 1.2 A binding pose of a ligand in the active site of a protein [8].

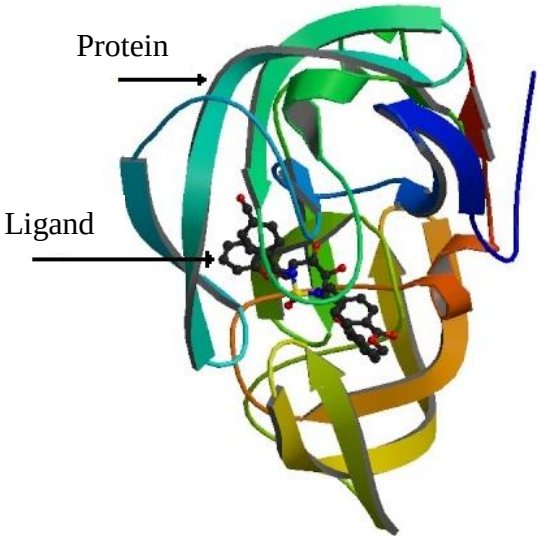


Figure 1.3 Crystal structure of HIV-1 protease in complex with inhibitor AHA024 [9].

1.1.3 Molecular Docking and Scoring Functions

In this work, the focus will be on protein-based drug design, where ligands are placed into the active site of a receptor as depicted in Figures 1.2 and 1.3 [8], [9]. Figure 1.2 is a solvent-accessible surface representation of a ligand's pose in the protein's active site and Figure 1.3 illustrates a ribbon diagram of an inhibitor (AHA024) binding to HIV-1 protease (protein). The orientation of a molecule and its position in the active region of the protein's 3D structure is

known as *conformation*. The process in which a set of ligand conformations is generated and placed in the binding pocket is generally known as *docking*. Every ligand's conformation (or pose) during the course of docking is evaluated and the binding mode of the ligand is identified.

Typically, an internal engine built inside the docking tool known as *a scoring function* is used to discriminate between the promising binding modes from poor ones. Such a function takes a specific ligand's conformation in the protein's binding cavity, and estimates a score that represents the binding free energy of that orientation. The outcome of the docking run therefore is a set of ligands ranked according to their predicted binding scores. Top ranked ligands are considered the most promising drugs so they proceed to the next phase where they will be synthesized and physically screened using HTS. Figure 1.4 depicts a simplified view of the drug design process.

The success of docking simulation relies upon the effectiveness of the docking algorithm and the quality of the utilized scoring function. Various docking protocols adopt different strategies in exploring the conformational space of ligands and their targets. The treatment of ligand and protein molecules in terms of their flexibility is one of the distinguishing features of a docking method. The most rapid docking techniques usually assume a rigid ligand interacting with a rigid target, and some tools at the other end of the extreme take the flexibility of both objects into consideration. Although the latter paradigm yields more accurate results, its application is computationally demanding and therefore it fails to provide one of the most attractive features of virtual screening: speed. Ideally, in order to trade-off between accuracy and high throughput, the mainstream docking systems assume the target is held fixed or minimally flexible in its crystal structure and simulate the conformational flexibility of the ligands [10], [11], [12].

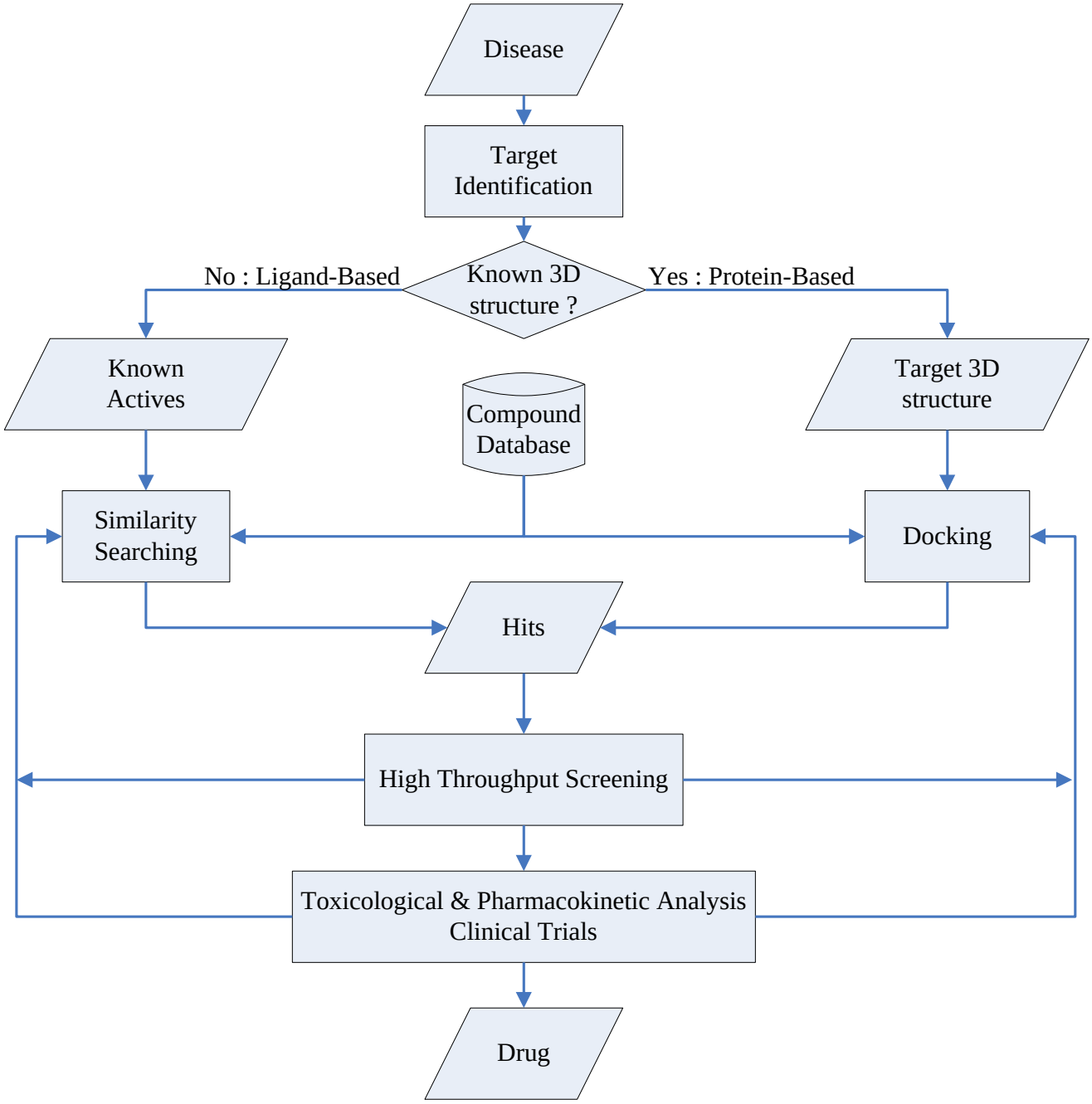


Figure 1.4 The drug design process.

1.1.4 Types of Scoring Functions

No matter how exhaustive and efficient the docking algorithm is in exploring the conformational space, the entire *in silico* run may not reveal interesting hits unless the scoring engine is effective enough. The scoring function should be precise and fast in predicting the binding energy of protein-ligand snapshots from the vast conformational space. Precision and speed are inherently conflicting requirements. To make scoring functions more accurate, they have to capture deeper details regarding interacting forces between protein and its ligand. Thorough accommodation of such factors involved in determining binding affinity is computationally intensive and therefore practically infeasible for virtual screening. Several scoring schemes have been formulated that attempt to calculate accurate binding energies within reasonable time limits. The most popular schemes can be categorized under three major families: force-field, empirical, and knowledge-based scoring functions.

In force-field methods, the potential of a system is formulated by calculating physical atomic interactions between the protein and ligand that include van der Waals forces, bonding and non-bonding interactions, and electrostatic energies. Parameters that define these intermolecular interactions are derived from experimental data and *ab initio* simulations [13]. The limitations of these classes of functions arise from inaccurate implicit accounting for solvation, enthalpy, and absence of entropy [8], [14]. Examples of some well-known docking programs that employ this class of scoring functions are DOCK [10], [15], [16], [17] and AutoDock [18], [19].

Empirical scoring functions adopt a different philosophy for calculating the free energy of the system. The energy as a whole in this type of functions is assumed to be composed of weighted energy terms. Each term describes a chemically intuitive interaction such as hydrogen

bonding, desolvation effect, van der Waal interactions, hydrophobicity, and entropy. A fitting to a training dataset of known experimental binding energy is typically conducted to calibrate the coefficients of interest via linear regression [20]. Due to its intrinsic simplicity, empirical scoring is the method of choice when screening speed is of great importance [21].

Finally, knowledge-based scoring is founded on the theory that large databases of protein-ligand complexes can be statistically mined to deduce rules and models that are implicitly embedded in the data. Distance-dependent occurrences of certain atom pair types are counted from experimentally determined protein-ligand structures and used as descriptors or free energies. There are two common approaches to utilizing these descriptors in predicting the binding energy. The first one is by summing up the free energies of the interatomic contacts obtained according to inverse Boltzmann relation [22], [23]. In this case, experimental measurements of binding affinity are not required. Examples of such an approach are PMF and DrugScore [24], [25], [26], [27], [28], [29].

The second type of knowledge-based scoring function is inspired from empirical models, where descriptors are used by a machine-learning method to fit the binding affinities of a training set. Ballester and Mitchell borrow this technique in building their RF-Score function [30]. Knowledge-based scoring functions' accuracy and speed are comparable to those of empirical functions [1].

1.1.5 Scoring Challenges

Despite the aforementioned variety of docking methods and scoring functions, several recent studies report that there is no superior docking tool (docking algorithm and scoring function) that outperforms its counterparts [31], [32]. This can be attributed to imperfections in the docking

process as well as deficiencies in scoring. In addition to the insufficient amount of protein flexibility considered during docking that leads to the limited predictive power, the other source of the problem can be traced to the capability of employed scoring function. Lack of full and accurate accounting of intermolecular physicochemical interactions and improper solvent modeling are believed to be the main causes of errors in computing binding energies. Experimental data to which regression techniques are fitted also poses a challenge to the quality of resulting scoring models. More specifically, uncertainties may be associated with collected binding affinities of protein-ligand complexes that are typically used to build scoring functions. Binding affinities are usually assembled from several research groups who obtain such values under different experimental conditions [8].

1.2 Motivation and Problem Statement

It has been difficult to capture the unknown relationship correlating binding affinity to interactions between a ligand and its receptor. Conventional empirical scoring functions rest on the hypothesis that a linear regression model is capable of computing the binding affinity. Such an assumption fails to explain intrinsic nonlinearities latent in the data on which such models are calibrated. Instead of assuming a predetermined theoretical function that governs the unknown relationship between different energetic terms and binding affinity, one can employ more accurate methodologies that are generally referred to as *nonparametric* models. In these techniques, the underlying unknown function is "learnt" from the data itself and no assumption regarding its statistical distribution is made. Various nonparametric machine-learning methods inspired from statistical learning theory are examined in this work to model the unknown function that maps structural and physicochemical information of a protein-ligand complex into a corresponding binding energy value. In order to make the derived models capture more information about protein-ligand interactions, we considered a variety of descriptors that are normally employed either for knowledge-based or for empirical scoring functions. Namely, these energy terms include hydrogen bonding, hydrophobic forces, and van der Waals interactions employed in empirical scoring functions and QSAR (quantitative structure activity relationships) descriptors such as pairwise atomic statistics used in knowledge-based scoring functions.

We seek to advance structure-based drug design by designing scoring functions that significantly improve upon the protein-ligand binding affinity prediction accuracy of conventional scoring functions. Our approach is to couple the modeling power of flexible machine learning algorithms with training datasets comprising thousands of protein-ligand

complexes with known high resolution 3D crystal structures and experimentally-determined binding affinities and a variety of features characterizing the complexes. We will compare the predictive accuracies of several machine learning based scoring functions and existing conventional scoring functions of all three types, force-field based, empirical, and knowledge-based, on diverse and independent test sets.

In similar studies reported in the literature, the focus was more or less on the bioactivity of a ligand when it binds to its receptor. In such cases, the given ligand is classified as either an active or an inactive molecule for the target protein. Although in other works the binding affinity as a numerical value was of interest, the data used to train and test prediction power of different techniques was either restricted to very few protein families and/or the size of the dataset itself was not as ample and diverse as the ones we are exploiting in our experiments. Furthermore, the abundance of different techniques to extract features characterizing protein-ligand complexes has motivated us to experiment building models that are trained on diverse sets of descriptors.

More specifically, in this work, we will evaluate and compare the performance of different conventional and machine-learning-based scoring functions. The issues we will explore include:

- Assessment of scoring and ranking accuracies of scoring functions based on comparison between predicted and experimentally-determined binding affinities.
- Dependence of scoring function accuracy on the size of training data.
- Dependence of scoring function accuracy on the number of features describing protein-ligand complexes.
- Performance of scoring functions on family-specific proteins and novel targets.
- Descriptive power of scoring functions (i.e., their interpretability).

- Throughput of scoring functions in terms of the time required to extract the features and calculate binding affinity.

1.3 Related Work

Several comparative studies of different prediction methods applied in virtual screening show that there is no universal best method. This includes both classification and regression based approaches. Even when the task is to predict binding energy of protein-ligand complexes using the same family of scoring functions, say empirical techniques, there are cases where some functions do better and in other scenarios those functions are outperformed by others. Here we will review the most relevant comparative studies that have assessed the performance of different prediction methodologies in virtual screening and drug design.

Cheng and co-workers recently conducted an extensive test of 16 scoring functions employed in mainstream docking tools and researched in academia [32]. The team analyzed the performance of each scoring function in predicting the binding energy of protein-ligand complexes whose high resolution structures and binding constants are experimentally known. The main test set used in this study consisted of 195 diverse protein-ligand complexes and four other protein specific sets. Considering different evaluation criteria and test datasets, they concluded that no single scoring function was superior to others in every aspect. In fact, the best scoring function in terms of predicting binding constants that are most correlated to the experimentally calculated ones was not even in the top five when the goal was to identify the correct binding pose of the ligand. These findings agree to some extent with an earlier study conducted by Wang et al. that considered very similar data sets and scoring functions [33].

In both studies mentioned above, scoring functions examined were force-field based, empirical, or knowledge based, but none based on sophisticated machine learning algorithms. Recently, Ballester and Mitchell took advantage of the diverse test set collected by Cheng et al.

as a benchmark to evaluate their RF-Score scoring function [30]. It is a random forest model fitted to geometric features that count the number of occurrences of particular atom pairs within a predefined distance [34].The results of this work indicate that the random forest model is significantly more accurate compared to the 16 scoring functions considered in Cheng et al.'s work in predicting the binding energies of 195 diverse complexes. In quantitative terms, RF-Score predictions of binding affinities have a Pearson correlation coefficient of R_P = 0.776 with the actual values, whereas the correlation coefficient of the top scoring function among the 16 mentioned earlier is just 0.644. It should be noted that this high gain in performance is mainly a result of the characteristic of the modeling approach rather than the data itself, since both sets of scoring functions were trained and evaluated on identical training and test datasets.

The usage of machine learning methods in identifying lead drugs is not just restricted to the scoring task which is a crucial step in molecular docking. In fact, machine learning methods have also been applied for QSAR modeling, where the modeling is exclusively based on ligand molecular properties. As an example, Svetnik and co-workers studied random forests' ability in predicting a compound's quantitative and/or categorical biological activity [35]. Their random forest model and other machine learning schemes were trained and evaluated on six QSAR publicly-available datasets. Datasets such as BBB [36], Estrogen [37], P-gp [38], and MDRR [39] were treated as classification problems, while Dopamine [40] was treated as a regression problem and COX-2 [41] was present in both contexts. The researchers reported the accuracy in classification and the correlation coefficient in regression for the modeling algorithms: random forests (RF), decision trees (DT), partial least squares (PLS) [42], [43], artificial neural networks (ANN), support vector machines (SVM), and k-Nearest neighbors (kNN). On both classification and regression datasets, RF models were always among the top performers. SVM and an

enhanced version of kNN have shown satisfactory performance, where in some tests they outperform the random forest approach.

Plewczynksi and co-workers carried out an assessment study for prediction capabilities of the models: SVM, ANN, NB, kNN, RF, DT, and trend vectors (TV) [44]. These machine-learning techniques are used in the context of classification where a large database of compounds is partitioned into active and inactive compounds for five biological targets. The targets considered are HIV-reverse transcriptase, COX-2, dihydrofolate reductase, estrogen receptor, and thrombin which were retrieved from the MDDR (MDL Drug Data Report) database [45]. For each target, several ligands known to be active are included in the compound database to be screened. The team has shown that different QSAR methods exhibit different strengths and weaknesses. For example, SVM was found to be superior when the goal is to retrieve high number of true positives. When the goal was to identify a small subset with the highest enrichment, TV was found to be the most suitable method. When the aim was to increase precession, TV, RF, and SVM are recommended. SVM, kNN, and RF achieve high recall (measures the percentage of correct predictions) values. Thus, the research results recommend that the choice of the appropriate classification model should be application dependent.

Other studies have been conducted that compare different nonlinear regression methods in virtual screening. For example, in a comparison between ANN and SVM to predict the apoptosis-inducing activity of several 4-aryl-4-H-chromenes, Fatemi et al. demonstrated that SVM shows more accurate predictions than ANN [46]. Goodarzi and co-workers reached also to the same conclusion when they considered SVM and ANN capabilities to predict the inhibitory activity of glycogen synthase kinase-3β [47]. SVM was also the winner when it was evaluated against kNN to identify inhibitors for histone deacetylases (HDAC) [48].

There is a plethora of other comparative assessment studies carried out to address different QSAR modeling and ligand-based drug design techniques [49], [50], [51], [52], [53], [54], [55], [56]. However, to our knowledge, there are no such comparative studies conducted for machine learning-based scoring functions. Therefore, in this work we aim to pragmatically analyze and compare several well-studied machine learning approaches in the context of molecular docking and binding affinity scoring.

Prior to this work, S.K. Namilikonda and N.R. Mahapatra in collaboration with L.A. Kuhn and with assistance from M.I. Zavodszky had performed preliminary investigations with a view toward improving the original X-Score [57] and original AffiScore [11], [57], [58], [59], [60] scoring functions. They performed feature subset selection separately on X-Score features and AffiScore features to design two sets of multiple linear regression (MLR) based empirical scoring functions: one based on a subset of X-Score features and the other based on a subset of AffiScore features. This investigation was done using the 2005 PDBbind database [61] and yielded some improvements in scoring accuracy. The setup used in this prior work served as a helpful starting point for this thesis work. However, the focus and scope of our work are different. All our experiments have been performed using the 2007 and 2010 PDBbind refined sets. In the initial stages of this thesis research, feature subset selection was comprehensively explored using not only X-Score and AffiScore features, but also RF-Score features [30] and their combinations. For the larger datasets (2007 and 2010 PDBbind refined sets) that we used, feature subset selection, in the main, was found to be not necessary to obtain improved MLR-based empirical scoring functions from X-Score, AffiScore, and RF-Score features and their combinations. Additionally, it was quite computationally time-intensive to perform feature subset selection on larger number of features. Therefore, we do not report any results for feature

subset selection in this work. More importantly, the focus in this work is on machine-learning based scoring functions in contrast to MLR-based empirical scoring functions that have been studied in the past.

1.4 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 describes the compound database used in our study and presents the tools used to extract the features that characterize interaction factors between proteins and their ligands. The other half of the chapter explains the mechanics of each machine-learning algorithm considered in this work.

Chapter 3 starts with a discussion of the common evaluation criteria employed to judge the performance of scoring functions. These criteria are then investigated in subsequent sections where several experiments addressing different aspects are conducted. Chapter 4 concludes this work and discusses some potential future improvements.

CHAPTER 2

Materials and Methods

In this chapter, we will discuss the materials and methods used to build and evaluate different scoring functions. First, we will introduce the protein-ligand complex database that serves as our data source. Then, as sketched out in Figure 2.1, we will discuss the features used to characterize the 3D structure of a protein-ligand complex and how they are calculated. Essentially, this step transforms complex information from geometrical coordinate representation into tuples of features. These features will be used to construct training and testing datasets for predictive learning models.

The second half of this chapter will present the machine-learning methods that will be trained and evaluated on the dataset alluded to above. We will explain the mathematical models behind the prediction algorithms and the software packages that implement them. Finally, we will list the 16 existing scoring functions that will be compared against the machine-learning models considered in this work.

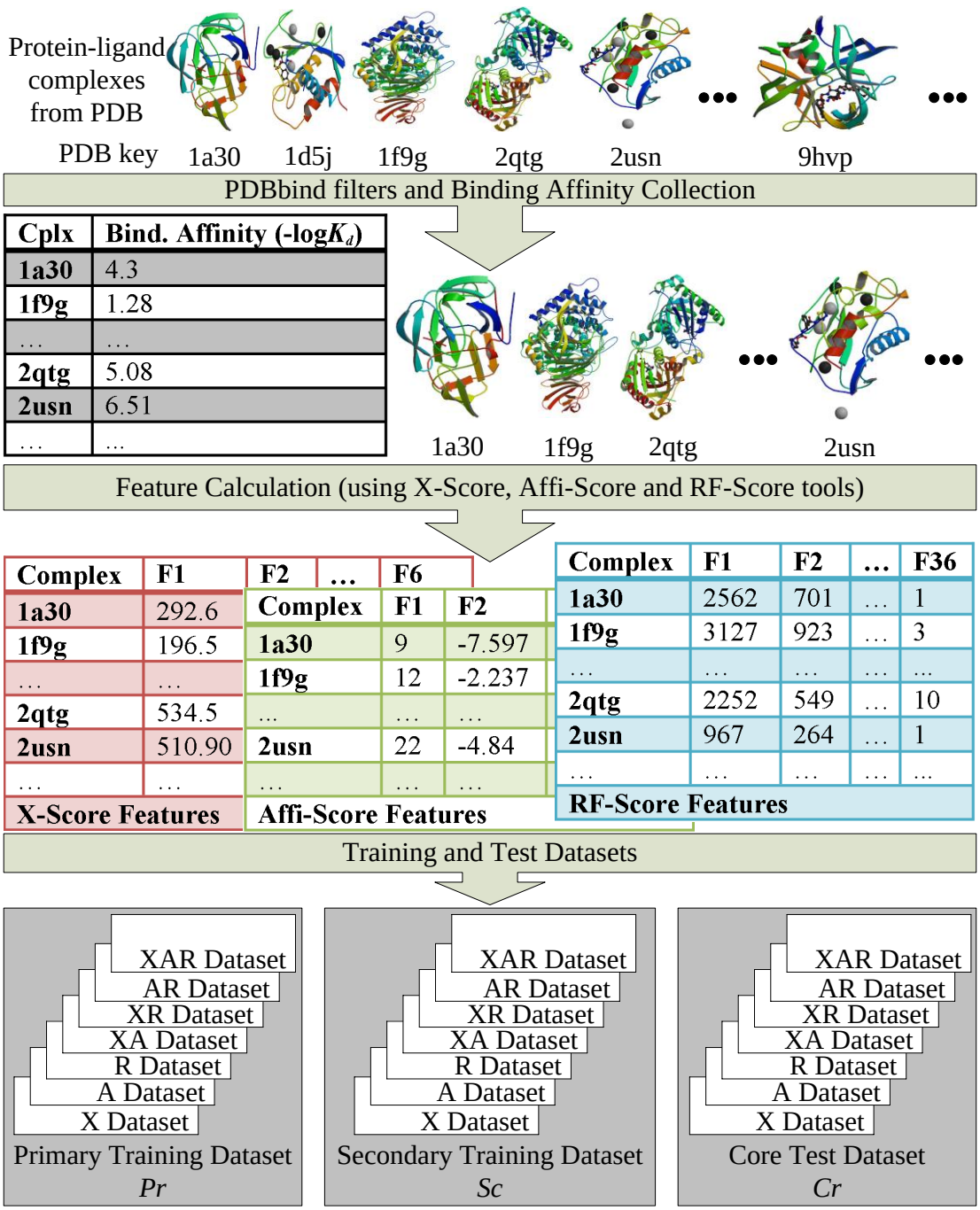


Figure 2.1 Data preparation workflow.

2.1 Compounds Database

We used the same complex database that Cheng et al. benchmarked in their recent comparative assessment of sixteen popular scoring functions [32]. They obtained the data from PDBbind, which is a selective compilation of the Protein Data Bank (PDB) database [61], [31]. Both databases are publicly accessible and continually updated. The PDB is periodically mined and only complexes that can be utilized in drug discovery are filtered into the PDBbind database.

PDBbind has been used as a benchmark for many scoring-function related studies due to its diversity, reliability, and high-quality [26], [30], [62], [63], [64], [65], [66]. In PDBbind, a number of filters are imposed to obtain high-quality protein-ligand complexes with both experimentally known binding affinity and three-dimensional structure from PDB. First, complexes that consist of only one protein and one valid ligand attached to it are collected. The validity of a ligand is determined by its molecular weight (which should not exceed 1000) and the number of its non-hydrogen atoms (that must be 6 or more). The binding affinities of these protein-ligand complexes are then assembled from the literature. The binding affinities considered are dissociation constant (K_d), inhibition constant (K_i), and concentration at 50% inhibition (IC_{50}). For these measurements to be accepted, they must be reported at normal room temperature and neutral pH values.

Second, a high-quality subset of the complexes identified in the first step is compiled into a list referred to as the *refined-set*. Each protein-ligand complex in the refined-set must meet the following requirements: (i) it must have a stable experimental binding affinity value such as K_d or K_i; complexes with only known IC_{50} values are excluded due to its dependence on binding

assay conditions; (ii) only one ligand is bound to the protein; (iii) protein and ligand molecules must be non-covalently bound to each other; (iv) the resolution of the complex crystal structure does not exceed 2.5 Angstrom (Å); and (v) for compatibility with molecular modeling software, neither the residues in the protein binding pocket nor the ligand are allowed to have organic elements other than C, N, O, P, S, F, Cl, Br, I, and H.

Finally, hydrogenation of both molecules in complex structures is carried out using Sybyl tool [67]. Protonation and deprotonation of specific groups were performed for proteins and ligands under neutral pH scheme.

The PDBbind curators compiled another list out of the refined set. It is called the *core-set* and is mainly intended to be used for benchmarking docking and scoring systems. The core set is composed of diverse protein families and diverse binding affinities. BLAST was employed to cluster the refined set according to the protein sequence similarity with a 90% cutoff [68]. From each resultant cluster, three protein-ligand complexes were chosen to be its representatives in the core set. A cluster must fulfill the following criteria to be admitted into the core set: (i) it has at least four members, (ii) the binding affinity of the highest-affinity complex must be at least 100-fold of the complex of the lowest one. The representatives were then chosen based on their binding affinity rank, i.e., the complex having the highest rank, the middle one, and the one with the lowest rank. The approach of constructing the core set guarantees unbiased, reliable, and biologically rich test set of complexes.

In order to be consistent with the comparative framework used to assess the sixteen scoring functions mentioned above, we consider the 2007 version of PDBbind. This version has a 1300-complex refined set and a 195-complex core-set (with 65 clusters.) We also take advantage of the

newly deposited complexes in the 2010 version of the database. Considering larger samples of data makes our conclusions regarding performance of scoring functions in some experiments more statistically sound. The refined set of 2010 PDBbind has 2061 entries, from which the 2007 refined list is a subset. A small portion of the 2007-version set is not included in the newly released one due to some modifications in the filtering criteria.

2.2 Compounds Characterization

To accurately estimate the binding affinity of a protein-ligand complex, one needs to collect as much useful information about the compound as possible. We attempt here to satisfy this requirement by considering features from two different scoring families. The first is empirical scheme where extracted features correspond to energy terms. The second is knowledge-based that mainly employ QSAR descriptors. As for empirical-based features, X-Score and AffiScore algorithms are used to calculate the energy terms [11], [57], [58], [59], [60]. X-Score is implemented by Wang et al. as a standalone scoring function, whereas AffiScore is developed for the SLIDE docking tool, and it supports standalone scoring mode as well. Knowledge-based geometrical descriptors were obtained from a tool developed by Ballester and Mitchell. It serves the same purpose for their RF-Score scoring function [30]. X-Score, AffiScore (SLIDE), and RF-Score are publicly available and in this section we briefly discuss the types of features extracted by each one of them. For a detailed review of these tools, we refer the interested reader to the relevant work conducted by their authors [11], [30], [57], [58], [59], [60].

X-Score utility outputs the binding free energy for a given three-dimensional protein-ligand structure as a sum of weighted terms. The terms calculated in X-Score are: (i) van der Waals interactions between the ligand and the protein; (ii) hydrogen bonding between the ligand and the protein; (iii) the deformation effect which is expressed as a contribution of the number of rotors (rotatable bonds) for the ligand (the protein contribution to this term is ignored); and (iv) the hydrophobic effect. The last term is computed out in three different ways, which in turn leads to three different versions of scoring functions. One version models the hydrophobic effect by counting pairwise hydrophobic contacts between the protein and the ligand. The second one

computes this term by examining the fit of hydrophobic ligand atoms inside the hydrophobic binding site of the protein. The last version of the scoring function accounts for hydrophobicity by calculating the hydrophobic surface area of the ligand buried upon binding. We consider the three versions of the hydrophobic interaction as distinct features and accordingly we obtain a total of six descriptors for each protein-ligand complex.

SLIDE's scoring function, AffiScore, calculates the binding free energy in a similar fashion to X-Score. AffiScore is also an empirical scoring function whose energy terms include: (i) a term capturing hydrophobic complementarity between the protein and the ligand; (ii) a polar term, which is the sum of the number of protein-ligand H-bonds, the number of protein-ligand salt-bridges, and the number of metal-ligand bonds; and (iii) the number of interfacial unsatisfied polar atoms. We did not limit ourselves to these terms only; we considered the other 25 features that AffiScore also calculates. Most of these contribute to the five main terms listed above. Table 2.1 provides a description of each term of these. The twenty-five features are not directly accounted for in the final AffiScore model. The reason is perhaps to prevent the model from overfitting. Several features are linear combinations of each other. That is in addition to a significant amount of redundancy among them. Our experiments suggest that such inter-correlation and redundancy is not a big issue for some of the machine learning models we consider.

The features listed (from X-Score and AffiScore) above are computed mathematically from the geometrical coordinates and chemical properties of protein-ligand atoms in 3D space. Usually, such sophisticated terms are naturally interpretable and have solid biochemical meaning. A simpler approach inspired from knowledge-based scoring functions does not require deriving such complex terms. Instead, a straightforward algorithm outputs a number of terms,

each of which represents number of occurrences of a certain protein-ligand atom pair within a predefined distance. Atoms of nine chemical elements are considered, which translates to a total of (9 x 9 =) 81 frequencies of atom pairs. The nine common elements taken into account in protein and ligand molecules are C, N, O, F, P, S, Cl, Br, and I. As pointed out by the algorithm's author (and confirmed by our experiments), 45 of the resultant features are always zero across all PDBbind compounds. Such terms were discarded and we only include nonzero atom-pair number of occurrences (36 terms) to achieve maximal data compactness, see Table 2.1. The distance cutoff we use here is 12Å. This value is recommended by PMF [24] for being adequately large to implicitly capture solvation effects [30].

Table 2.1 Protein-ligand interactions used to model binding affinity.

X-Score Features	
Feature name	**Description**
vdw_interactions	van der Waals interactions between the ligand and the protein.
hyd_bonding	Hydrogen bonding between the ligand and the protein.
frozen_rotor	The deformation effect, expressed as the contribution of the number of rotors for the ligand.
hphob_p	Hydrophobic effect: accounted for by counting pairwise hydrophobic contacts between the protein and the ligand.
hphob_s	Hydrophobic effect: accounted for by examining the fit of hydrophobic ligand atoms inside hydrophobic binding site.
hphob_m	Hydrophobic effect: accounted for by calculating the hydrophobic surface area of the ligand buried upon binding.
AffScore Features	
Feature name	**Description**
number_of_ligand_nonh_atoms	Number of non-hydrogen atoms in the ligand.
total_sum_hphob_hphob	The total sum of hydrophobicity values for all hydrophobic protein-ligand atom pairs.
total_diff_hphob_hphob	The total difference in hydrophobicity values between all hydrophobic protein-ligand atom pairs.

Table 2.1 (cont'd)

total_diff_hphob_hphil	The total difference in hydrophobicity values between all protein-ligand hydrophobic/hydrophilic mismatches.
total_sum_hphil_hphil	The total sum of hydrophobicity values for all hydrophilic protein-ligand atom pairs.
total_diff_hphil_hphil	The total difference in hydrophobicity values between all hydrophilic protein-ligand atom pairs.
number_of_salt_bridges	The total number of protein-ligand salt bridges.
number_of_hbonds	The total number of regular hydrogen bonds.
number_of_unsat_polar	The total number of uncharged polar atoms at the protein-ligand interface which do not have a bonding partner.
number_of_unsat_charge	The total number of charged atoms at the protein-ligand interface which do not have a bonding partner.
ratio_of_interfacial_ligand_heavy_atoms	The fraction of ligand heavy atoms (non-hydrogen) which are at the protein-ligand interface (defined as atoms within 4.5 Angstrom of opposite protein or ligand atoms).
number_of_interfacial_ligand_atoms	The total number of ligand atoms at the interface.
number_of_exposed_hydro_ligand_atoms	The number of hydrophilic ligand atoms which are not at the interface.
number_of_ligand_flexible_bonds	The number of rotatable single bonds in the ligand molecule.
number_of_all_interfacial_flexible_bonds	The number of protein and ligand rotatable single bonds at the interface.
total_avg_hphob_hphob	The sum of the average hydrophobicity values for all ligand atoms. The average hydrophobicity value for the ligand atoms was determined by calculating the average hydrophobicity value of all neighboring hydrophobic protein atoms.
old_slide_total_sum_hphob_hphob	The sum of hydrophobicity values for all hydrophobic protein-ligand atom pairs (as computed in the old AffiScore version).
number_of_flexible_interfacial_ligand_bonds	Number of rotatable single bonds of the ligand at the interface.
total_hphob_comp	The sum of the degree of similarity between protein-ligand hydrophobic atom contacts. A higher value indicates the protein and ligand atoms had more similar raw hydrophobicity values.

Table 2.1 (cont'd)

total_hphob_hphob_comp	A normalized version of the degree of similarity between the ligand atoms and their hydrophobic protein neighbors. This is the version currently used in Affiscore.
contact_hphob_hphob	The pairwise count of hydrophobic-hydrophobic protein-ligand contacts.
contact_hphil_hphil	Pairwise count of hydrophilic-hydrophilic protein-ligand contacts.
contact_hphob_hphil	Pairwise count of hydrophobic-hydrophilic protein-ligand contacts.
total_target_hphob_hphob	Total of protein hydrophobicity values for protein atoms involved in hydrophobic-hydrophobic contacts.
increase_hphob_environ	For hydrophobic/hydrophilic mismatch pairs, sum of the hydrophobic atom hydrophobicity values.
total_target_hydro	Total hydrophilicity of all protein interfacial atoms.
norm_target_hphob_hphob	A distance normalized version of total_target_hphob_hpob
total_target_hphob_contact	Total of protein hydrophobicity values for protein atoms at the interface.
norm_target_hphob_contact	A distance normalized version of total_target_hphob_contact.
Total_ligand_hydro	Total of ligand hydrophilicity values for interfacial ligand atoms.
RF-Score Features	
Feature Name	**Description** (The number of protein-ligand …)
C.C	Carbon-Carbon contacts.
N.C	Nitrogen-Carbon contacts.
O.C	Oxygen-Carbon contacts.
S.C	Sulfur-Carbon contacts.
C.N	Carbon-Nitrogen contacts.
N.N	Nitrogen-Nitrogen contacts.
O.N	Oxygen-Nitrogen contacts.
S.N	Sulfur-Nitrogen contacts.
C.O	Carbon-Oxygen contacts.
N.O	Nitrogen-Oxygen contacts.
O.O	Oxygen-Oxygen contacts.
S.O	Sulfur-Oxygen contacts.
C.F	Carbon-Fluorine contacts.
N.F	Nitrogen-Fluorine contacts.
O.F	Oxygen-Fluorine contacts.
S.F	Sulfur-Fluorine contacts.

Table 2.1 (cont'd)

C.P	Carbon-Phosphorus contacts.
N.P	Nitrogen-Phosphorus contacts.
O.P	Oxygen-Phosphorus contacts.
S.P	Sulfur-Phosphorus contacts.
C.S	Carbon-Sulfur contacts.
N.S	Nitrogen-Sulfur contacts.
O.S	Oxygen-Sulfur contacts.
S.S	Sulfur-Sulfur contacts.
C.Cl	Carbon-Chlorine contacts.
N.Cl	Nitrogen-Chlorine contacts.
O.Cl	Oxygen-Chlorine contacts.
S.Cl	Sulfur-Chlorine contacts.
C.Br	Carbon-Bromine contacts.
N.Br	Nitrogen-Bromine contacts.
O.Br	Oxygen-Bromine contacts.
S.Br	Sulfur-Bromine contacts.
C.I	Carbon-Iodine contacts.
N.I	Nitrogen-Iodine contacts.
O.I	Oxygen-Iodine contacts.
S.I	Sulfur-Iodine contacts.

2.3 Binding Affinity

Dissociation constant (K_d) is commonly used to quantify the strength with which a protein binds with a ligand. This equilibrium constant measures the tendency for a complex to separate into its constituent components. We utilize it here to describe the degree of tightness of proteins to their binders. That is, by interpreting complexes whose components are more likely to dissociate (high dissociation constants) as loosely bound (low binding affinities) and vice-versa. We can write the binding reaction for a protein-ligand complex as:

$\text{Protein-Ligand}_{\text{complex}} \leftrightarrow$ Protein + Ligand,

and the dissociation constant in terms of mol/L is given by:

K_d = [P] [L] / [PL],

where [P], [L], and [PL] are the equilibrium molar concentrations for the protein, ligand, and protein-ligand complex; respectively.

The higher the concentration of the ligand needed to form a stable protein-ligand complex, the lower the binding affinity will be. Put differently, as the concentration of the ligand increases, the value of K_d also increases and this corresponds to a weaker binding.

In some experimental settings, such as competition binding assays, another measure of binding affinity is widely employed. This measure is known as *inhibitor constant* and denoted by K_i. Inhibitor constant is defined as the concentration of a competing ligand in a competition

assay that would occupy 50% of the receptor in the absence of radioligand. Inhibitor constant is expressed by the following Cheng-Prusoff equation [69]:

$K_i = IC_{50}/(1+[L] / K_d)$,

where IC_{50} is the half maximal inhibitory concentration that measures the effectiveness of the ligand to inhibit a biochemical or biological function by half, K_d is the dissociation constant of the radioligand, and [L] is the concentration of the ligand L. IC_{50} could also be thought of as the concentration of the ligand required to displace 50% of the radioligand's binding. IC_{50} value is typically determined using *dose-response curve* [70] and it should be noted that it is highly dependent on experimental conditions under which it is calculated. To overcome such dependency, the inhibitor constant formula K_i is developed such that it has an absolute value. An increase in its value means a higher concentration of an inhibitor needed to block certain cellular activity.

It is customary to logarithmically convert the dissociation and inhibitory constants to numbers that are more convenient to interpret and deal with. These constants are scaled according to the formula:

$pK_d = -\log(K_d)$, $pK_i = -\log(K_i)$,

where higher values of pK_d and pK_i reflect an exponentially greater binding affinity. The data we use in our work are restricted to complexes whose either K_d or K_i values are available. Therefore, all regression models that will be built are fitted so that the response variable is either

pK_d or pK_i. In terms of pK_d and pK_i, it is worth knowing that the binding affinities of the complexes in PDBbind database range from 0.49 to 13.96, spanning more than 13 orders of magnitude.

2.4 Training and Test Datasets

Several training and test datasets are used in our experiments in order to evaluate scoring functions in various ways. As discussed in Section 2.1, two versions of the PDBbind database are employed in our study. We consider the "*refined*" and "*core*" sets of the 2007 version that include 1300 and 195 complexes, respectively. The latest version, 2010, whose refined set contains 2061 complexes is also used. For each complex in the three complex sets, we extracted its X-Score, AffiScore, and RF-Score features. Then out of these features, many datasets were constructed that serve different functions (i.e., training, testing, or both); see Figure 2.1. Each dataset can be thought as a table of several rows and columns. Each row represents a unique protein-ligand complex and each column corresponds to a particular feature.

One dataset derived from the 2007 refined set is referred to as the *primary* training set and we denote it by *Pr*. It is composed of the refined complexes of 2007, excluding those complexes present in the core set of the same year's version. There are also nine complexes in the refined set that we excluded in the primary training set due to difficulty in generating their descriptors. Thus the primary training set consists of an overall (1300 - 195 - 9 =) 1096 complexes.

We also extracted features for the complexes in the core set and we stored them in a dataset called the *core* test set which is denoted by *Cr*. The number of complexes in this dataset is 192, as we were not able to extract features for one protein-ligand complex. Therefore that problematic complex and the other two in the same cluster were not included (195 - 3 = 192) in the core set.

The last main dataset was built from complexes in the refined set of the 2010 version of PDBbind database. We refer to this set as the *secondary* training set and it is denoted by *Sc*. Not

all of the 2061 complexes in the 2010 refined set were compiled to the *Sc* dataset. That is because there were several complexes that could not be processed by feature-extraction tools. Furthermore, the complexes that appear in the core set *Cr* (of the 2007 version) were not considered in *Sc*. As a result, the total number of records (complexes) in the *Sc* dataset reduced to 1986. It should be noted that the dataset *Pr* is a subset of *Sc* (except, as noted in Section 2.1, some complexes present in *Pr* are not included in *Sc* due to modifications in the filtering criteria in 2010).

Table 2.2 Training and test sets used in the study.

Data Set **Feature Set**	**Primary (2007)** ***Pr***	**Secondary (2010)** ***Sc***	**Core (2007)** ***Cr***	**# of Features**
X-Score	Pr_X	Sc_X	Cr_X	6
AffiScore	Pr_A	Sc_A	Cr_A	30
RF-Score	Pr_R	Sc_R	Cr_R	36
X-Score AffiScore	Pr_{XA}	Sc_{XA}	Cr_{XA}	36
X-Score RF-Score	Pr_{XR}	Sc_{XR}	Cr_{XR}	42
AffiScore RF-Score	Pr_{AR}	Sc_{AR}	Cr_{AR}	66
X-Score AffiScore RF-Score	Pr_{XAR}	Sc_{XAR}	Cr_{XAR}	72
Number of Complexes	1096	1986	192 (64 clstr)	

For each dataset, *Pr*, *Sc*, and *Cr*, seven versions are generated (see the white boxes toward the bottom of Figure 2.1). Each version is associated with one feature type (or set). These types refer to features extracted using X-Score, AffiScore, RF-Score, and all their possible combinations (which are seven). For instance, features extracted using X-Score and RF-Score tools for complexes in *Pr* set, when combined in one dataset we obtain the version Pr_{XR} of *Pr*. Note also that we distinguish versions from each other using subscripts that stand for the first letter(s) of the tool(s) employed to calculate the respective feature set. One should be careful that

features generated using the tool RF-Score does not necessarily mean that only Random Forest-based scoring function will be applied to such descriptors. In fact, we assume these features can be employed to build any machine learning-based scoring function. This is the case even though such features were originally used by Ballester and Mitchell in conjunction with Random Forests to build their RF-Score scoring function. Table 2.2 summarizes the datasets discussed above and the numbers of their constituent features and complexes.

2.5 Machine Learning Methods

The collected protein-ligand training data is assumed to be independently sampled from the joint distribution (population) of (X,Y) and consists of N of (p+1)-tuples (or vectors)$\{(X_1,Y_1),.....,(X_i,Y_i), \ldots, (X_N,Y_N)\}$. Here X_i is the i^{th} descriptor vector that characterizes the i^{th} protein-ligand complex. Y_i is the binding affinity of the i^{th} complex. These tuples form training and test datasets on which the machine learning models of interest will be fitted and validated. A dataset can be denoted more concisely by:

$$D = \{X_i, Y_i\}_{i=1}^{N},\ X_i \in R^P, Y_i \in R.$$

We categorize the models to be compared into single machine learning models and ensemble learners. The single models investigated in this work are Multiple Linear Regression (MLR), Multivariate Adaptive Regression Splines (MARS), k-Nearest Neighbors (kNN), and Support Vector Machines (SVM). Ensemble models include Random Forests (RF), Neural Forests (NF), and Boosted Regression Trees (BRT). Consensus models from single learners, ensemble learners and both will also be briefly investigated at the end of the thesis.

2.5.1 Single Models

In this category, we construct one parametric model which is MLR and three non-parametric models: MARS, kNN and SVM. Generally single-learner algorithms are faster to train than their ensemble counterparts, and require less memory from the system. Besides they tend to have more intuitive descriptive power at the cost of less predictive performance.

2.5.1.1 Multiple Linear Regression (MLR)

Multiple linear regression (MLR) model is built based on the assumption that there exists a linear function relates explanatory variables (features) *X* and the outcome *Y* (binding affinity). This linear relationship takes the form:

$$\widehat{Y} = f(X) = \widehat{\beta}_0 + \sum_{j=1}^{P} \widehat{\beta}_j X_j . \quad (2.1)$$

MLR algorithm attempts to estimate the coefficients $\beta_0..\beta_P$ from training data by minimizing the overall deviations of fitted values from measured observations. Such fitting methodology is widely known as linear lest-squares. Despite the fact that MLR is simple to construct and fast in prediction, it suffers from overfitting when tackling high-dimensional data. Moreover, linear models have a relatively large bias error due to their rigidity. The reason for developing simple linear models in this study is two-fold. First, they are useful in estimating the behavior of conventional empirical scoring functions. Second, they serve as base-line benchmarks for the nonlinear models we investigate.

2.5.1.2 Multivariate Adaptive Regression Splines (MARS)

Instead of fitting one linear spline to training data as in MLR, a more flexible approach is to fit several additive splines. Such a technique, known as Multivariate Adaptive Regression Splines (MARS), was proposed by Friedman two decades ago to solve regression problems [71]. Nonlinearity in MARS is automatically addressed by partitioning the feature space into *S* regions each of which is fitted by one spline. The final nonlinear model is obtained by truncating together the *S* resulting splines (or terms). Every term is composed of one or more two-sided

basis functions that are generated in pairs from the training data. Each pair (a.k.a. hinge function) is characterized by an input variable X_j and a *knot* variable *k*. The knot variable acts as a pivot around which one function pair has a positive value from one side, and zero in the other side. More formally:

$$h(X_j, k)_+ = \max(0, X_j - k) \quad and \quad h(X_j, k)_- = \max(0, k - X_j) \tag{2.2}$$

The +/- sign in the subscripts denotes that the function has the value of $X_j - k / k - X_j$ right/left to the knot and 0 to its left/right. MARS algorithm adaptively determines both the number of knots and their values from the training data. It also employs least-squares to optimize the weights $\hat{\beta}_i$ associated with each term (*i=1..S*) in the final MARS model that can be written as:

$$\hat{Y} = f(X) = \hat{\beta}_0 + \sum_{i=1}^{S} \hat{\beta}_i H_i(X) \tag{2.3}$$

where $\hat{Y}$ is the estimated binding affinity for the complex *X*, $\hat{\beta}_0$ is the model's intercept (or bias), and *S* is the number of splines (or terms). Note that the function *f(x)* can be either additive combination of "piecewise" splines or interactive "pairwise" mix of hinge functions, or a hybrid of both. Thus each H_i term is expressed as

$$H_i(X) = \prod_{l}^{d} h_l(X, k) \qquad 1 \le d \le 2NP \tag{2.4}$$

The value of one for *d* implies that the term H_i is a linear spline and contributes in an additive fashion to the final model, whereas $d \geq 2$ implies an interaction depth of *d* and thus the corresponding term is nonlinear.

MARS algorithm firstly builds a complex model consisting of a large number of terms. To prevent the model from overfitting, this step is followed by a pruning round where several terms are dropped. An iterative backward elimination according to generalized cross-validation (GCV) criterion is typically performed to determine which terms to be removed. The GCV statistic penalizes models with large number of terms and high degree when their (least-squares) goodness-of-fit are measured. During the training process, a greedy search is performed in both the forward (terms adding) and backward (terms elimination) passes. MARS models investigated in this work were allowed to grow the number of terms in the forward pass (before pruning) as many as the number of descriptors plus one, or 21, which larger -- i.e. max(P+1, 21).

The implementation of MARS algorithm used in this comparison is a publicly available R [72] package. The package is named EARTH (version 2.4-5) which provides default values for all parameters [73]. We have used these values except for the degree of each term and the optimal GCV penalty whose optimal values are determined using 10-fold cross-validation. More detailed discussion about the optimization procedure is presented in Section 3.2.

2.5.1.3 k-Nearest Neighbors (kNN)

An intuitive strategy to predict the binding energy of a protein-ligand complex is to scan the training complex database for a similar protein-ligand compound and outputs its binding affinity as estimation for the complex in question. The most similar complex is referred to as the nearest neighbor to the protein-ligand compound we attempt to predict its binding energy. This simple algorithm is typically augmented to take into account the binding affinities of k nearest neighbors, instead of just the nearest one. In the kNN algorithm used here, each neighbor contributes to the complex binding energy by an amount proportional to its distance from the

target complex. The distance measure employed in such algorithm is Minkowski distance which is expressed as:

$$D(X_i, X_j) = \left[\sum_{m=1}^{p} (x_{im} - x_{jm})^q \right]^{1/q} \quad (2.5)$$

where the final kNN model is established as:

$$\hat{Y} = f(X) = \frac{1}{K} \sum_{i=1}^{K} W(D, X, X_i) Y_i \quad (2.6)$$

where X and X_i are the target and the i^{th} neighbor complexes; respectively. The weighting method W is a function of the distance metric D. The q parameter in Equation 2.5 determines the distance function to be calculated. Euclidean distance is obtained when q=2, whereas Manhattan distance is realized when q is set to 1. Both the number of nearest neighbors and the Minkowski parameter q are user defined. We ran a 10-fold cross-validation to obtain the optimal values of these two parameters that minimize the overall mean-squared error. No feature selection has been conducted since the number of features we use is not very large.

It should be noted that the learning process in kNN is lazy local. It is lazy because the learning process is differed until a complexes' binding affinity needs to be predicted. And local because only a subset of the training instances, the neighbors, is involved in the final output as opposed to a global participation of the whole training dataset.

We use the R package KKNN (version 1.0-8) that implements a weighted version of k-Nearest Neighbors [74]. It is publically available and also supported by a publication [75].

2.5.1.4 Support Vector Machines (SVM)

Support Vector Machines is a supervised learning technique developed by Vapnic et al. to solve classification and regression problems [76]. The first step in constructing an SVM model is to map data to high-dimensional space using a nonlinear mapping function. Several such transforming functions are employed by SVM algorithms and the most popular two are polynomial and radial basis functions. In this work we choose radial basis function [RBF] (kernel) due to its effectiveness and efficiency in the training process [77]. Another attractive feature of this kernel is that it has only one parameter, namely the radial basis width, which can be calibrated to reflect the layout and distribution of the training data.

After mapping to the new high-dimensional space, a linear model can be constructed. In regression formulation, SVM algorithms induce from training data an optimal hyperplane that approximates the underlying relationship *f(x)* between the covariates (in the new feature space) and the output variable. The predicting function *f(x)* can be more formally defined as:

$$\hat{Y} = f(X) = \langle w, G(X) \rangle + b \tag{2.7}$$

where $\langle , \rangle$ represents the dot product and *G(X)* is a nonlinear function that maps the input vector to another feature space. The rightmost term *b* is the model's bias. The norm of vector *w* is minimized in order for the model to generalize well for unseen data. At the same time the model should fit the training data by minimizing the ε-insensitive loss function that is defined as,

$$L_{\varepsilon}(X) = \max(0, |y - f(X)| - \varepsilon) \tag{2.8}$$

This loss function defines a region around the hyperplane out of which lying data points are penalized. The penalty is introduced by slack variables ξ, ξ^*. SVM algorithm therefore can be formulated as an optimization problem that minimizes the functional:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{2.9}$$

where *C* is a user defined parameter. This equation can be solved more easily by applying quadratic programming on its dual formulation. The solution to Equation 2.9 is henceforth given by,

$$\max_{\alpha,\alpha^*} W(\alpha,\alpha^*) = \max_{\alpha,\alpha^*} \sum_{i=1}^{N} \alpha_i^*(y_i - \varepsilon) - \alpha_i(y_i + \varepsilon) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(X_i, X_j) \tag{2.10}$$

$$\text{subject to the constraints, } \begin{array}{l} \alpha_i, \alpha_i^* = [0, C], \quad i = 1,\ldots, N \\ \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right) = 0 \end{array} \tag{2.11}$$

where α_i and α_i^* are Lagrange multipliers that are obtained by solving Equation 2.10 under the constraints given in Equation 2.11. After obtaining Lagrange multipliers we can calculate the right hand terms of Equation 2.7 as follows,

$$\langle w, G(X)\rangle = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)K(X, X_i)$$

and (2.12)

$$b = -\frac{1}{2}\sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\left(K(X_i, X_r) + K(X_i, X_s)\right)$$

In our work we choose the kernel $K(X,X_j)$ to be the radial basis function (RBF) which takes the following Gaussian form,

$$K(X, \tilde{X}) = e^{-\frac{\left\|X - \tilde{X}\right\|^2}{2\sigma^2}} \quad (2.13)$$

where σ is the kernel width. The data points X_r and X_s in Equation 2.12 are any arbitrary support vectors located on both sides of the hyperplane. Support vectors are data instances located outside the ε-insensitive region, and as this region gets wider (larger ε value), fewer support vectors are obtained and the model becomes flatter. The width of the ε-tube is not the sole parameter that controls the model complexity. The variable C (Equations 2.9 and 2.11) functions as a trade-off between the model flatness and the degree to which deviations larger than ε are tolerated. Very small C values may produce under-fitted model, while too large C increases the model complexity and hence it may overfit.

Most SVM implementations treat the method's meta-parameters as user defined inputs and usually are not known beforehand. They should be carefully tuned to ensure proper model generalization and predicting performance. A 10-fold cross-validation is performed to select the optimal complexity parameter C, ε value and the width of the RBF kernel σ. Section 3.2 provides more details about the tuning procedure and the resulting optimal values.

We use the well-known SVM implementation built in the C++ library LIBSVM (version 2.6) [78]. LIBSVM empowers both supervised (classification and regression) and unsupervised learning modes. An R interface to this engine is provided by the package E1071 (version 1.5-24) [79]. Among many other features, this package allows automatic parameter selection using cross-validation, predicting new data, and provides intuitive interface to LIBSVM.

2.5.2 Ensemble Models

In the context of regression analysis, ensemble learning is based on a simple concept of combining predictions of a set of individual learners either by averaging (co-operative) or weighting (competitive). Such an approach is proved to be powerful in machine learning. In order to gain maximum performance from ensemble schemes, the constituent base learners should be accurate and diverse [80]. The first requirement is achieved by adopting nonlinear techniques such as prediction trees. As for the second requirement, several methods are proposed in the literature to maximize learners' diversity by minimizing inter-correlation among them. Most popular ones promote such diversity by manipulating the training data set from which regressors are to be induced. Each learner is fit to a data sample drawn from the main training set according to some random mechanism. Bagging, adaptive-weight-boosting of training examples, sampling without replacement, and stacking are commonly applied approaches to inject randomness whenever a tree (or any base learner) is built. Another perturbation techniques that target the input feature space are also applied in ensemble learning [81], [82], [34]. In this approach, a random subset of features is chosen each time a hypotheses is constructed. The ensemble's i^{th} learner is trained on data formed by the i^{th} random subset of the whole feature

space. In this work we will build ensemble models based on manipulating the training set and input features.

2.5.2.1 Random Forests (RF)

Random Forests is a powerful ensemble tool developed by Leo Breiman to solve classification and regression problems [34]. It has been applied in many drug-discovery related studies where it showed excellent results [30], [44], [35], [83], [84]. A Random Forests model consists of an ensemble of prediction trees (typically a few hundreds.) Each tree is fitted to a bootstrap sampled from training set. When a tree is constructed, at each node a binary-split is made on *mtry* features randomly sampled from the *P*-dimensional feature space ($mtry \leq P$). Each tree is grown independently to its full size without pruning. When the task is to predict the binding affinity of a new protein-ligand complex, the output is then the average of the predictions of the comprising individual trees. This mechanism can be formally written as:

$$\hat{Y} = f(X) = \frac{1}{L}\sum_{l=1}^{L} T_l\ (X, \theta_l) \tag{2.14}$$

where X is a protein-ligand complex whose binding affinity needs to be predicted, L is the number of generated trees, and T_l is the l^{th} tree in the ensemble. Each tree in the ensemble is characterized by the values Θ_l of its terminal nodes.

Upon building an RF model, the two parameters that must be assigned are L and *mtry*. As we pointed out earlier, *mtry* is the number of random features that need to be sampled each time a node is split. The variable *mtry* is a user defined parameter and it is practically the only variable that could be tuned. Although experience suggests that the performance of RF models is not very

sensitive to *mtry* value. This is in fact what we obtained based on search for *mtry* that minimizes the root of out-of-bag mean-squared error (OOBMSE) -- see Section 3.2 for more details. Out-of-bag refers to observations that are not sampled from the original training set when bootstrap sets are drawn. On average about 34% of the training set is left out in each tree-generating iteration. These out-of-sample examples are valuably utilized to estimate the generalization error of the RF model as follows:

$$OOBMSE = \frac{1}{N}\sum_{1}^{N}(Y_i - \bar{\hat{Y}}_{i_{OOB}})^2 \tag{2.15}$$

where $\bar{\hat{Y}}_{i_{OOB}}$ is the average predictions of the i^{th} example over all the trees in which this example is an OOB.

The second parameter in an RF model is the number of trees (*L)* grown in the forest. We fixed *L* to 1000 in our experiments; even though the OOBMSE statistic started to stabilize a few hundred trees before this value. This finding conforms to the Strong Law of Large Numbers that assures such convergence. Our finding and the supporting theory of large numbers explain why RF models do not overfit by adding more trees. Computationally wise, growing extra number of trees does not significantly hurt the algorithm's speed neither in training nor during prediction. This can be partially attributed to rapidity in trees prediction and the speed of creating regression trees with only subset of the feature space.

Besides the ease of tuning and internal unbiased general-error estimation (OOBMSE), random forests have other attractive features such as: built-in variable importance estimation, partial independence analysis, handling large number of features without degradation in

accuracy, and low bias and variance. Variable importance is a very important metric in prediction models. It allows drug designers, for instance, to identify the most critical protein-ligand interactions that provide the predictive accuracy. Breiman [34] proposed a simple method to calculate such metric using OOB examples. That is achieved by permuting input variables (one at the time) and then computing the corresponding OOBMSE (permute) after this noising step. By comparing the intact OOBMSE to thus-computed OOBMSE (permute), the percentage increase in MSE is evaluated. For each tree *t* in the forest, the regular OOBMSE is determined as follows:

$$OOBMSE_t = \frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} (Y_i - \hat{Y}_i)^2 \tag{2.16}$$

where $\tilde{N}$ is the number of OOB examples in the t^{th} tree, and the iterator *i* is their index. The criterion OOBMSE for the same tree when permuting the input variable X_j is defined as:

$$OOBMSE_t^j = \frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} (Y_i - \hat{Y}_i^j)^2 \tag{2.17}$$

For each variable X_j in each tree *t* in the forest the increase in *MSE* is evaluated. The result is then averaged over all the trees in the forest. The overall increase in MSE for an input feature X_j is therefore regarded as its influence on predicting the binding affinity.

In virtual screening, input variables can be sorted based on their influence and plotted to give a visual insight into different protein-ligand interactions. Another powerful visualization tool for drug designers provided by RF is partial dependence plots. Such plots are produced by

calculating the effect of an input variable X_j on the outcome variable while the rest of the input variables $X_{\setminus j}$ are integrated out. More formally, the dependence of the RF approximated function *f(X)* on the input variable X_j is calculated as follows:

$$\tilde{f}_j(x) = \frac{1}{N}\sum_{i=1}^{N} f(x, \mathrm{x}_{i,\setminus j}) \tag{2.18}$$

For any input value *x* for the variable X_j, the summand in (2.18) is iterated over all the data patterns (typically the training data) to account for the average effects of the rest of the input variables $X_{\setminus j}$. It should be noted that the resultant plots do not show interplays between different input variables. However, such plots are an excellent basis for interpretation.

In our experiments we use the RANDOMFOREST package (version 4.5-36) of R language [85]. The package is an interface to the RF's FORTRAN algorithm that is implemented by the technique's author Breiman and colleagues.

2.5.2.2 Neural Forests (NF)

We propose here an ensemble method based on Artificial Neural Network (ANN). The algorithm is inspired from the Random Forests approach in forming the ensemble. Instead of using decision trees as base learners, we employ here multi-layered perceptron (MLP) ANN's.

So far, the focus has been more or less biased to using decision trees in ensemble learning. Choosing trees as base learners is mainly due to their high flexibility and variance (low stability). High variance decreases inter-correlation between trees and therefore increases the overall model's accuracy.

Artificial Neural Networks, such as MLP, share several characteristics with prediction trees. They are non-parametric, non-linear and have high variance. Besides, both techniques are very fast in prediction. As for the training, neural networks are relatively slower in learning the unknown function. Our model uses hundreds of back-propagation iterations to train each neural network in the ensemble. Such neural networks are composed of *mtry* input neurons, an arbitrary number of neurons (20 in our experiments) in the hidden layer, and one output neuron for the output layer. These neurons are interconnected via weighted links as shown in Figure 2.2. The outputs of the input neurons are directed to all the neurons in the hidden layer. The outputs of the hidden layer neurons are also directed forward to the output neuron. The output of each network *l* in the ensemble is calculated at the output neuron according to the formula:

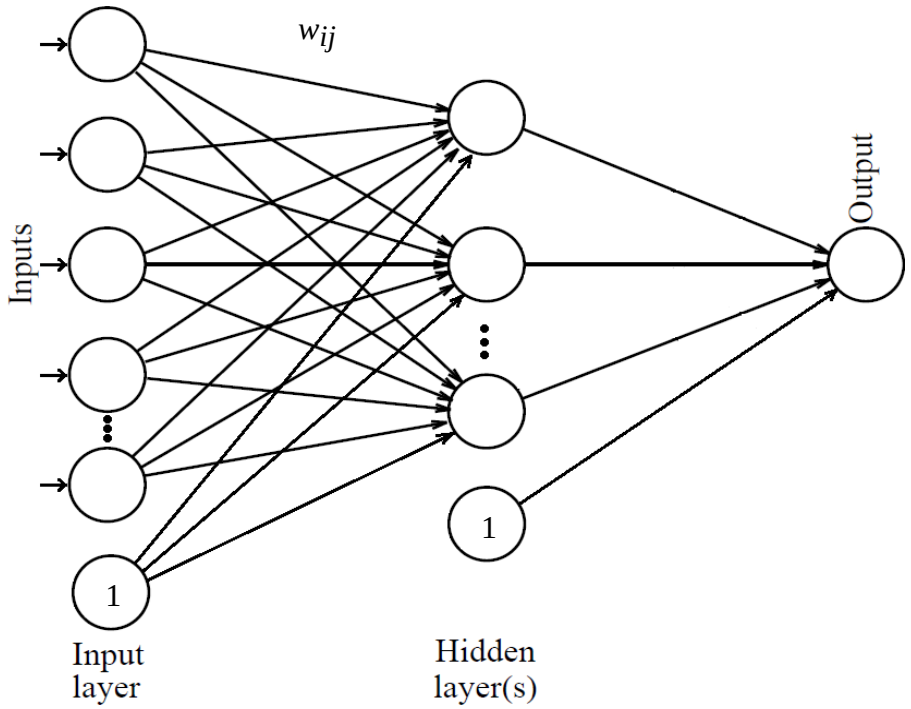


Figure 2.2 Multi-layered perceptron, feed-forward neural network

$$\hat{Y}_l = f(X) = f_o\left(\sum_h w_{ho} f_h\left(\sum_i w_{ih} x_i\right)\right) \tag{2.19}$$

where *f(X)* is the function that maps the input protein-ligand *X* to binding affinity $\hat{Y}_l$, f_o is the activation function of the output neuron (linear in our case), f_h is the activation function for the hidden-layer neurons (sigmoid in this work), w_{ho} refers to the weights associated with the links connecting the hidden to the output layer, w_{ih} represents the weights of input-to-hidden layer links, and x_i is the i^{th} component of the input pattern *X*. The network weights, w_{ih} and w_{ho}, are optimized such that they minimize the fitting criterion *E* that is defined as

$$E = (Y - \hat{Y})^2 + \lambda \sum_{ij} w_{ij}^2 \tag{2.20}$$

where Y and $\hat{Y}$ are respectively the measured and predicted binding affinities, and λ is a regularization parameter. The later parameter is also known as the weight decay and it guards weights from getting large values. Introducing the weight decay parameter avoids the scenario of saturation at the output of the hidden-layer neurons. We scaled the input variables to the range [0-1] to effectively optimize the weights when regularization is considered [86].

Learning the ensemble in NF is identical to Random Forests approach except MLP networks are fitted to the training data instead of decision trees. Each network Net_l in the ensemble is fitted to a bootstrap of the training data described by a random subset of features. The size of this subset, *mtry*, and the number of networks *L* to train are user defined variables as it is the case in RF method. After building the model, the binding affinity of a new protein-ligand complex *X* is computed by applying the formula:

$$\hat{Y} = f(X) = \frac{1}{L}\sum_{l=1}^{L} Net_l(X, w_l) \tag{2.21}$$

where each Net_l in the ensemble is defined by its weights set w_l. As given in (2.21), the mean of the outputs of these networks (2.19) is taken to calculate the final value of the binding affinity. Such value is dependent on the values of weight decay λ and *mtry*. We tuned these parameters such that the root of OOB mean-squared error (OOB RMSE) is minimized. More details about this procedure and the resulting optimal values for λ and *mtry* are given in Section 3.2.

The proposed NF model addresses the most important five issues associated with traditional MLP networks. These issues are weight optimization, dealing with high-dimensional data, training speed, model interpretability, and model's generalization. Firstly, if we were to build a scoring function based on a single MLP network, then there would always be a chance that the network weights may yield very poor predictions. That could be due to improper optimization of such weights. They may get trapped in local minima when they are optimized to minimize the loss function *E*. When generating several neural networks, the chance that all of them realize poorly optimized weights is almost zero. For instance, let us assume that there is a probability of 0.7 that weights' final values yield poor model (i.e. trapped in local minima). Then an ensemble model that uses just 20 networks will have a probability of ($0.7^{20} \approx$) zero that all networks realize such imperfect weights.

Secondly, dealing with high-dimensional data can be problematic in MLP networks. Large number of inputs means building networks with large number of neurons and links connecting them. Weights associated with such links need sufficiently large number of training patterns to reliably use the network for predicting future examples. In many cases the available training data is limited. Therefore a feature selection or reduction becomes necessary for fitting a robust MLP

model. Such preprocessing procedure may fail when, for instance, the input variables are of equal importance and/or it is desirable to use the original feature space. The Neural Forest approach in handling the input space is very effective when the data has a large number of dimensions. By selecting only a random subset of an acceptable size one can avoid the possibility of immature weights' optimization. Since this random selection of features is repeated several times (1000 in our experiments), it is very likely that all the features are taken into account in more than one network.

Thirdly, training MLP networks using back-propagation technique is a time consuming process. The time required for the weights to converge to their optimum values scales with their number. As discussed above, the number of weights is dependent on the number of input neurons in the network (given the number of neurons in the hidden layer is fixed). NF that uses a subset of the inputs is faster to train than a single-network model that accounts for the whole input space. That is the case even though in NF hundreds of networks are fitted. The parameter *mtry* directly influences the training speed when the number of units in the hidden layer is fixed.

Another issue of neural networks is that they are not transparent and they are always regarded as black boxes when used for prediction. The lack of understandability of neural-network models limits their use in solving non-linear regression and classification problems. Similar to RF models, NF can provide end users with a way to identify the most relevant inputs to the response variable. In RF and NF, the influence of each input feature on the response variable can be expressed by an absolute number. NF can also, as any other prediction method studied in this work, apply Equation 2.18 to produce marginal dependence plots.

Finally, estimating the generalization ability of a predictive model is very important to judge its effectiveness in predicting future data patterns. Typically, out-of-sample portion of training

data is used to evaluate the model's performance. In many cases, data available for training is already scarce and further reducing it may render the model unreliable. To use all the available data for training and also for reliable model's validation, cross-validation or bootstrap techniques are widely applied. Despite the effectiveness of these techniques, they are computationally expensive. This problem is even more pronounced when cross-validation and bootstrapping are used in tuning the model's hyper parameters. Like RF, NF offers a very efficient and reliable way to test model's generalization error. That is by using out-of-bag (OOB) data patterns as out-of-sample test examples. The out-of-sample examples are tested on networks whose training data did not include these OOB data patterns. This process is very efficient because testing can be run simultaneously with training. Once the ensemble model is built, its estimated generalization error is readily available. We use the OOB RMSE statistic as estimation for NF generalization error.

The NNET R-language package is used to build the NF networks [87]. This package implements algorithm for training single-hidden-layer, feed-forward, back-propagation networks. NNET allows users to fit both classification and regression networks with different loss and activation functions. It also provides a function to plot the fitted networks.

2.5.2.3 Boosted Regression Trees (BRT)

BRT is an ensemble machine-learning technique based on a stage-wise fitting of regression trees. As the name implies, the technique attempts to minimize the overall loss by boosting the examples having highest predicted errors. That is by fitting regression trees to (accumulated) residuals made by the previous ensemble learners.

There are several different implementations of the boosting concept in the literature. The difference mainly arises from the employed loss functions and treatment of most erroneous

predictions. In this work we employ the stochastic gradient boosting strategy proposed by Friedman [88]. This approach builds a stage-wise model as listed in Figure 2.3 below. The algorithm starts with fitting the first regression tree to all training data. Step 3 (in Figure 2.3) shows that the tree constructed in Steps 1 and 2 is the first term in the BRT additive model. In each subsequent stage, a tree is fitted to residuals obtained by applying the entire model on a random subset of the training data (Steps 4.a, b and c). It should be noted that split variables and values of the internal nodes of every tree (except the first) are decided from the current residuals. Whereas the constant values at the terminal nodes are optimized so that they minimize the loss of the entire model. Trees generation continues as long as their number does not exceed a predefined limit. At the end of each tree-generating iteration, the BRT model is updated according to the formula in Step 4.e. A tree joins the ensemble with a shrunk version of itself. The variable that controls its contribution to the final model is known as *shrinkage* or *learning rate* λ. The smaller the value of the learning rate, the more confident we reach local minima of the error surface. However, the reduction in the learning rate should be compensated by increasing the number of fitted trees by almost the same factor to get the same performance. In our experiments we fixed the shrinkage to 0.005 and evaluated the corresponding optimal number of trees using cross-validation.

The other parameter in the algorithm (listed in Figure 2.3) that requires assignment is the number of terminal nodes created in every tree. This parameter directly affects the size of trees and therefore controls the interaction depth between the explanatory variables. In this text we refer to this variable by the *interaction depth* (which equals the number of terminal nodes - 1). Assigning the value of one to the interaction depth renders an ensemble of stumps. Two-level trees are built when the interaction depth is set to two, and so on. Interaction depth, the number

of trees and the learning rate are related to each other. Usually larger trees (larger interaction depth) requires less number of trees to be fitted given that the learning rate is fixed. Therefore, these parameters should be optimized as a combination to generate accurate models. Section 3.2 discusses the details of the tuning procedure.

Another parameter employed in BRT algorithm is the bag fraction. This parameter controls the subset size of data selected at random each time a tree is built. Data perturbation has two advantages. The first has to do with accuracy; it is known that stochasticity lowers the variance of the overall model. The other is computational benefit, since constructing tree on the whole training data takes longer time than fitting a tree to just a subset of the data. In our analysis we choose the subset size to be 50% of the total training data. From our experience, subset sizes of 40% to 75% work equally well.

1- T_0 = Fit a S-terminal node tree to data $\{X_i, Y_i\}_{i=1}^{N}$

2- $\theta_0 = \arg\min_{\theta} \sum_{i=1}^{N} \psi\left(Y_i, T_0(X_i, \theta)\right)$

3- $F_0(X) = T_0(X, \theta_0)$

4- For l=1 to L do

   a. $\left\{X_i^l, Y_i^l\right\}_{i=1}^{\tilde{N}}$ = random_subset_wo_replacement($\{X_i, Y_i\}_{i=1}^{N}$, 50%)

   b. $\left\{\tilde{Y}_i^l\right\}_{i=1}^{\tilde{N}} = \left\{Y_i^l - F_{l-1}(X_i^l)\right\}_{i=1}^{\tilde{N}}$

   c. T_l = Fit a S-terminal node tree to data $\left\{X_i^l, \tilde{Y}_i^l\right\}_{i=1}^{\tilde{N}}$

   d. $\theta_l = \arg\min_{\theta} \sum_{i=1}^{\tilde{N}} \psi(Y_i^l, F_{l-1}(X_i^l) + T_l(X_i^l, \theta))$

   e. $F_l(X) = F_{l-1}(X) + \lambda . T_l(X, \theta_l)$

5- End for.

Figure 2.3 BRT algorithm.

Since BRT algorithm involves creating large number of trees, the resulting model may seem very complex to interpret. However, similar to RF, BRT offers a method to compute relative influence of the individual input variables on the prediction. Unlike RF, BRT determines such criterion by calculating the number of splits that were made on the input variable X_j instead of perturbing it. The importance estimation I for the variable X_j is calculated for the t^{th} tree as:

$$I_t^j = \sum_{n=1}^{S-1} i_n^2 1(v_n = j), \tag{2.22}$$

where v_n represents the non-terminal nodes of the S-terminal node tree t. The statistic i_n denotes the improvement in performance (reduction of the squared-error) when a split at node n is made on the variable j ($v_n = j$). To generalize the importance metric for all the trees, we take the average over the ensemble as follows:

$$I^j = \frac{1}{T} \sum_{t=1}^{T} I_t^j . \tag{2.23}$$

BRT also generates partial dependence plots the same way RF produces them. We will present and analyze the results of this feature in more detail in Chapter 3.

BRT models were also built in R using the publically available GBM package (version 1.6-3.1) [89]. GBM implements Friedman's gradient boosting machine [90]. It has a built-in cross-validation tool that allows choosing the optimal number of trees automatically. It also supplies default values for all the parameters. Furthermore, GBM implements algorithm for computing

and presenting the relative importance of explanatory variables and produces partial dependence plots. These functions are similar to the ones implemented in RF package.

2.6 Scoring Functions under Comparative Assessment

A total of sixteen popular scoring functions are compared to machine-learning-based scoring functions in this study. The sixteen functions are either used in the mainstream commercial docking tools and/or have been developed in academia. The functions were recently compared against each other in a study conducted by Cheng et al [32]. This bundle includes five scoring functions in the Discovery Studio software version 2.0 [91]: LigScore [92], PLP [93], [94], PMF [24], [25], [27], [29], Jain [95] , and LUDI [20], [96]. Five scoring functions in SYBYL software (version 7.2) [67]: D-Score, PMF-Score, G-Score, ChemScore, and F-Score. GOLD software version 3.2 [97], [98] contributes three scoring functions: GoldScore, ChemScore [99], [100], and ASP [101]. GlideScore [12], [102], in the Schrodinger software version 8.0 [103]. Besides, two standalone scoring functions developed in academia are also assessed, namely, DrugScore [26], [28] and X-Score version 1.2 [57].

Some of the scoring functions have several options or versions, these include LigScore (LigScore1 and LigScore2), PLP (PLP1 and PLP2), and LUDI (LUDI1, LUDI2 and LUDI3) in Discovery Studio; GlideScore (GlideScore-SP and GlideScore-XP) in the Schrodigner software; DrugScore (DrugScore-PDB and DrugScore-CSD); and X-Score (HPScore, HMScore and HSScore). For brevity, we only report the version and/or option that yields the best performance on the PDBbind benchmark that was considered by Cheng et al. [31], [32], [104].

CHAPTER 3

Results and Discussion

After having discussed the compound dataset and the machine-learning methods we will use, in this chapter, we focus on the construction and evaluation of scoring functions. First, in Section

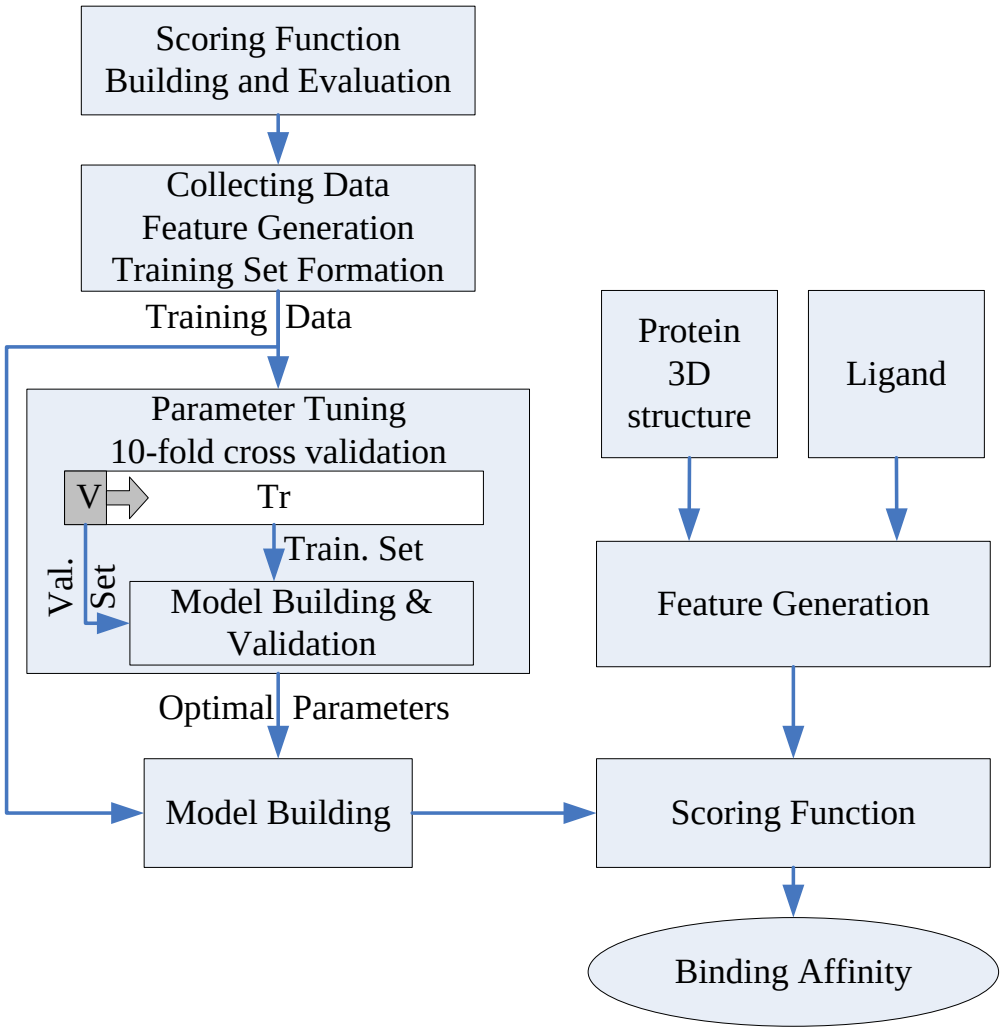


Figure 3.1 Workflow of scoring function construction.

3.1, we discuss the criteria that will be used in later sections to evaluate the effectiveness of scoring functions. Next, in Sections 3.2 and 3.3 we discuss how we tune the modeling parameters of different scoring functions. Figure 3.1 depicts the process of building tuned models, which are then trained on a training dataset and subsequently used for binding affinity prediction using features extracted from independent test protein-ligand complexes. Finally, we describe a number of experiments and present results in Sections 3.4 through 3.10 to assess scoring and ranking accuracies, interpretive capability, and computational complexity of scoring functions.

3.1 Evaluation of Scoring Functions

Perhaps the most important step in the docking process is scoring ligands' conformations in their respective binding sites. This core step affects the outcomes of the drug search campaign. That is because the scoring function predictions determine which binding orientation/conformation is the best, the absolute binding affinity, and which ligand from a database is likely to be the most effective drug. Correspondingly, three main characteristics that a reliable scoring function should have are: (i) ability to identify the correct binding modes of ligands, (ii) accuracy in predicting the binding affinity of a protein-ligand complex, and finally, (iii) capability of ranking active ligands above inactive ones. These three performance attributes were referred to by Cheng et al., as *docking power*, *scoring power*, and *ranking power*, respectively [32]. It should be noted that all three aspects are related to each other and are dependent on the prediction of the absolute binding affinity.

Docking power measures the ability of a scoring function to distinguish a native binding mode from a poor one. Typically, generated conformations are ranked in descending order according to their predicted binding affinities. Ligands' poses that are very close to the experimentally determined ones should also be ranked high. The closeness is measured in Angstroms (Å) as the root mean square deviation (RMSD) from the true binding pose. Generally, in docking, a pose that has RMSD of 2 Å or less is considered successful. This criterion is simple and effective when poses generated are in the range of hundreds to thousands. However, we did not attempt to evaluate the docking power of scoring functions in this work. That is mainly because almost all scoring functions investigated by Cheng et al. showed high docking power

performance [32]. Therefore, we focus on the other two criteria in which the performance of these scoring functions showed high variance and left room for major improvements.

The second aspect of scoring function performance is its ability to accurately estimate the binding energy of a protein-ligand complex. The binding affinity represents the degree of tightness between a ligand and protein. Accurate estimation of binding energy/affinity is a very challenging task and is much more difficult compared to estimating the correct binding mode. In our experiments, we measure the scoring power of every machine-learning-based scoring function using Pearson and Spearman correlation coefficients, and the standard deviation (*SD*) of errors between predicted and measured binding affinities. Pearson correlation coefficient is widely used to describe linear dependence between two random variables and is defined as:

$$R_P = \frac{\sum_{i=1}^{N}(\hat{Y}_i - \bar{\hat{Y}})(Y_i - \bar{Y})}{\sqrt{\left[\sum_{i=1}^{N}(\hat{Y}_i - \bar{\hat{Y}})^2\right]\left[\sum_{i=1}^{N}(Y_i - \bar{Y})^2\right]}},$$

where N is the number of complexes and $\hat{Y}_i$ and Y_i are the predicted and measured binding affinities of the i^{th} complex, respectively. The average values of the predicted and experimentally measured affinities for all complexes are $\bar{\hat{Y}}$ and $\bar{Y}$, respectively. In order to evaluate the correlation between the predicted and measured binding affinities in terms of their ranks, Spearman coefficient should be used instead, and is defined as follows:

$$R_S = 1 - \frac{6\sum_{i=1}^{N} d_i^2}{N(N^2 - 1)}$$

where d_i is the difference between the rankings of the predicted and measured binding affinities for the i^{th} complex. These rankings are assigned based on the order of the complexes according to their known binding affinities.

Both R_P and R_S range between -1 and 1 and they are practically related. The scoring function that achieves the highest correlation coefficient for some dataset is considered more accurate (in terms of scoring power) than its counterparts that realize smaller R_P (and/or R_S) values.

Another accuracy measurement reported here is the standard deviation (*SD*) of errors between predicted and measured binding affinities (in $-\log K_i$ or $-\log K_d$ units). For a given scoring function, a linear model that correlates predicted scores $\hat{Y}$ to the measured ones Y is first evaluated. The *SD* statistic can then be computed as follows:

$$Y = \beta_0 + \beta_1 \hat{Y},$$

$$SD = \sqrt{\frac{\sum_{i=1}^{N} \left[Y_i - (\beta_0 + \beta_1 \hat{Y}_i) \right]^2}{N - 2}},$$

where β_0 and β_1 are, respectively, the intercept and the slope of the linear model. For most of our experiments on machine-learning-based scoring functions, we report their scoring power by at least one of the above statistics (i.e., by at least one of R_P, R_S, and *SD*.)

The third performance criterion for assessing scoring functions is ranking power. Ideally, a scoring function should give high binding affinity scores to ligands that will physically bind

tightly to the given target. By the same token, ligands that do not (or loosely) bind to a receptor should receive low binding affinity scores. By ranking ligands according to their estimated binding scores, one can choose top ligands as the ones that will have higher chances to tightly bind to the target protein in HTS. This is the main idea used when screening a large database of ligands. Ligands with high ranks proceed to the next stages of drug design. In this work, the ranking power of all machine-learning-based scoring functions is calculated across a set of more than 60 different proteins. These protein sets can be regarded as clusters of complexes, where each cluster consists of three complexes with the same protein family.

The ligands corresponding to the complexes within a cluster are chosen such that when they bind with their protein they maximize the range of binding affinity values for that common protein. More specifically, for each protein family, we consider three complexes: these are the complexes with the highest, lowest, and median experimentally-determined binding affinities among all complexes corresponding to a given protein. Calculating the ranking power for a scoring function is then straightforward. First, each protein-ligand complex is scored. Then, for each protein cluster (i.e., the three protein-ligand complexes associated with a common protein), complexes are ordered according to their predicted binding affinity scores. Any given cluster is considered correctly ranked if its predicted-affinity-based order matches its corresponding measured-affinity-based order. The ratio of the number of correctly ranked clusters to the overall number of clusters is considered in this work as an estimation of the ranking power of a given scoring function. Ranking power of scoring functions is only calculated for the core test set, which is composed of 64 protein families. For test sets that are not composed of clusters, one can interpret Spearman correlation coefficient (R_S) as an alternative measure for the ranking power.

3.2 Parameter Tuning and Cross-Validation

Most machine learning approaches we employ for constructing scoring functions in this study need some sort of parameter value selection. Specifically, the kNN method has two parameters that require optimization; these are the neighborhood size *k* and the degree of Minkowski distance *q*. MARS's most important parameters are the number of terms in the model, the degree of each term, and the penalty associated with adding new terms. For the SVM model, we have three parameters: namely, the complexity constant *C*, the width of the ε-insensitive zone ε, and the width (σ) of radial basis function. Random forests (RF) algorithm has effectively only one important parameter, *mtry*, which is the number of features to randomly select at each node split when growing the forest's trees. In addition to *mtry*, neural forests (NF) employs the weights decay parameter that needs to be tuned. Boosted regression trees (BRT), on the other hand, has several parameters. However, the one we will try to tune is the interaction depth. Performance of some of these predictive techniques is quite sensitive to the values assigned to their meta parameters. Generally, the optimum values for such parameters are unknown for any given application and need to be estimated from available data.

One may think that parameter values could be selected in such a way that their respective model best fits the training data, i.e., to minimize the training error. Although it is an intuitive way to select values for meta-parameters, its biggest drawback is the fact that the resultant model may overfit when it is applied to external data. Therefore, we must optimize the model's parameters so that the expected generalization error is minimized instead of the training error. For this purpose, we employ a simple method known as cross-validation which is widely applied in parameter selection. This technique utilizes all the available training data efficiently for

parameter tuning. Models built using the resulting optimal parameters usually generalize well for unseen data.

In *K*-fold cross-validation, the training data (*Pr* in our case) is partitioned into *K* non-overlapping and approximately equal-sized regions. During the k^{th} iteration of *K*-fold cross-validation based parameter optimization, *k = 1* through *K*, a model *f* with an arbitrary parameter set α_j is fit into *K*-1 parts of the data, and validated on the remaining k^{th} part as shown in Figure 3.2. In each iteration, the loss value (typically in RMSE) of the k^{th} model is calculated for the k^{th} validation section and accumulated.

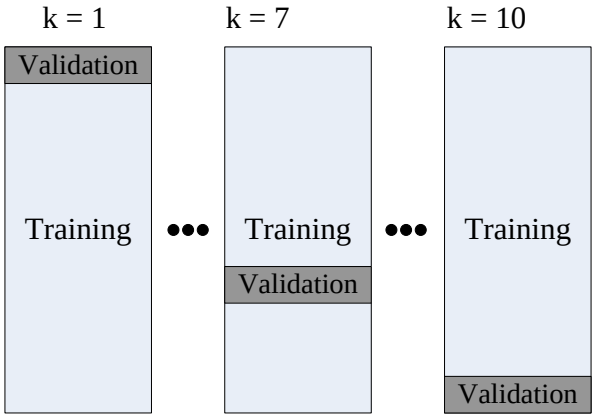


Figure 3.2 10-fold cross-validation.

These accumulated values are averaged over all K folds at the end of the K^{th} iteration (10^{th} in our case). The resultant quantity $avg(CV_k)$ is the cross-validation estimate of the prediction error for the parameter set α_j. The same process is repeated for the next set of parameters α_{j+1} and again the new $avg(CV_k)$ is preserved. We stop when *j* reaches θ (i.e., the last parameter set to

examine). When the given parameter space has been walked, the set of parameters that is associated with the minimum overall cross-validation error is selected as the optimum parameter set α_{opt}. The parameter optimization algorithm is listed in more detail in Figure 3.3.

1. Initialize the arguments of the optimization algorithm
   a. Let Γ be the machine-learning method whose parameters need to be tuned.
   b. Let K=10
   c. Let the training set Tr = Shuffle(Pr), i.e. randomly shuffle the training data.
   d. Split Tr into K "equal" and non-overlapping partitions.
   e. Let the parameter space $\alpha = \{\alpha_j\}_1^{\theta}$.
   f. Let the set CV_α={ } be a storage for cross-validation estimate of the prediction errors associated with every parameter set α_j
2. For j=1 to θ do
   a. Let k=1
   b. Let the set CV_k={} be a buffer for K RMSE's.
   c. For k=1 to K do
      i. Let $\tilde{N}$ be the number of data instances in the k^{th} partition.
      ii. Let the validation set be denoted by $Ts_k = \{X_i, Y_i\}_{i=1}^{\tilde{N}}$
      iii. Let the k^{th} training data be $Tr_k = Tr \setminus Ts_k$, i.e. exclude the k^{th} partition from the original dataset Tr.
      iv. Fit the model f to the training data Tr_k using the fitting method Γ, and the parameter set α_j.
      v. $CV_k = CV_k \bigcup \left[\frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} (Y_i - f(X_i))^2 \right]^{1/2}$
   d. End For loop of step 2.c.
   e. $CV_\alpha = CV_\alpha \bigcup avg(CV_k)$
3. End For loop of step 2.
4. The optimum parameter set $\alpha_{opt} = \arg\min(CV_\alpha)$, i.e. α_j associated with the lowest value in the set CV_α.

Figure 3.3 Parameter tuning algorithm.

An important question that may arise is how to decide on the optimal value for *K*. Unfortunately, there is no precise answer for this question. However, according to some studies, values of 5 to 10 usually yield good results [105], [106]. If it is desired to choose larger or smaller values for *K*, one should be cautious about the model's variance and bias. Typically, larger values of *K* yield high variance models because of the large similarity between datasets on which models were fitted. The largest value K can take is the number of training instances minus one (*N*-1). In such a scenario, the technique is called leave-one-out (or LOO) cross-validation. On the other hand, when *K* is too small, biased models are likely to be generated. That is because such models are trained on smaller datasets, therefore they do not "learn enough."

We tuned MARS and kNN models with the 10-fold cross-validation algorithm presented above. SVM and the ensemble methods RF, NF, and BRT have built-in techniques that can be utilized to tune their parameters. So we used a modified version of the tuning algorithm discussed above. The most significant modification is to replace the internal cross-validation loop in the algorithm by similar tools built internally in SVM, RF, and BRT packages. However, for all machine-learning methods we use grid search to locate the optimal parameters [107]. In grid search, tuples of parameters are tried and the one with the lowest overall loss is picked. The parameter space for any method is constructed by trying linearly increasing sequences of values of the parameters. The only exception to this rule is when we optimize SVM's meta parameters. The parameter sequences in this case are grown exponentially. Next, we further discuss this approach and present the optimal parameters we obtained for each machine-learning method.

3.2.1 Multivariate Adaptive Regression Splines (MARS)

The algorithm of this regression method has several parameters. Many of these parameters are left to take on their default values. The default values are suggested by the authors of the MARS implementation we use in our study [73]. Among these parameters, for instance, is the maximum number of terms allowed in the forward pass, which takes on the value 21 if the number of descriptors of the dataset is less than 10. Otherwise, the default value for this parameter is twice the number of descriptors plus one.

Another parameter that could have been tweaked is the maximum number of terms in the final model (after pruning). This parameter is useful when imposing an upper limit on the model's size is desired. However, we leave this task to the algorithm to decide on the optimal model's size using more important parameters.

One of the only two optimized parameters in MARS is the interaction degree between variables. The second one is GCV (generalized cross-validation) penalty associated with each knot during the pruning pass of the algorithm. We performed a nested sweep for these two parameters to construct the parameter space. The degree parameter was linearly swept from one (i.e., to build an additive model) to three (the final model may have three knots in each term) in step sizes of one. For each degree value, the penalty parameter was also swept from one to ten with a step size of one. By applying this procedure on all optimization datasets (Pr_X, Pr_A, etc.), we noticed that when models are trained and tuned on relatively high-dimensional datasets, the degree of one was always automatically selected. The overall cross-validation error, on the other hand, was not very sensitive to changes in the penalty values. This conclusion is illustrated in Figure 3.4, where model parameters were tuned on Pr_{XAR} dataset. Although the curve looks

quiet flat, the graph shows the penalty of six yields the lowest overall cross-validation RMSE value when the degree of the model was set to one. The optimal values for MARS parameters on all datasets are listed in Table 3.1.

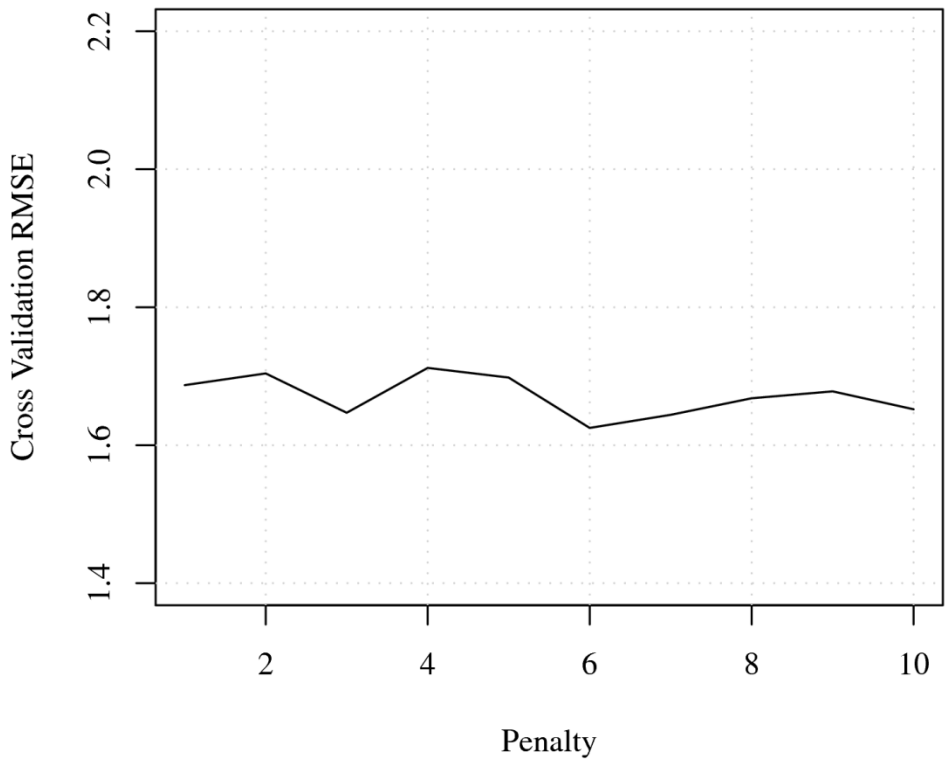


Figure 3.4 Tuning the parameter *GCV-penalty* in MARS.

3.2.2 k-Nearest Neighbors (kNN)

The kNN algorithm we use has only two parameters: namely, the number of neighbors (k) and the order of Minkowski distance (q). We swept k from one to twenty and q from one to five to build the parameter grid we aim to search. It turns out that Manhattan distance ($q = 1$) produces relatively most accurate models for all datasets. In addition, the optimal value of neighborhood size (k) lies somewhere beyond ten. This finding agrees with the general rule of thumb that the

value ten or higher for *k* works usually well. The left panel in Figure 3.5 shows the average cross-validation error when the number of neighbors iterates from one to twenty, and the *q* parameter is fixed at its optimal value, one. The right panel, on the other hand, shows the effect of changing the order of Minkowski distance (*q*) on the models' performance when the neighborhood size is set to eighteen. It can be noticed that low values result in slightly better performing models. Both plots are a result of tuning kNN model parameters on the dataset Pr_{XAR}. Refer to Table 3.1 for the other datasets.

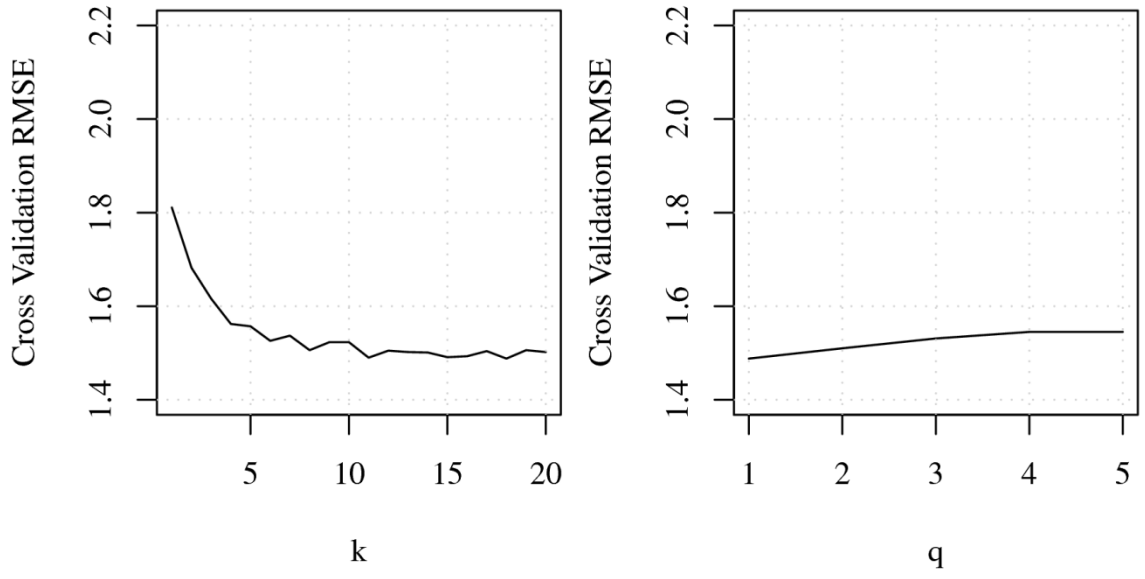


Figure 3.5 Tuning the parameters *k* and *q* in kNN.

3.2.3 Support Vector Machines (SVM)

Perhaps the most sensitive machine-learning technique (compared to other methods considered in this study) to parameter tuning is SVM. The optimization algorithm targets three parameters: the cost parameter *C*, the thickness of the ε-insensitive zone (ε), and the width of radial basis function (σ). An exhaustive grid search for the optimal combination of parameters is conducted

by first expanding the parameter space. This is achieved by generating sequences of exponentially increasing values for the three parameters. More specifically, C is swept from 2^{-10} to 2^{10} in steps of 2, and for each C value, ε iterates from 2^{-15} to 2 with a step size of 2. In addition, for each C-ε pair, the last parameter σ is also varied from 2^{-15} to 2 in steps of 2.

Figure 3.6 on the right shows the resultant cross-validation error when the parameters are optimized for Pr_{XAR} dataset. The values of $C = 1$ (2^0), $\varepsilon = 0.25$ (2^{-2}), and $\sigma = 0.03125$ (2^{-5}), generates a model with lowest expected generalization error. The upper panel displays the cost parameter C plotted against the cross-validation RMSE when ε and σ are fixed at their optimal values 2^{-2} and 2^{-5}, respectively. The cross-validation error quickly declines as C increases, then it slowly rises after it reaches its minimum at $C = 1$. We also obtain the plot illustrated in the middle panel when we record the cross-validation error at different ε values by fixing $C = 1$ and $\sigma = 2^{-5}$. From the resultant

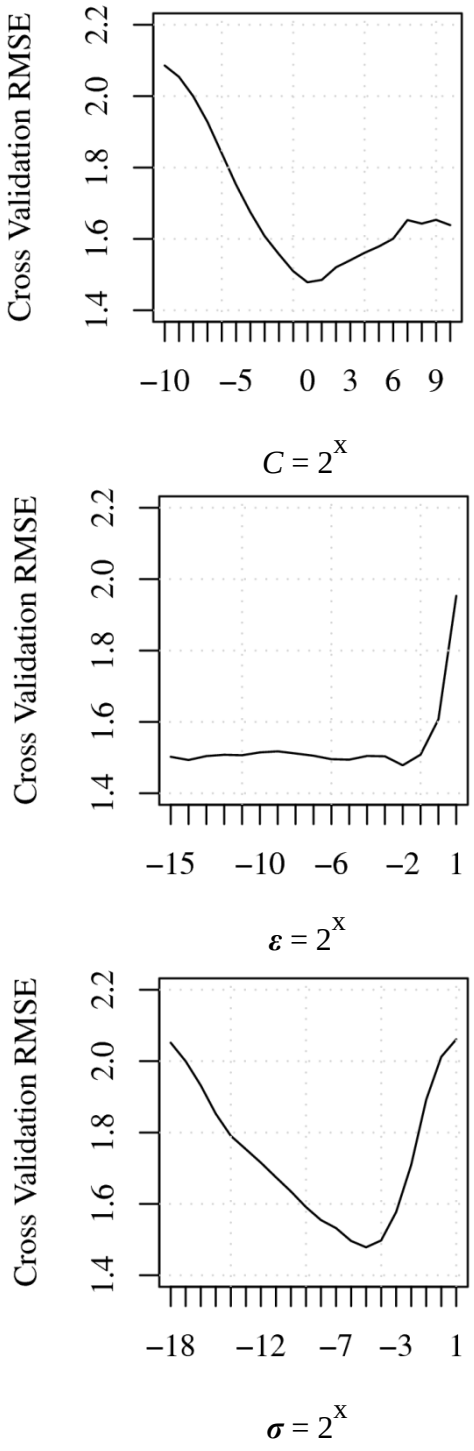


Figure 3.6 Tuning the parameters C, ε and σ in SVM.

curve, we notice that the SVM model is insensitive to changes in ε so long as it is below 0.5. The width of the radial basis function, on the other hand, is quite an influential parameter on the performance of SVM model. The bottom plot supports this claim where it shows a sharp increase in the error as we move away from a σ value of 2^{-5}. We will be using the optimal values, $C = 1$, $\varepsilon = 0.25$, and $\sigma = 0.03125$, in our experiments for XAR-based data or the values listed in Table 3.1 for other datasets.

3.2.4 Random Forests (RF)

In contrast to the other prediction approaches investigated in this work, this ensemble method is the least sensitive to changes in its parameters, which are just two. The number of trees grown in the model does not have any significant impact on the performance so long as it is set to several hundreds. We fixed it to 1000 for all datasets. The other parameter is the number of features randomly sampled at each node split when growing the model's 1000 trees (known as *mtry*). Breiman, the author of Random Forests algorithm, proposes that *mtry* could be set to a value somewhere around the third of the dataset dimension [34]. He also suggests that the model's accuracy around that region is roughly flat. When *mtry* was swept from 2 to 72 for the dataset Pr_{XAR}, we obtained the data depicted in Figure 3.7.

The out-of-bag error does not seem to be affected by changes in *mtry*. Similar conclusions were drawn for the other datasets for the *mtry* values listed in Table 3.1. Such a finding is yet one more powerful characteristic that random forests algorithm offers. In other words, the low number of the technique's parameters and its insensitivity to their fluctuations makes RF an appealing choice as an off-the-shelf prediction tool.

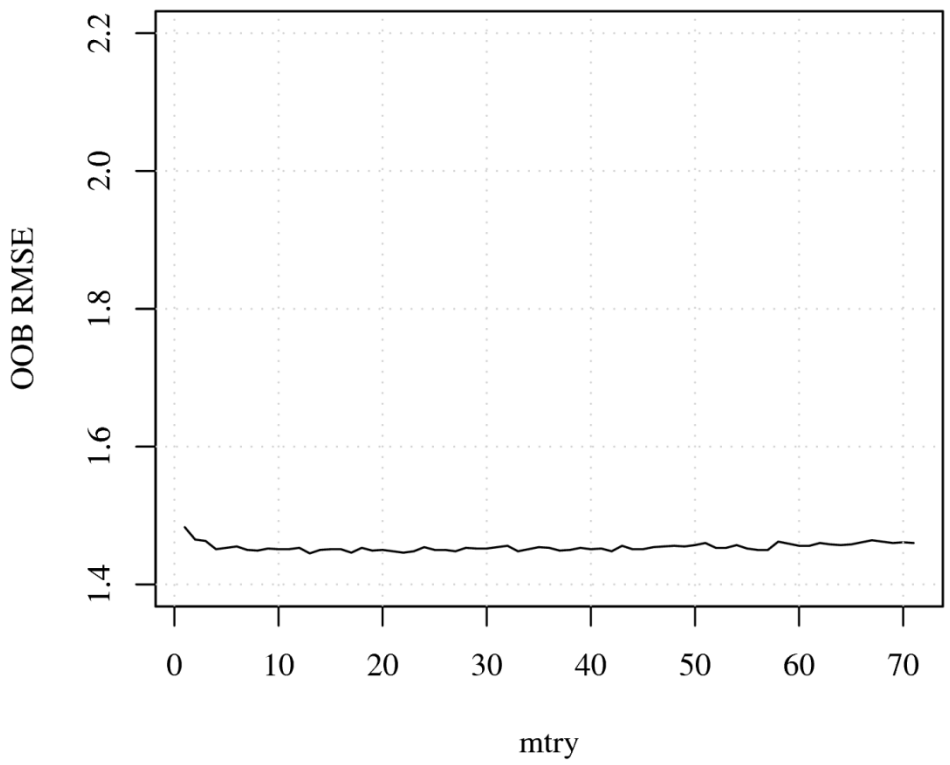


Figure 3.7 Tuning the parameter *mtry* in RF.

Cross-validation was not performed to tune RF parameters as pointed out in Figure 3.3. Instead, we employed the out-of-bag statistic to estimate the generalization error.

3.2.5 Neural Forests (NF)

The NF model was allowed to learn 1000 MLP networks. The number of hidden-layer neurons in these networks is set to 20, where the number of inputs is controlled by the variable *mtry*. Weights in each network in the ensemble were optimized using back-propagation and regularization techniques. As discussed in Section 2.6.2.2, the weight decay parameter λ needs to be tuned in order for the weights to converge to their optimum values.

We conducted a grid search for the optimal pair values of *mtry* and λ that minimize the OOB root means-squared error (RMSE). The parameter *mtry* was swept linearly between two

boundary values *a* and *b*, and with a step size of *c*. For most datasets (i.e., Pr_{XA}, Pr_R, etc.), *a*, *b* and *c* were, respectively, set to 10, 35, and 5. For datasets with smaller number of features, such as X and A, their *a*, *b*, and *c* values were set to 2, 6, and 1 for Pr_X and 10, 25, and 5 for Pr_A datasets. The weight decay parameter λ was exponentially varied between 2^{-15} and 2 with a step of size of 2.

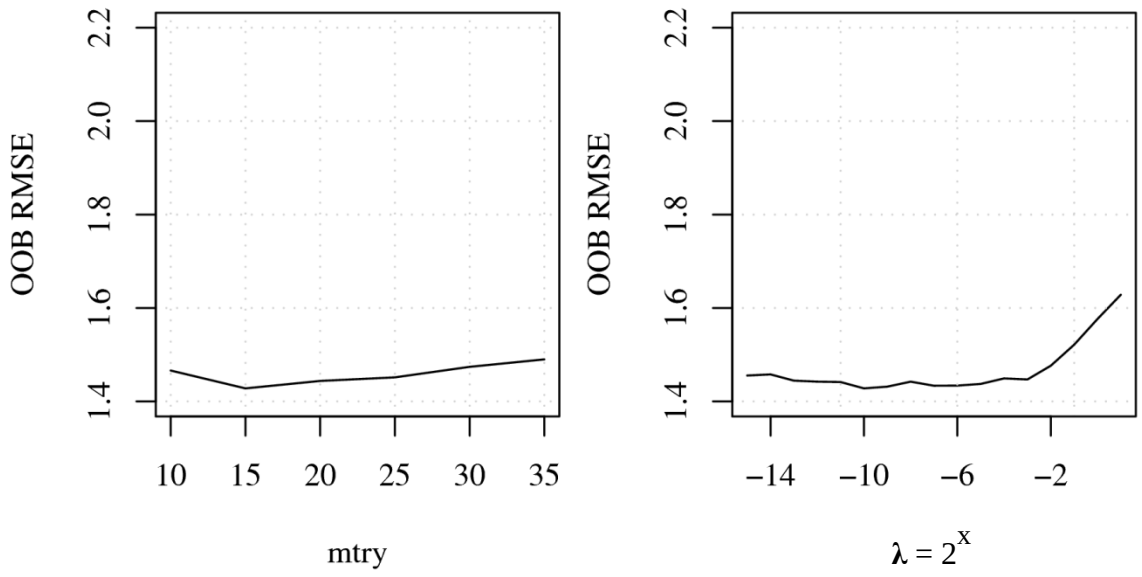


Figure 3.8 Tuning the parameters *mtry* and the *weight decay* (λ) in NF.

The optimal values of *mtry* and *λ* for complexes described by XAR features are ≈ 0.001and 15, respectively. As shown in Figure 3.8 above, NF performance is slightly more sensitive to the value of *mtry* than RF model. This curve was obtained when λ was fixed at it is optimal value (0.001). From the right panel of Figure 3.8 we also notice that OOB RMSE is fairly insensitive to the exact value of λ as long as it is less than ≈ 0.06 (2^{-4}). Again, data depicted in the right panel was collected when *mtry* was fixed at 15. The optimal values for *mtry* and *λ* are almost the same for the other feature sets as listed in Table 3.1.

3.2.6 Boosted Regression Trees (BRT)

The BRT algorithm has several parameters that determine the model's goodness-of-fit and generalization performance. Among the most influential parameters are the number of trees to fit, the shrinkage factor that determines the contribution of each tree to the whole ensemble, the interaction depth of each tree, and less importantly the bag fraction. The last parameter, bag fraction, controls the number of instances randomly drawn from the original training data in order to grow the next tree.

The algorithm's shrinkage parameter, or the learning rate, was fixed at 0.005. The interaction depth of the grown trees was swept from one to twenty with a step size of one. For each interaction depth we obtained the optimal number of trees. The optimal number of trees is calculated internally in BRT algorithm. BRT employs a k-fold cross-validation procedure to obtain the number of trees (up to a maximum of 8000) that minimizes the overall error. Therefore, this nested procedure allows us to obtain the optimal interaction depth and the optimal number of trees for this depth (which are 16 and 2213 for Pr_{XAR}, respectively).

The results of the tuning run on Pr_{XAR} dataset are summarized in Figure 3.9. It is clear that the cross-validation error holds steady even when relatively large trees are grown. However, the model's optimal number of trees for each interaction depth drops as trees get larger. This can be seen in the right panel of Figure 3.9. One can observe that the model's size (in terms of the number of trees) at interaction depth of 5 (~ 6000) is three times that when the interaction depth climbs to 20 (~ 2000 trees).

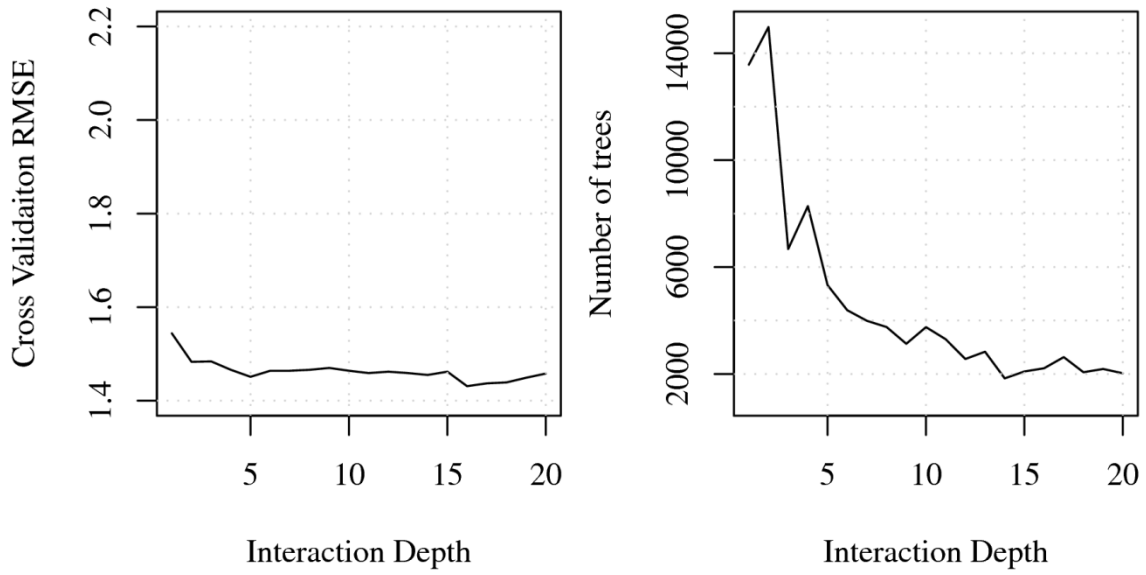


Figure 3.9 Tuning the parameter *Interaction Depth* in BRT.

Table 3.1 Optimal parameters for MARS, kNN, SVM, RF, NF and BRT models.

Methods	**Parameters**	**When complexes are described by the features :**						
		X (X-Score)	A (AffiScore)	R (RF-Score)	XA	XR	AR	XAR
MARS	*Degree*	2	1	1	1	1	1	1
	Penalty	2	6	5	7	2	7	6
kNN	*k*	15	13	14	9	19	19	18
	q	1	1	1	1	1	1	1
SVM	*C*	2	2	1	2	4	2	1
	ε	0.5	0	0.25	0.125	0.125	0.25	0.25
	σ	1	0.25	0.125	0.125	0.031	0.031	0.031
RF	*mtry*	3	18	8	31	5	10	14
NF	*mtry*	5	15	15	15	15	15	15
	λ	0.001	0.004	0.001	0.002	0.001	0.001	0.001
BRT	*Int. depth*	15	17	18	19	15	19	16
	N. Trees	1114	1523	1573	1371	2113	2950	2213

3.3 Tuning and Evaluating Models on Overlapped Datasets

In some experiments we carry out, we use *Pr* and *Sc* datasets for building scoring functions and *Cr* for testing them. In other cases, though, *Pr and Sc* will be used for both training and testing. As was pointed out in Section 3.2, the entire *Pr* dataset was already used to tune the free parameters for machine learning models we intend to use as scoring functions. Consequently, there is some overlap between the data used (in one way or another) to build the model and the data on which it will be evaluated. That is to say, to fairly judge the performance of a machine-learning-based scoring function, parameter tuning should be performed on a dataset that is independent of the test set, as it is the case for training. However, in some situations, it is computationally expensive to tune a model's parameters whenever we train and test it, particularly, if the experiment involves building ensemble of models (in order to obtain unbiased and robust results, for instance).

For the prediction techniques under study, we ask: how much gain in performance can be achieved when there is an overlap between tuning and test datasets? If the gain is insignificant, we may relax the constraint that parameter selection and model testing must be conducted on disjoint datasets. In order to estimate the performance gain for kNN, SVM, and MARS, we applied the algorithm shown in Figure 3.10.

The algorithm starts with splitting training data *Pr* into 10 non-overlapping training (90%) and test (10%) datasets. It proceeds to create two models; one is tuned on the whole *Pr* dataset (overlapping model), while the other is tuned on just the k^{th} training part of the data (disjoint). The optimal parameters are subsequently used to fit both the models to the k^{th} training partition.

The scoring functions are then evaluated on the k^{th} test set. When the last fold is reached, the cross-validation errors of the overlapping and disjoint models are averaged and compared. Table 3.2 shows the results (gain in performance) for kNN, MARS, and SVM scoring functions across

1. Initialize the arguments of the algorithm
   a. Let Γ be the machine-learning method whose parameters need to be tuned.
   b. Let K=10
   c. Let the training set Tr =Shuffle(Pr), i.e. randomly shuffle the training data.
   d. Split Tr into K "equal" and non-overlapping partitions.
   e. $\alpha_{ovr} = tune(Tr, \Gamma)$. Tune the parameters for the "*overlapping*" model f_{ovr}.
   f. Let the set CV_{ovr}={ } and CV_{dis}={ } be RMSE buffers for overlapping f_{ovr} and disjoint f_{dis} models; respectively.
2. For k=1 to K do
   a. Let $\tilde{N}$ be the number of data instances in the k^{th} partition.
   b. Let the validation set be denoted by $Ts_k = \{X_i, Y_i\}_{i=1}^{\tilde{N}}$
   c. Let the k^{th} training data be $Tr_k = Tr \setminus Ts_k$, i.e. exclude the k^{th} partition from the original dataset Tr.
   d. $\alpha_{dis} = tune(Tr_k, \Gamma)$. Tune the parameters for the "*disjoint*" model f_{dis}.
   e. Fit f_{ovr} and f_{dis} models to the training data Tr_k using the fitting method Γ, and the parameter sets α_{ovr} and α_{dis}; respectively.
   f. $$CV_{ovr} = CV_{ovr} \cup \left[\frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} (Y_i - f_{ovr}(X_i))^2 \right]^{1/2}$$
   g. $$CV_{dis} = CV_{dis} \cup \left[\frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} (Y_i - f_{dis}(X_i))^2 \right]^{1/2}$$
3. End For loop.
4. $RMSE_{ovr} = avg(CV_{ovr})$
5. $RMSE_{dis} = avg(CV_{dis})$
6. $$\text{Gain in Performance}(\%) = \frac{RMSE_{dis} - RMSE_{ovr}}{RMSE_{dis}} \times 100$$

Figure 3.10 Algorithm for tuning and evaluating models on overlapped datasets.

the seven *Pr* datasets. Based on the numbers listed in the table, it would not be a subjective claim if one concluded that tuning parameters of MARS, kNN, or SVM models on overlapping datasets does not significantly help their performance. Most of the enhancements in performance are less than 1%. In addition, there are some models that perform slightly better when tuned and tested on independent datasets. Therefore, it should not be an issue when models are tuned on datasets that share instances with datasets on which these models are tested.

Table 3.2 Results of tuning and evaluating models on overlapped datasets.

Methods	**Criteria**	**When complexes are described by the features:**						
		X (X-Score)	A (AffiScore)	R (RF-Score)	XA	XR	AR	XAR
MARS	Disjoint	1.708	1.653	1.706	1.625	1.604	1.695	1.689
	Overlap	1.679	1.642	1.679	1.628	1.600	1.704	1.680
	Gain(%)	1.7	0.7	1.6	-0.2	0.2	-0.5	0.5
kNN	Disjoint	1.576	1.580	1.603	1.523	1.548	1.504	1.495
	Overlap	1.563	1.572	1.603	1.515	1.547	1.501	1.495
	Gain(%)	0.8	0.5	0	0.5	0.1	0.2	0
SVM	Disjoint	1.599	1.613	1.625	1.581	1.546	1.571	1.561
	Overlap	1.596	1.629	1.622	1.584	1.556	1.575	1.569
	Gain(%)	0.2	-1.0	0.2	-0.2	-0.6	-0.3	-0.5

3.4 Machine-Learning vs. Conventional Scoring Functions

We have created several scoring functions using different regression algorithms in conjunction with different feature sets extracted from the same compound set *Pr*. More specifically, we considered the primary training set that is described in seven different forms: namely, Pr_X, Pr_A, Pr_R, Pr_{XA}, Pr_{XR}, Pr_{AR}, and Pr_{XAR}. Then seven machine learning models (MLR, MARS, kNN, SVM, BRT, RF, and NF) were fitted to these datasets to generate 49 (7 x 7) different scoring functions. We distinguish each one of these using the notation *Machine-learning technique::tools used to calculate features*. For instance, kNN::XA indicates that the scoring function is a k-Nearest neighbor model that is trained and tested on datasets (the primary training set and the core set, respectively) described by XA features (i.e., features extracted by using the tools X-Score and AffiScore).

We compared the performance of these 49 scoring functions against each other and against the 16 presented in Section 2.6. Similar notation is also used to write these 16 scoring functions that were originally studied by Cheng et al. [32]. For example, when referring to the scoring function LigScore1 in Discovery Studio software, the notation DS::LigScore1 is used. In fact, if we considered all scoring functions that could result from the modeling tools mentioned earlier in Section 2.6, we would end up having large numbers of them. That is because every modeling tool typically supplies more than one version/option for the same scoring function. Each such variation could be regarded as a different scoring function. Therefore, only the version/option that yields the best performing scoring function was considered. LigScore, for example, comes in two different options (LigScore1 and LigScore2).

We follow the same approach for selecting a smaller set from the 49 scoring functions mentioned above. For each machine-learning technique, only the best performing model was selected out of the possible seven. Table 3.3 lists the performance of machine-learning-based scoring functions and the other 16 on the core test set *Cr*. The scoring functions are ordered based on their scoring power in terms of Pearson correlation coefficient. It is clear that most machine-learning-based scoring functions significantly outperform the 16 scoring functions. This is not only the case for Pearson correlation coefficient, but also in terms of Spearman rank correlation coefficient and prediction standard error *SD* of each function. BRT, NF, and RF models top the list of the 23 scoring functions whose correlation coefficients are close to 0.8. When the best performing machine-learning model (BRT) is compared (in terms of its Pearson correlation coefficient) against the best performing conventional scoring function (X-Score::HMScore) out of the 16, we realize about 24% improvement in scoring power. It should be noted that such a boost in performance comes purely from the powerful fitting algorithm and the diversity of features describing protein-ligand complexes (since features used in both empirical and knowledge-based scoring functions are employed by the best machine-learning based approaches).

Moreover, the primary training dataset on which the machine learning models were fitted does not overlap with the core set. Whereas, training datasets of some of the 16 existing scoring functions share some examples with the test set. Such overlap may, although not necessarily, benefit scoring functions that were trained and evaluated on non-independent datasets. However, overlapping should be avoided when possible. It is impossible to guarantee such a requirement in all the 16 scoring functions considered by Cheng et al. This is because some of them were built in different studies and details about their training data are poorly reported. Cheng and co-

workers calibrated two versions of their X-Score scoring function. One, X-Score v1.2 that is reported in Table 3.3, was trained and tested on partially overlapping datasets. The other, X-Score v1.3, was fitted to the primary training data and evaluated on the core set. Performance of the two functions was almost the same. The overlapping scoring function resulted in R_P= 0.644 while X-Score v1.3 yielded R_P = 0.649.

Table 3.3 Comparison of the scoring power of 23 scoring functions on the core test set *Cr*.

Scoring Function	N	R_P	R_S	*SD*
BRT::XAR	192	0.797	0.781	1.43
NF::XAR	192	0.796	0.784	1.43
RF::XR	192	0.794	0.788	1.44
SVM::XAR	192	0.768	0.792	1.51
KNN::AR	192	0.735	0.736	1.6
MARS::XR	192	0.704	0.732	1.68
MLR::XAR	192	0.672	0.756	1.75
X-Score1.2::HMScore	195	0.644	0.705	1.83
DrugScoreCSD	195	0.569	0.627	1.96
SYBYL::ChemScore	195	0.555	0.585	1.98
DS::PLP1	195	0.545	0.588	2.00
GOLD::ASP	193	0.534	0.577	2.02
SYBYL::G-Score	195	0.492	0.536	2.08
DS::LUDI3	195	0.487	0.478	2.09
DS::LigScore2	193	0.464	0.507	2.12
GlideScore-XP	178	0.457	0.435	2.14
DS::PMF	193	0.445	0.448	2.14
GOLD::ChemScore	178	0.441	0.452	2.15
SYBYL::D-Score	195	0.392	0.447	2.19
DS::Jain	189	0.316	0.346	2.24
GOLD::GoldScore	169	0.295	0.322	2.29
SYBYL::PMF-Score	190	0.268	0.273	2.29
SYBYL::F-Score	185	0.216	0.243	2.35

One may expect that ranking power of scoring functions should be reflective of their scoring capabilities. After all, ranking power is merely ordering ligands that bind to a common protein

according to their affinity scores. Moreover, to identify the correct ranking of ligands, their predicted binding scores do not have to linearly correlate with the measured affinity constants. Nevertheless, Cheng et al. and our results indicate that scoring functions perform less satisfactorily in ranking as opposed to scoring. Table 3.4 shows that the top scoring function has succeeded in correctly ranking the three ligands in less than two-thirds of targets (40 out of 64).

Table 3.4 Comparison of the ranking power of scoring functions on the core test set *Cr*.

Scoring function	**N**	***(%)***
BRT::XA	192	62.5
RF::XAR	192	60.9
NF::XAR	192	59.4
X-Score1.2::HSScore	195	58.5
SVM::XAR	192	57.8
DS::PLP2	195	53.8
KNN::AR	192	53.1
DrugScoreCSD	195	52.3
MLR::XAR	192	51.6
MARS::XR	192	50
SYBYL::ChemScore	195	47.7
SYBYL::G-Score	195	46.2
SYBYL::D-Score	195	46.2
GOLD::ASP	193	43.1
DS::LUDI3	195	43.1
DS::LigScore2	193	43.1
DS::PMF	193	41.5
DS::Jain	189	41.5
SYBYL::PMF-Score	190	38.5
GOLD::ChemScore	178	36.9
GlideScore-XP	178	33.8
SYBYL::F-Score	185	29.2
GOLD::GoldScore	169	23.1

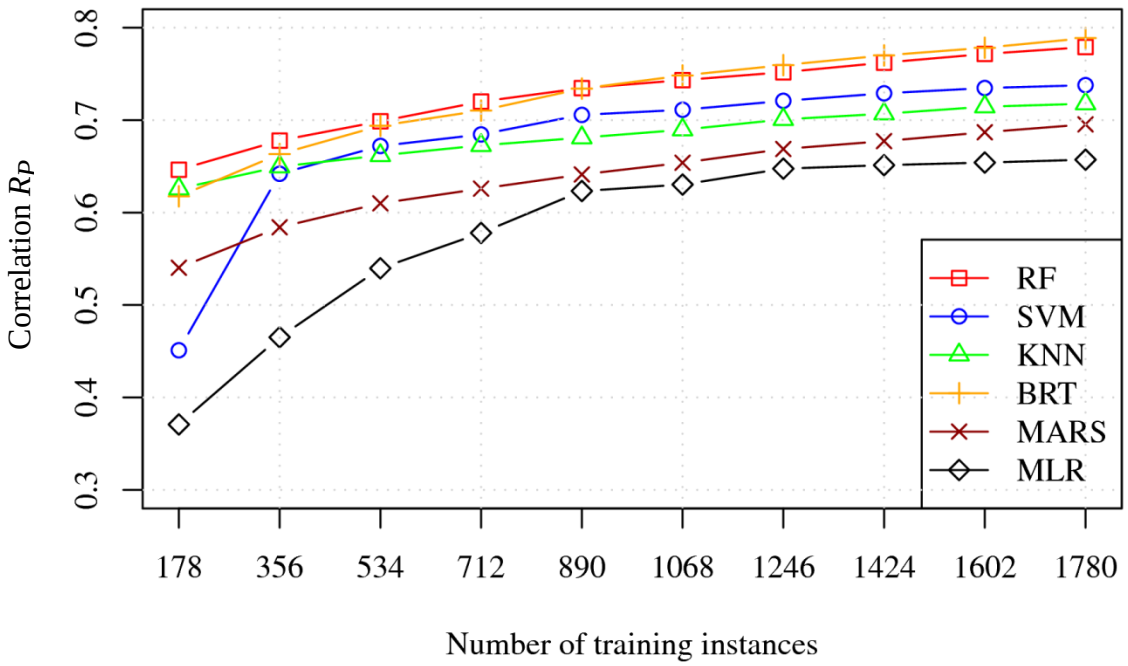
The ranking power performance difference between the top performing machine-learning-based scoring function (BRT::XA) and X-Score::HSScore is less than 15% as compared to 24%

improvement in scoring power. In terms of ranking power, it should be noted that the top three scoring functions were also obtained by applying ensemble methods (BRT, RF, and NF). The surprising result, however, is that the other machine-learning predictive models that significantly outperform X-Score scoring function in scoring power are lagging behind in terms of ranking power.

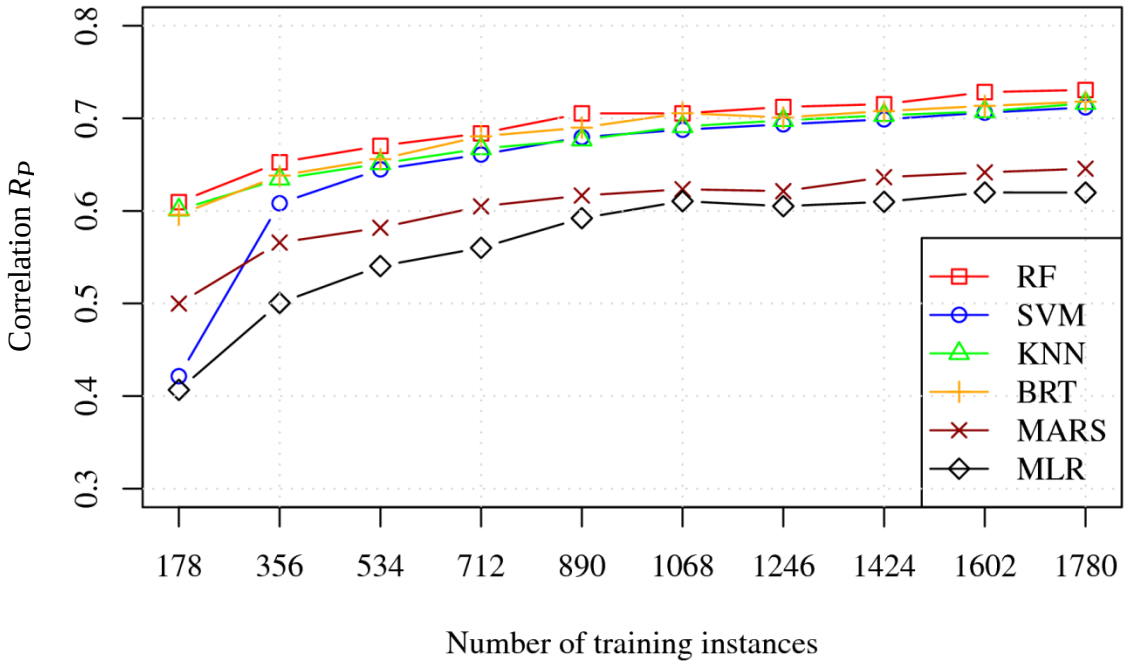
3.5 Dependence of Scoring Functions on the Size of Training Set

Experimental information about structures and binding affinities of new protein-ligand complexes are regularly determined and made publically available. Protein Data Bank (PDB), for instance, is a large macromolecular archive that stores huge and diverse information about proteins, nucleic acids, and complexes. The size of this database, and other similar public biochemical repositories, in addition to corporate compound banks, are regularly growing. In this work, we use an annually updated database called PDBbind [31], [104]. As it is explained in Section 2.1, PDBbind provides binding affinities of high-quality 3D structures of protein-ligand complexes that can be readily utilized in building and testing new scoring functions. More than 750 new protein-ligand complexes have been deposited into this database from the year 2007 to 2010. Here, we take advantage of such abundance of data to assess the impact of growing the size of training set on the effectiveness of scoring functions.

We use the secondary training data set *Sc* to build and test different machine-learning-based scoring models. Every model is trained on an increasing subset from *Sc* and validated on two independent test sets. Models are first fit to small-sized (starting with 178 instances) datasets and over ten steps this number gradually grows to 1780 instances (178 instances increase in each step). At each training size, 206 instances are randomly drawn (without any overlap with the just sampled training data) to constitute one of the test sets. The second test sample is the core test set *Cr*. To obtain reliable results, this process is repeated 50 times for each size of training data. During each iteration, bootstrap mechanism (i.e., selection without replacement) is used to select training data. After averaging the 50 values of the correlation coefficients of the 50 generated scoring functions, we obtained the data shown in Figures 3.11 and 3.12.

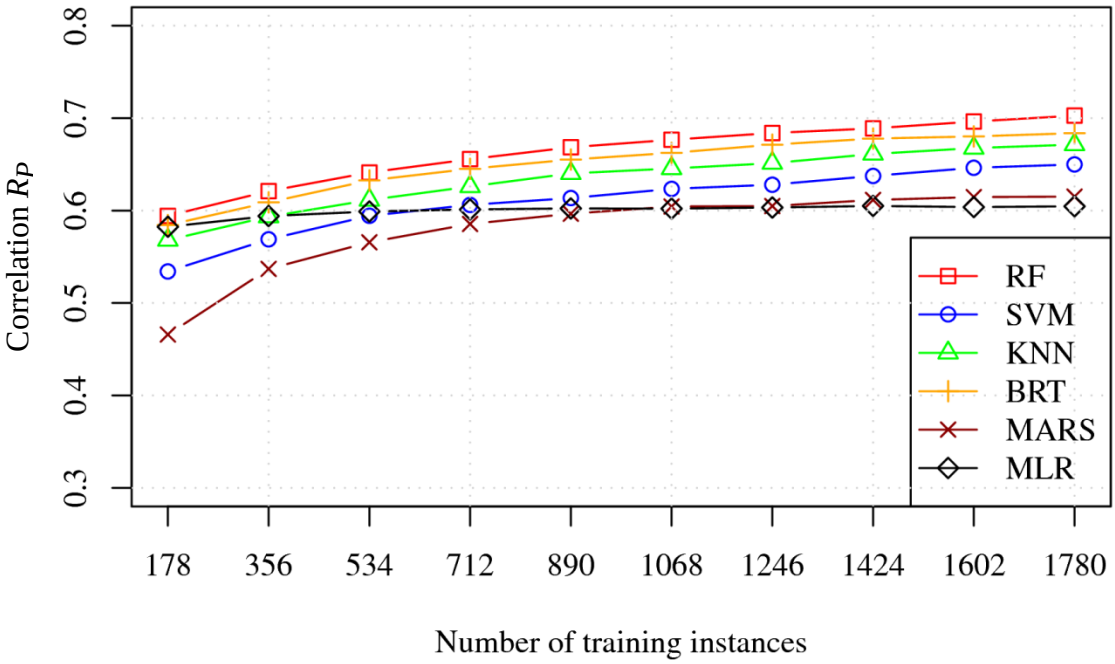


a. Core test set.

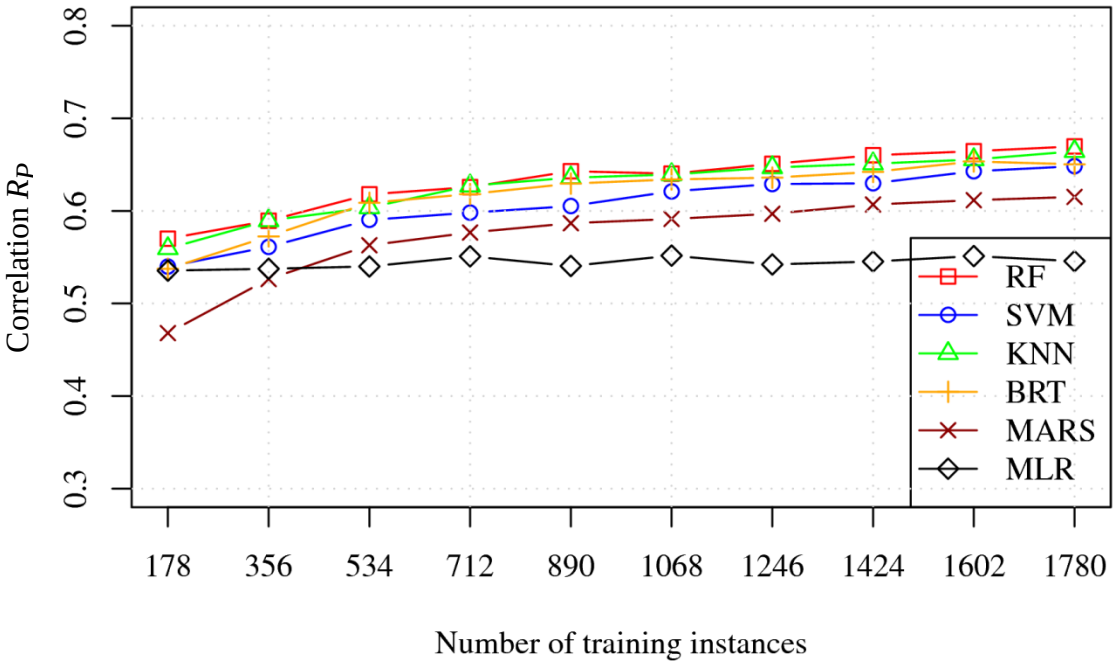


b. Random test set.

Figure 3.11 Dependence of XAR-based scoring functions on the size of training set.



a - Core test set.



b - Random test set.

Figure 3.12 Dependence of X-based scoring functions on the size of training set.

Notice that NF is not included in this experiment due to its long training time. The two panels in Figure 3.11 show the results when the datasets are characterized using XAR features, while the plots in Figure 3.12 are for the case when complexes are described by X features. In the four cases (two test sets and two characterization methods), we observe that performance of all models improves as the size of the training data increases.

When the number of features is high (XAR = 72) and the size of training data is relatively small, we notice poor accuracy of some scoring functions such as MLR and SVM. This is perhaps a consequence of overfitting since the ratio of training data size to number of features is small. The performance of MLR and SVM is boosted when the number of data patterns rises and/or lower dimensional data is used (as shown in Figure 3.12). RF, BRT, and kNN based scoring functions are almost always the leading models regardless of the data dimensionality and the size of training set.

It is evident that the scoring power of machine-learning models is greater when the test is conducted on the core set. In other words, it may be thought that testing scoring functions on randomly drawn instances should yield results similar to those obtained on the diverse test set *Cr*. Comparing curves on the top panels to the ones on the bottom show that all scoring functions do better job in predicting binding affinity of the core set complexes. This gap in performance can be attributed to the large variance of binding energies of protein-ligand complexes constituting the core test set. Remember that upon building the core set the emphasis was to diversify it in terms of protein families and to maximize the range of their binding constants. It is therefore easier for scoring functions to estimate binding constants of very different protein-ligand structures whose binding constants are far apart. Such variation is not guaranteed in case of randomly drawn protein-ligand complexes and hence their binding constants are rather less

distinctive. Additionally, the way the core set is constructed, there is at least one complex in *Pr* for each cluster represented in *Cr*. This implies that *Pr* has training complexes relevant to the complexes in *Cr*.

By looking to curves in Figure 3.11, one may claim that scoring functions will keep improving because the ratio of data size to the number of dimensions (features) is increasing and thus models are less susceptible to overfit. Although this argument seems valid, it is still worth investigating the behavior of scoring functions when the data size to dimension ratio becomes relatively very large. For that purpose, we carried out a related experiment on two datasets characterized by different numbers of features (i.e., Sc_X and Sc_{XAR}). We assume that differences in performance between predictive models extracted from Sc_X and those from Sc_{XAR} are mainly resulting from the difference in number of features instead of feature types. Such an assumption is sensible to some extent since both datasets share the descriptors extracted via X-Score tool. Moreover, the focus will be on performance trends rather than on the absolute values of the correlation statistic.

As stated earlier, X-based models (i.e., RF::X, BRT::X, etc.) tend to improve with increase in training dataset sizes. SVM and MLR curves in Figure 3.12 show that such models no longer suffer from overfitting. Furthermore, the scoring power of MLR model does not show significant change as the size of training data increases. On the other hand, the improvement in performance for the rest of the models is similar to that seen for the corresponding XAR-scoring functions.

According to findings summarized in Figures 3.11 and 3.12, we can conclude that most machine-learning-based scoring functions have a potential for improvement as more training data becomes available. That is the case weather the ratio of training dataset size to number of

dimensions is low (178/72 = 2.47 to 1780/72 = 24.72) or relatively high (178/6 = 29.67 to 1780/6 = 296.67). Scoring function designers can conduct similar experiments to estimate accuracy enhancement when their proposed functions are recalibrated on larger number of data instances. Take, for example, the BRT model. We can approximately project its scoring power on the core test set after a few years from now. If we averaged its improvement slope over the last three increments in training size, we obtain roughly 0.01 increase in correlation for each 178 increase in number of training patterns. By assuming that another 750 protein-ligand complexes will be deposited into PDBbind in the next 3 years, one can then expect BRT::XAR scoring power to go up to about R_P = 0.822 (from the current R_P = 0.779, i.e., a 5% increase) in 2013. Perhaps it does not seem a great leap in improvement, but this enhancement would certainly have a great impact when the goal is to only choose promising drugs from databases that contain millions of drug-like molecules.

MLR::X model, which resembles empirical scoring functions, has very small potential for improvement in the future. This is due to the rigidity of linear models whose performance tends to "saturate". Many of the 16 scoring functions compared to the machine-learning-based models are empirical and thus they are also most likely to suffer from the same limitation. In fact, the best performing model of those 16 in terms of scoring power, X-Score::HPScore, is very similar to MLR::X. That is because both scoring functions use almost the same features and both are linearly fit to the same training data. Therefore, one should consider better prediction approaches to derive accurate models from training data available on hand and from future updates.

3.6 Dependence of Scoring Functions on the Number of Features

The binding affinity of a protein-ligand complex depends on many physicochemical interaction factors involving a protein and its ligand. Capturing all relevant factors is not possible and calculating some of them is not a trivial task. Furthermore, quantifying these factors into values known as features or descriptors involves many assumptions and approximations. To minimize the effect of employing such imperfect features, we utilize more than one tool to calculate them. As it was detailed in Section 2.2, we considered in our analysis three tools to extract features for the scoring functions: X-Score, AffiScore, and RF-Score. Each tool generates different features and even the ones shared by more than one tool are computed differently. Utilizing all features provided by these tools (even the redundant ones) provides a fuller picture of the various protein-ligand interaction effects.

Table 3.5 below shows correlation coefficients of machine-learning models that were trained on the primary dataset *Pr*. All resultant scoring functions were tested on the core test set *Cr*. The first three columns in the table show the performance when protein-ligand complexes are described using X, A, and R features. The last column lists the performance when the complexes are characterized by all these features combined, i.e., XAR. Evidently, better results are obtained when regression models are fit to datasets described with more and diverse features. Similar results were seen when the impact of increasing the size of training set was investigated in the previous section (see Figures 3.11 and 3.12). In that case, scoring functions were induced from the secondary dataset *Sc*. The difference in performance for all functions is very obvious on both test data patterns (i.e., random and core sets).

For some machine-learning methods, their generalization error may increase dramatically by adding more dimensions to the data. That is because there would be a higher chance that irrelevant information is taken into account. Noise associated with features having weak relevance with the outcome variable could degrade models as well. Overfitting is another serious problem more likely to arise when the size of training data is inadequate.

Table 3.5 Performance of several machine-learning-based scoring functions on a compound test set described by different features.

	X (X-Score)	A (AffiScore)	R (RF-Score)	XAR
BRT	0.707	0.772	0.741	0.797
NF	0.713	0.771	0.751	0.796
RF	0.732	0.755	0.763	0.794
SVM	0.672	0.703	0.733	0.768
kNN	0.685	0.701	0.707	0.732
MLR	0.631	0.663	0.59	0.672
MARS	0.63	0.669	0.63	0.628

We propose a simple experiment to investigate how our machine-learning models (except NF) behave when the data dimension is increased. The experiment is conducted on the secondary dataset characterized by XAR features. R features in this experiment are calculated slightly different than in previous sections. Recall that the RF-Score tool extracts 36 features each of which is a count of the occurrence of specific protein ligand atom pairs within a predefined distance range. In previous XAR sets, the distance range was set to 12Å. Whereas, in this experiment, we use several distance ranges or bins. Atom pair statistics are counted over five contiguous four-angstrom bins, that cover an overall radius of (5 x 4) 20Å. Therefore, there are 36 features emerging from each bin. By combining the new R features (36 x 5 = 180) with XA's (36), we gather a feature pool of size 180 + 36 = 216 total number of features. Bin size and number are arbitrarily chosen. Actually, the quality of features in this experiment is not as

important as their number so long as they are relevant to the binding affinity. Note that the aim of this experiment is to investigate the impact of increasing data dimensionality on the performance of different scoring models.

After generating the feature pool we proceed to evaluating machine-learning-based scoring functions on datasets with an increasing number of features. We start with $k = 20$ features randomly chosen from the feature pool. Next, the data (complexes in *Sc*) is characterized by the selected features and then training and test data sets are randomly sampled without replacement and without overlap from *Sc*. At every random feature set and data sample, the performance of scoring functions in terms of scoring power is evaluated. Ensembles of models that are trained on fifty bootstrap data samples of the same features are then averaged. Likewise, another random set of features with the same size is drawn from the feature pool. This outer loop is also repeated fifty times. Then we boost *k* by 20 and rerun the same steps. The procedure terminates when k reaches 200.

The above procedure is listed in more detail in Figure 3.13. When this algorithm was applied for RF, BRT, SVM, kNN, MARS, and MLR, we obtained results depicted in Figure 3.14. All performance statistics of scoring functions rise relatively sharply when data dimensionality increases from 20 to 40 features. Scoring power of most predictive models keeps improving by increasing the number of features beyond 40, although at a lower rate than before. Performance of SVM and MLR models, on the other hand, appears to degrade as data dimension rises. We suspect that had SVM parameters been tuned for every random set of features, SVM model could have delivered better results. Recall from Section 3.3 that SVM performance is very sensitive to the appropriate selection of its meta parameters. However, tuning these parameters for each value of *k* is computationally expensive and therefore we did not attempt to search for the

optimal parameter set for every feature set size. As for MLR-based scoring function, the plot indicates that the larger the number of features, the more overfitting it experiences. Consequently, dimension reduction is necessary before building linear models when there is a plethora of features describing protein-ligand complexes.

1. Select and initialize the algorithm arguments
   A. Let Γ be the machine-learning method to be tested
   B. Let the set C_k={ } be a buffer for the correlation coefficient statistic for each value of *k*.
2. For *k* = 20 to 200, with increments of 20 do
   A. For *i* = 1 to 50 do // Counter for feature samples drawn from the feature pool.
      i. $C_i = 0$. // Accumulator for correlation statistics calculated for each *i* value.
      ii. $Fs_i^k = random\_wo\_replacement(FeaturePool, k)$ // Choose the i^{th} *k* random features from the feature pool.
      iii. For j = 1 to 50 do // Counter for models that will be fit to *j* training data sets characterized by Fs_i^k.
         a. $C_j = 0$. // Accumulator for correlation statistics calculated for each *j* value.
         b. $\{Tr_j, Ts_j\} = Bootstrap(Sc, Fs_i^k, 2/3)$ // Sample the j^{th} training and test sets. The size of Tr_j= 2/3 of *Sc*, Ts_j is the remaining third.
         c. $Tr_j = \{X_r, Y_r\}_{r=1}^{2N/3} \quad Ts_j = \{X_s, Y_s\}_{s=1}^{N/3} \quad r \neq s$
         d. $f_j(X) = Fit(Tr_j, \Gamma)$ // Fit the model f_j to Tr_j using the fitting technique Γ.
         e. $C_j = C_j + correlation\left(\{Y_s\}_{s=1}^{N/3}, \{f_j(X_s)\}_{s=1}^{N/3}\right)$
      iv. End For loop of Step 2.a.iii
      v. $C_i = C_i + C_j/50$.
   B. End For loop of Step 2.a.
   C. $C_k = C_k \bigcup C_i / 50$
3. End For loop of Step 2.
4. Return C_k.

Figure 3.13 Algorithm to assess the impact of data dimension on performance of scoring functions.

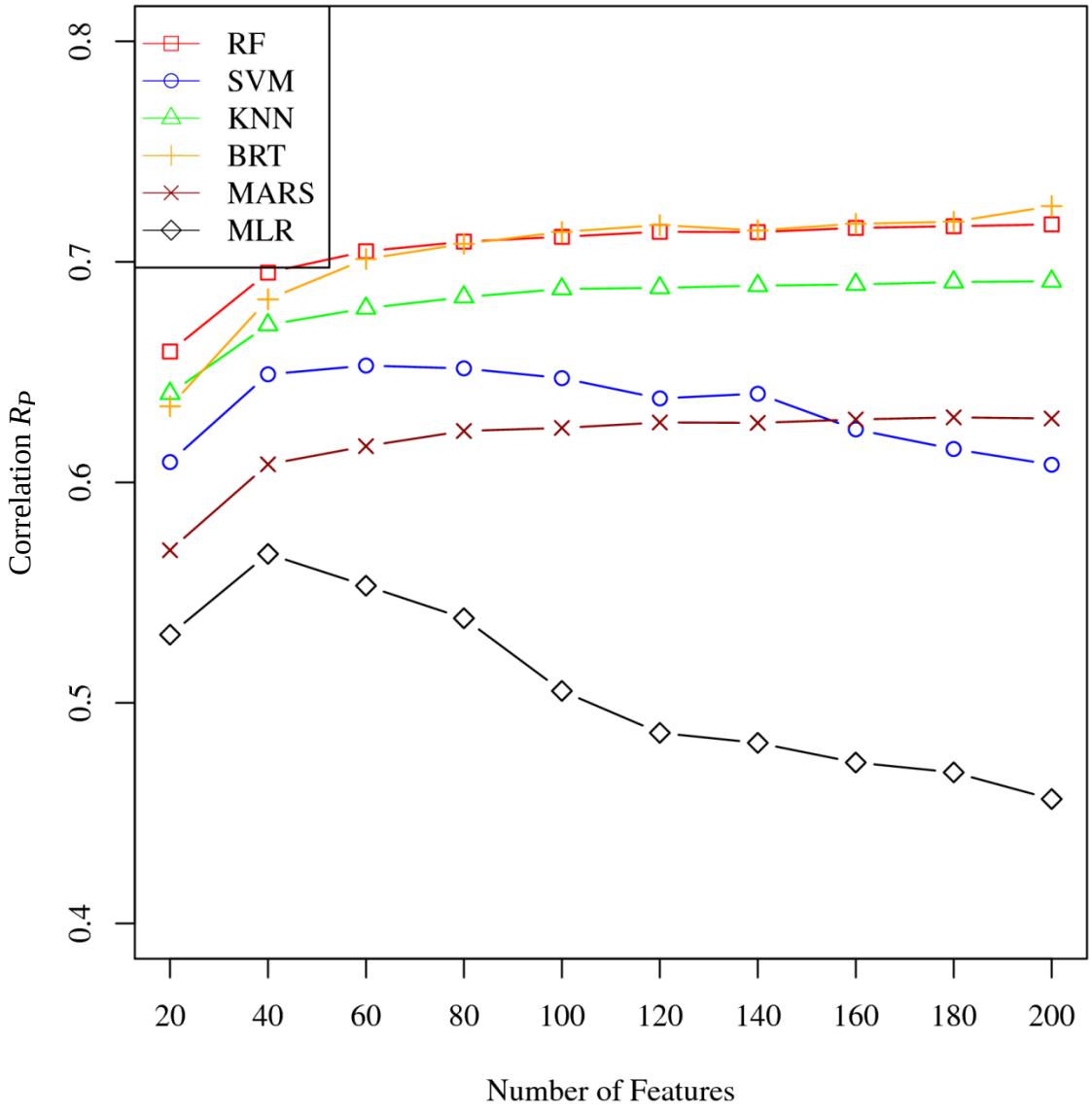


Figure 3.14 Data dimension impact on performance of scoring functions.

3.7 Family Specific Scoring Functions

In the previous section, performance of scoring functions was assessed on a diverse test set *Cr*. The core set consists of more than sixty different protein families. One may ask how these scoring functions perform on homogeneous datasets. In other words, how accurate are scoring functions in predicting binding affinities of ligands that form a complex with a common protein family? To answer this question, four additional test sets were constructed. Protein-ligand complexes in each set share a common protein family. The protein families are HIV protease, trypsin, carbonic anhydrase, and thrombin. The findings are shown in Table 3.6, where the results are ordered based on Spearman correlation coefficient. Thus the emphasis here is on the ranking correlation between observed binding scores and predicted ones. From the data in the table, however, we observe that scoring functions whose ranking correlation is high, their linear correlation is usually high as well, and vice versa.

It is interesting to see that scoring functions that perform well in case of predicting diverse protein-ligand complexes, are also among the best achievers in predicting binding scores of certain protein families. Scoring functions derived using BRT, RF, NF, SVM, and kNN approaches were superior in accurately predicting binding affinities for the heterogeneous test set. In the case of specific family protein-ligand complexes, these models' predictions are even more accurate. Across the four additional test sets there is no single model that always preserves remains on top. However, the top four machine-learning models are always the same across the four sets. The correlation coefficients of BRT, RF, NF, SVM, and kNN models approach excellent values (R_S and R_P = 0.91 to 0.98) with relatively low error variance (SD < 0.85 in -log K_d units given that the range of the binding affinities is from 1.01 to 12.1).

Table 3.6 Ranking accuracy of scoring functions on the four additional tests sets. The test samples overlap the training data.

HIV protease (N=112)				**Trypsin (N=73)**			
Scoring Functions	R_S	R_P	*SD*	Scoring Functions	R_S	R_P	*SD*
BRT::AR	0.980	0.973	0.376	SVM::A	0.974	0.954	0.509
RF::XA	0.976	0.96	0.457	RF::XA	0.940	0.958	0.484
SVM::A	0.963	0.955	0.487	BRT::AR	0.934	0.946	0.548
NF::XA	0.920	0.921	0.638	KNN::XA	0.927	0.943	0.563
KNN::XA	0.867	0.890	0.748	NF::XA	0.918	0.935	0.598
MARS::A	0.529	0.584	1.331	X-Score::HSScore	0.824	0.817	0.970
MLR::R	0.458	0.413	1.493	MLR::X	0.809	0.803	1.007
X-Score::HPScore	0.339	0.341	1.540	MARS::XR	0.808	0.820	0.967
SYBYL::ChemScre	0.228	0.276	1.580	DS::Ludi2	0.791	0.823	0.960
DS::PMF04	0.200	0.183	1.610	DS::PLP2	0.774	0.797	1.020
DrugScorePDB::PairSurf	0.170	0.225	1.600	SYBYL::ChemScore	0.773	0.829	0.950
Carbonic anhydrase (N=44)				**Thrombin (N=38)**			
Scoring Functions	R_S	R_P	*SD*	Scoring Functions	R_S	R_P	*SD*
SVM::A	0.929	0.909	0.580	BRT::AR	0.977	0.953	0.639
BRT::AR	0.919	0.959	0.393	RF::XA	0.972	0.952	0.647
RF::A	0.910	0.944	0.458	SVM::A	0.946	0.920	0.831
KNN::XA	0.910	0.910	0.577	NF::XAR	0.943	0.921	0.825
NF::XAR	0.889	0.931	0.508	KNN::XA	0.939	0.918	0.839
DS::PLP2	0.772	0.800	0.840	MARS::XR	0.733	0.784	1.317
MLR::XAR	0.709	0.747	0.926	DS::PLP1	0.672	0.692	1.530
SYBYL::G-Score	0.646	0.706	0.990	MLR::R	0.653	0.699	1.515
MARS::AR	0.645	0.734	0.945	SYBYL::G-Score	0.626	0.667	1.580
SYBYL::ChemScore	0.631	0.699	1.000	DrugScoreCSD::Pair	0.622	0.586	0.651
SYBYL::PMF-Score	0.618	0.627	1.090	X-Score::HSScore	0.586	0.666	1.580

Clearly, sophisticated machine-learning-based scoring functions exhibit high performance capabilities as compared to the less flexible models. This is obvious in case of HIV protease, where BRT::AR scoring function delivers accurate estimates (R_S = 0.98) as opposed to the embarrassing correlation value (R_S = 0.339) that X-Score::HPScore attains. This gap in performance is realized despite the fact that both models were trained on the same dataset (*Pr*)

when all proteins of the four families were included. Complete overlap between the family-specific datasets and the *Pr* set for the other 15 scoring functions cannot be verified.

Table 3.7 Ranking accuracy of scoring functions on the four additional tests sets. The test samples are independent of the training data.

HIV Protease (N=112)				**Trypsin (N=73)**			
Scoring Function	R_S	R_P	*SD*	Scoring Function	R_S	R_P	*SD*
MARS::X	0.484	0.458	1.457	X-Score::HSScore	0.824	0.817	0.970
MLR::XR	0.398	0.365	1.526	MLR::X	0.807	0.801	1.012
NF::AR	0.373	0.422	1.486	DS::Ludi2	0.791	0.823	0.960
X-Score::HPScore	0.339	0.341	1.540	MARS::A	0.784	0.739	1.138
BRT::A	0.331	0.415	1.491	DS::PLP2	0.774	0.797	1.020
RF::A	0.310	0.392	1.508	SYBYL::ChemScore	0.773	0.829	0.950
SYBYL::ChemScore	0.228	0.276	1.580	RF::XA	0.771	0.794	1.027
KNN::X	0.204	0.247	1.588	KNN::XAR	0.769	0.792	1.031
SVM::X	0.200	0.267	1.579	NF::AR	0.767	0.763	1.045
DS::PMF04	0.200	0.183	1.610	BRT::XA	0.766	0.78	1.058
DrugScorePDB::PairSurf	0.170	0.225	1.600	SVM::XA	0.737	0.787	1.044
Carbonic anhydrase (N=44)				**Thrombin (N=38)**			
Scoring Functions	R_S	R_P	*SD*	Scoring Functions	R_S	R_P	*SD*
DS::PLP2	0.772	0.800	1.250	MARS::RF	0.721	0.667	1.580
BRT::A	0.698	0.735	0.944	KNN::X	0.711	0.737	1.432
SYBYL::G-Score	0.646	0.706	0.990	RF::XA	0.704	0.730	1.448
RF::XA	0.632	0.652	1.056	NF::X	0.696	0.749	1.404
MLR::XR	0.631	0.716	0.972	DS::PLP1	0.672	0.692	1.530
SYBYL::ChemScore	0.631	0.699	1.000	SVM::X	0.667	0.720	1.471
SYBYL::PMF-Score	0.618	0.627	1.090	BRT::XR	0.656	0.718	1.475
MARS::XAR	0.566	0.555	1.158	MLR::R	0.640	0.676	1.561
KNN::XA	0.531	0.717	0.970	SYBYL::G-Score	0.626	0.667	1.580
SVM::AR	0.426	0.490	1.214	DrugScoreCSD::Pair	0.622	0.651	1.610
NF::AR	0.421	0.597	1.117	X-Score::HSScore	0.586	0.666	1.580

In addition to their low generalization error, BRT, RF, NF, SVM, and kNN models are known for being adept at predicting well for data patterns on which they are trained. That is perhaps due to their high-degree of nonlinearity and flexibility. Linear models on the other hand are rigid and

tend to be less amenable to presence of particular data patterns. MARS combines characteristics of both families of models. MARS nonlinearity is achieved by combining several linear models in a piece-wise fashion.

We retrained BRT, RF, NF, SVM, kNN, MARS, and MLR models on the primary training dataset after we removed complexes whose proteins belong to the four protein families under consideration. More specifically, for scoring functions that will be evaluated on a particular protein family, all complexes of that protein family were filtered out from the *Pr* set before training. Table 3.7 shows performance statistics of the resulting scoring functions on the four proteins datasets. RF, NF, BRT, SVM, and kNN are now inferior to linear and MARS based scoring functions. In fact, correlation coefficients of almost all machine-learning models have dropped drastically when they are reevaluated on disjoint datasets. This is particularly true in case of HIV protease, where none of the scoring functions reach a value of 0.5 for both R_P and R_S. Such a poor performance can perhaps be attributed to difficulty in sampling conformational changes between HIV protease and ligands [32]. Besides, scoring functions are poor in accounting for enthalpic and entropic factors upon binding. By contrast, scoring functions predicted binding energies more accurately in the other three protein-family datasets. Correlation coefficients in such datasets are not far from those obtained on the diverse test set.

What if the purpose of a screening campaign was to identify new ligands for HIV protease or to conduct lead optimization? We saw how scoring function accuracy suffers when there are no known binding ligands to this protein in the training set. How much the predictive power of scoring functions can change if we recalibrate them by taking into account several ligands that we

know bind to HIV protease? Figure 3.15 answers this question for different values of the number of known binders. These values are expressed as percentage of the total number of complexes in

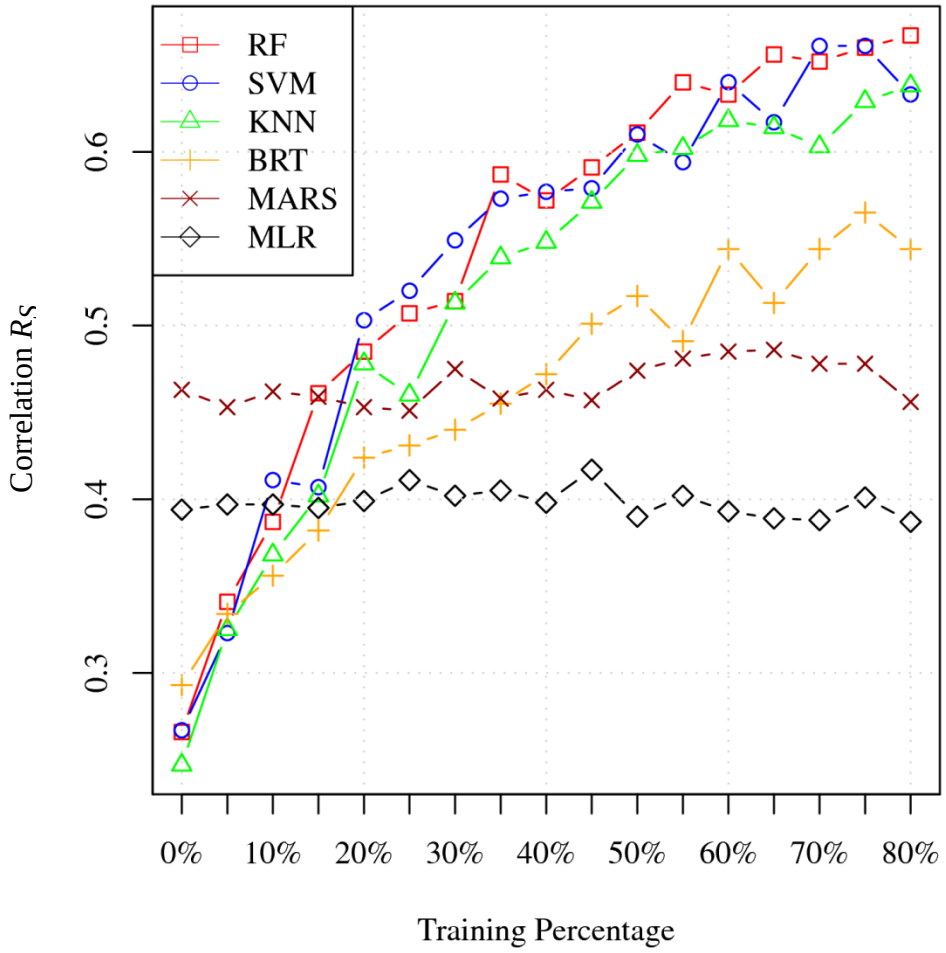


Figure 3.15 Dependence of scoring functions on the number of known binders to HIV protease in the training set.

HIV protease dataset (112). For instance, 20% means that 22 protein-ligand complexes from HIV protease dataset were added to the primary training set Pr_X. After reconstructing scoring functions on the updated version of *Pr*, they are evaluated on the other 80% of the HIV data (i.e., 90

complexes). For each percentage (0% to 80% in steps of 5%), this procedure was repeated 50 times to collect reliable results.

When the training dataset includes none to very few HIV protease complexes (roughly less than 10), we notice the curves in the plot are consistent to some extent with the data presented in Table 3.7. In such a scenario, MLR and MARS models outperform other scoring functions. As the number of HIV complexes increase in the training set, the accuracy of both models does not show clear responsiveness to changes in training data. On the other hand, RF, SVM, and kNN models gain sharp increase in performance as patterns of HIV protease become more and more prevalent in the training data. For the BRT::X scoring function, the improvement is less pronounced. This is surprisingly not satisfying for a BRT model given its high performance profile shown in other test sets.

It should be noted that results given in Table 3.7 of some machine-learning-based scoring functions were fitted to datasets characterized with different feature types. On the other hand, the curves plotted in Figure 3.15 show the performance of functions driven by a complex set (*Pr*) identified by the X features only. Curves obtained on other feature sets are not very different (not reported in this text).

The analysis of the effectiveness of different regression approaches on specific families of proteins has very useful practical advantages in virtual screening and drug discovery. Typically, the task in drug design is to conduct computational screening of large numbers of ligands against a certain protein family. Given the number of known binders to that protein present in the training set and having scoring functions' characteristic plot (similar to the one in Figure 3.15) could help specialists to utilize the correct scoring function.

3.8 Performance of Scoring Functions on Novel Targets

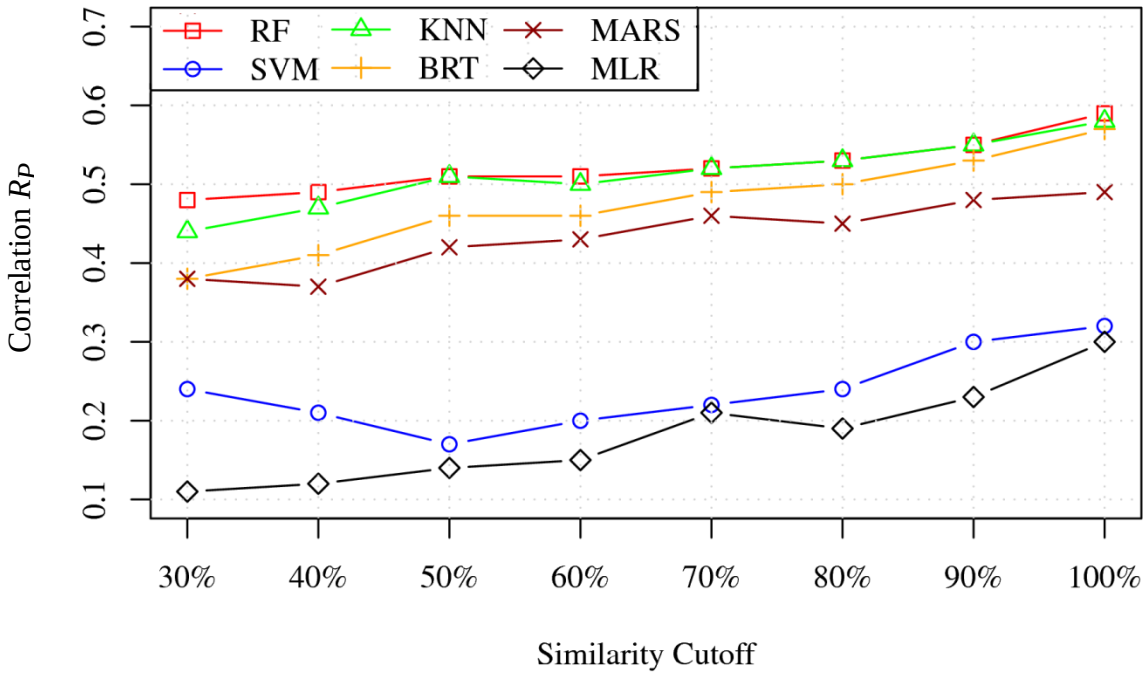
In Section 3.4, we investigated the ability of different scoring functions to predict the binding affinities of diverse protein-ligand complexes that constitute the core test set *Cr*. This test set was constructed by Cheng et al. by including three representative protein-ligand complexes from all clusters containing at least four complexes that were found in the 2007 PDBbind refined set based on BLAST sequence similarity (with a 90% cutoff) [32]. The complexes in each cluster typically consisted of those formed by a particular type of protein. The primary training set *Pr* is comprised of refined set complexes not included in *Cr*, thus ensuring there are no complexes common to both *Pr* and *Cr*. The way the training-test set pair (*Pr*, *Cr*) is formed ensures that: (1) *Pr* has complexes that are relevant to the test set *Cr* since *Pr* contains at least one complex from each cluster represented in *Cr*; (2) *Cr* has maximal diversity, in terms of both protein and ligand molecules in a complex, and minimal redundancy: all clusters with four or more complexes are represented with only three complexes per cluster (or type of protein), binding constants of complexes span over 12 orders of magnitude, and ligand molecular weights and number of rotatable bonds range from 103 to 974 and 0 to 32, respectively; (3) *Cr* is suitable for not only assessing the scoring power of scoring functions, but also their ranking power since it contains three complexes from each cluster (or type of protein). The training-test set pair (*Pr*, *Cr*) (or similarly (*Sc*, *Cr*)) is a useful benchmark when the aim is to evaluate the performance of scoring functions on targets that have some degree of sequence similarity with at least one protein present in the complexes of the training set. This is typically the case since, in practice, it has been observed that more than 92% of today's drug targets are similar to known proteins in the PDB [108], high-quality complexes from which are included in the PDBbind refined set [61].

When the goal is to assess scoring functions in the context of novel protein targets, however, (*Pr*, *Cr*) (or similarly (*Sc*, *Cr*)) is not that suitable since the training set has at least one complex with a protein represented in the core set. We considered this issue to some extent in the previous section (see Table 3.7), where we investigated the performance of scoring functions on four different protein-specific test sets after training them on complexes that did not have the protein under consideration. This resulted in a drop in performance of all scoring functions, especially, in the case of HIV protease as target. However, even if there are no common proteins between training and test set complexes, different proteins at their binding sites may have sequence and structural similarity, which influence protein-ligand binding affinity. To more rigorously and systematically assess scoring function performance on novel targets, we performed a separate set of experiments in which we limited BLAST sequence similarity between the binding sites of proteins present in the training and test set complexes.

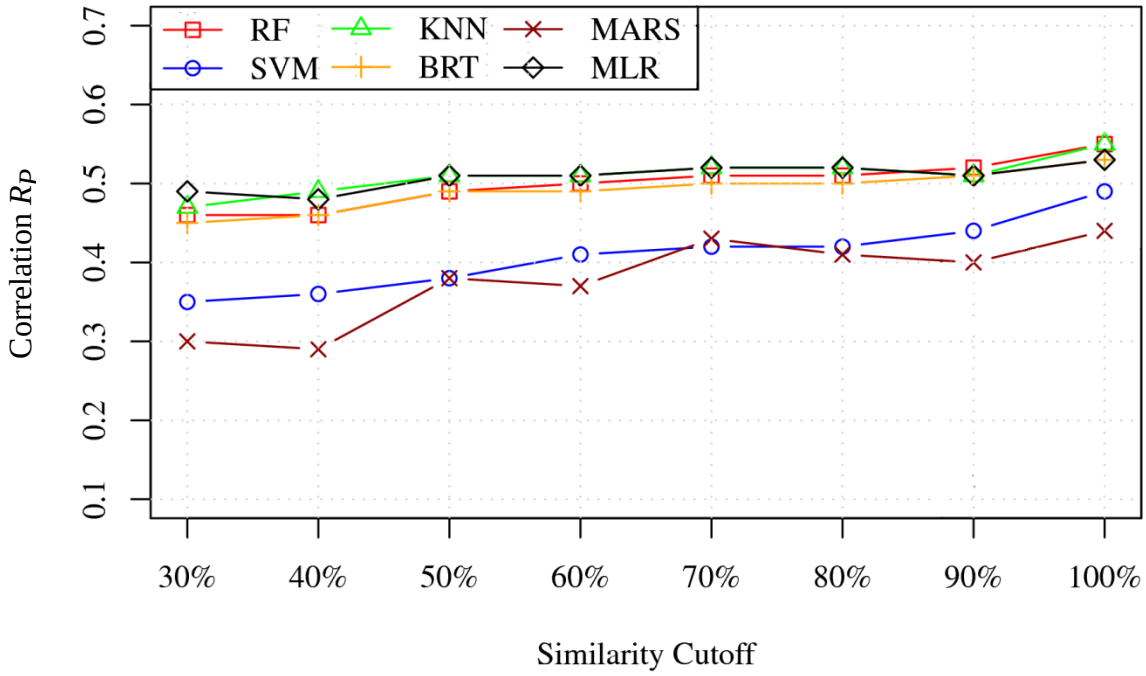
Specifically, for each similarity cut-off value S = 30%, 40%, 50%, …, 100%, we constructed 100 different independent 150-complex test and T-complex training set pairs, trained the scoring models (MLR, MARS, kNN, SVM, BRT, and RF) using two different feature sets (XAR and X features) on the training set and evaluated them on the corresponding test set, and determined their average performance over the 100 rounds to obtain robust results. Since scoring function prediction performance depends upon both similarity cutoff and training set size and since training set size is constrained by similarity cutoff (a larger S means a larger feasible T), we investigated different ranges of S (30% to 100%, 40% to 100%, 50% to 100%, etc.) and for each range set T close to the largest feasible for the smallest S value in that range. Each test and training set pair was constructed as follows. We randomly sampled a test set of 150 protein-ligand complexes without replacement from all complexes at our disposal: 1986 in *Sc* + 192 in

Cr = 2178 complexes. The remaining 2028 complexes were randomly scanned until *T* different complexes were found that had protein binding site similarity of *S*% or less with the protein binding sites of all complexes in the test set – if less than *T* such complexes were found, then the process was repeated with a new 150-complex test set.

The accuracy of the five scoring models in terms of Pearson correlation coefficient is depicted in Figure 3.16 for a similarity cutoff range from 30% to 100% and for a training set size of 120 complexes, which was about the largest training set size feasible for S = 30%. The upper plot shows these results when protein-ligand complexes are described by XAR features and the bottom plot when such compounds are characterized by X-Score (X) features alone. When XAR features are used, RF and kNN perform the best across the entire range of similarity cutoffs. When only the six X features are used, MLR, kNN, RF, and BRT perform the best. In both feature set cases, we see that no scoring function is able to predict the measured binding affinity with good accuracy ($R_P < 0.5$ when similarity cutoff = 30%). This can be attributed to three factors. First is the significant sequence, and hence structural, dissimilarity between the binding sites of proteins in training and test complexes. Second is the size of the training data. We saw how important the size of training data is in influencing the quality of scoring models in Section 3.5. Finally, the scoring model parameters used for this experiment were based on parameter tuning with respect to *Pr* as described in Section 3.2 rather than with respect to the training set used for this experiment, which is not only small but has a different mix of complexes governed by the small similarity cutoff of 30%. This is evident in Figure 3.16 from the inferior performance of several scoring functions (MLR, SVM, and, to a limited extent, BRT) when XAR features are used instead of X features (which is a subset of XAR features), signifying overfitting or suboptimal parameter values. The reason the RF model had relatively good



a - Complexes are characterized using XAR features.



b - Complexes are characterized using X features.

Figure 3.16 Performance of scoring models as a function of BLAST sequence similarity cutoff between binding sites of proteins in training and test complexes.

performance compared to the other generally-competitive machine-learning models for both XAR and X features is most likely because it is less sensitive to parameter tuning (see Section 3.2.4). Due to the time-intensive nature of tuning parameters of multiple scoring models for the many training scenarios we consider in this work, we used the parameters tuned with respect to *Pr* as explained in Section 3.2.

Figure 3.16 provides only a partial portrait of how sensitive scoring model accuracy is to similarity cutoff because of the small training dataset (only 120 complexes) used. To more fully capture this sensitivity, we need larger training datasets. Therefore, we determined the performance of scoring models for a number of similarity cutoff ranges (30% to 100%, 40% to 100%, etc. with increasingly larger training datasets) and these results are tabulated in Table 3.8. For a given similarity cutoff, not only does the scoring performance of models improve with the size of training dataset, the overfitting/parameter-tuning problem also gets alleviated or minimized. For example, for a similarity cutoff of 50%, BRT::XAR has inferior performance relative to BRT:X when training dataset has 120 complexes (0.46 vs. 0.49), but the opposite is true when the training set has 520 complexes (0.54 vs. 0.50). Also, the performance of RF and the other generally-competitive machine-learning models (BRT, kNN, and SVM) becomes comparable as training dataset size increases, signifying that parameter tuning becomes less of an issue.

Based on the data in Table 3.8, we plot in Figure 3.17 the percentage improvement in Pearson correlation coefficient of RF::XAR, relative to that for the smallest training dataset size (120 complexes), as the size of the training set increases. The reason for focusing on RF::XAR is that it is not only among the best scoring functions, but also, as noted earlier, its performance is less sensitive to parameter tuning and it performs better relative to other scoring functions when

considering the entire range of similarity cutoffs. As expected, for any given similarity cutoff, RF::XAR's accuracy improves with the size of training set. Moreover, increase in training set size helps improve accuracy somewhat more at larger than at smaller similarity cutoffs, although continuing gains in accuracy are evident for all similarity cutoffs at the largest training set sizes considered.

Table 3.8 Performance of scoring models as a function of BLAST sequence similarity cutoff between binding sites of proteins in training and test complexes for different cutoff ranges.

	XAR Features						**X Features**					
Similarity	**Training Set Size = 120**						**Training Set Size = 120**					
Cutoff %	MLR	MARS	kNN	SVM	BRT	RF	MLR	MARS	kNN	SVM	BRT	RF
30	0.11	0.38	0.44	0.24	0.38	0.48	0.49	0.30	0.47	0.35	0.45	0.46
40	0.12	0.37	0.47	0.21	0.41	0.49	0.48	0.29	0.49	0.36	0.46	0.46
50	0.14	0.42	0.51	0.17	0.46	0.51	0.51	0.38	0.51	0.38	0.49	0.49
60	0.15	0.43	0.50	0.20	0.46	0.51	0.51	0.37	0.51	0.41	0.49	0.50
70	0.21	0.46	0.52	0.22	0.49	0.52	0.52	0.43	0.52	0.42	0.50	0.51
80	0.19	0.45	0.53	0.24	0.50	0.53	0.52	0.41	0.52	0.42	0.50	0.51
90	0.23	0.48	0.55	0.30	0.53	0.55	0.51	0.40	0.51	0.44	0.51	0.52
100	0.30	0.49	0.58	0.32	0.57	0.59	0.53	0.44	0.55	0.49	0.53	0.55
Similarity	**Training Set Size = 220**						**Training Set Size = 220**					
Cutoff %	MLR	MARS	kNN	SVM	BRT	RF	MLR	MARS	kNN	SVM	BRT	RF
40	0.22	0.39	0.47	0.35	0.46	0.51	0.50	0.39	0.49	0.40	0.47	0.48
50	0.26	0.45	0.51	0.34	0.5	0.53	0.53	0.44	0.51	0.42	0.50	0.51
60	0.28	0.46	0.5	0.35	0.49	0.52	0.52	0.41	0.51	0.43	0.49	0.51
70	0.33	0.47	0.53	0.40	0.52	0.54	0.53	0.46	0.52	0.44	0.51	0.52
80	0.35	0.49	0.55	0.38	0.54	0.56	0.53	0.46	0.52	0.45	0.51	0.53
90	0.36	0.50	0.57	0.46	0.56	0.58	0.52	0.44	0.53	0.48	0.52	0.54
100	0.42	0.54	0.61	0.50	0.61	0.62	0.54	0.49	0.57	0.53	0.55	0.57
Similarity	**Training Set Size = 520**						**Training Set Size = 520**					
Cutoff %	MLR	MARS	kNN	SVM	BRT	RF	MLR	MARS	kNN	SVM	BRT	RF
50	0.36	0.48	0.52	0.53	0.54	0.55	0.53	0.46	0.51	0.44	0.50	0.52
60	0.38	0.48	0.52	0.53	0.54	0.55	0.53	0.48	0.52	0.46	0.51	0.53
70	0.42	0.51	0.55	0.55	0.56	0.57	0.54	0.50	0.52	0.47	0.52	0.54
80	0.46	0.52	0.58	0.57	0.59	0.59	0.54	0.52	0.54	0.49	0.53	0.55
90	0.49	0.54	0.61	0.60	0.61	0.62	0.53	0.53	0.57	0.53	0.56	0.57
100	0.55	0.58	0.65	0.64	0.66	0.67	0.55	0.54	0.61	0.58	0.60	0.61
Similarity	**Training Set Size = 720**						**Training Set Size = 720**					
Cutoff %	MLR	MARS	kNN	SVM	BRT	RF	MLR	MARS	kNN	SVM	BRT	RF
60	0.42	0.50	0.53	0.54	0.55	0.56	0.53	0.50	0.52	0.46	0.52	0.53
70	0.46	0.52	0.56	0.57	0.57	0.58	0.54	0.53	0.53	0.49	0.53	0.55
80	0.49	0.53	0.59	0.59	0.60	0.60	0.54	0.54	0.55	0.50	0.54	0.56

Table 3.8 (cont'd)

90	0.52	0.56	0.62	0.61	0.63	0.64	0.53	0.54	0.57	0.54	0.57	0.58
100	0.56	0.6	0.67	0.67	0.68	0.69	0.55	0.58	0.62	0.59	0.62	0.63
Similarity	**Training Set Size = 1020**						**Training Set Size = 1020**					
Cutoff %	MLR	MARS	kNN	SVM	BRT	RF	MLR	MARS	kNN	SVM	BRT	RF
70	0.49	0.54	0.57	0.58	0.59	0.59	0.54	0.52	0.53	0.50	0.55	0.56
80	0.54	0.55	0.60	0.60	0.62	0.62	0.54	0.54	0.55	0.52	0.56	0.57
90	0.55	0.58	0.63	0.63	0.64	0.65	0.53	0.56	0.58	0.56	0.58	0.59
100	0.59	0.62	0.69	0.68	0.70	0.70	0.55	0.59	0.63	0.61	0.63	0.65
Similarity	**Training Set Size = 1220**						**Training Set Size = 1220**					
Cutoff %	MLR	MARS	kNN	SVM	BRT	RF	MLR	MARS	kNN	SVM	BRT	RF
80	0.55	0.55	0.61	0.61	0.62	0.63	0.54	0.55	0.56	0.53	0.57	0.57
90	0.56	0.59	0.64	0.64	0.65	0.66	0.54	0.57	0.59	0.57	0.59	0.60
100	0.60	0.63	0.69	0.69	0.71	0.71	0.55	0.60	0.64	0.61	0.64	0.65
Similarity	**Training Set Size = 1620**						**Training Set Size = 1620**					
Cutoff %	MLR	MARS	kNN	SVM	BRT	RF	MLR	MARS	kNN	SVM	BRT	RF
90	0.58	0.61	0.65	0.65	0.67	0.67	0.54	0.57	0.60	0.58	0.60	0.60
100	0.61	0.65	0.71	0.70	0.72	0.73	0.55	0.60	0.65	0.63	0.65	0.66
Similarity	**Training Set Size = 2028**						**Training Set Size = 2028**					
Cutoff %	MLR	MARS	kNN	SVM	BRT	RF	MLR	MARS	kNN	SVM	BRT	RF
100	0.62	0.66	0.72	0.72	0.73	0.74	0.55	0.61	0.66	0.64	0.65	0.67

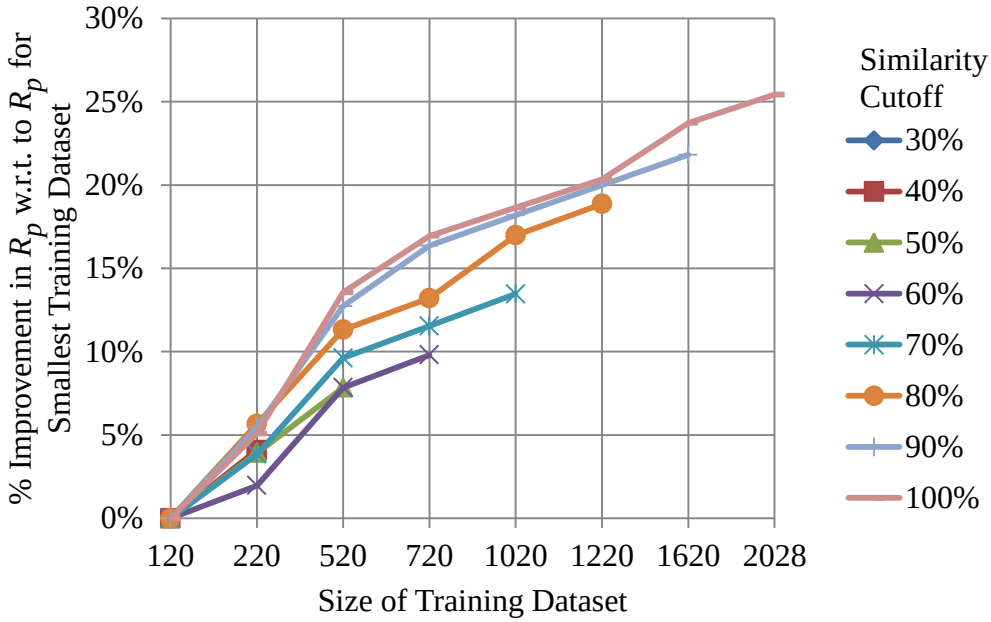


Figure 3.17 Sensitivity of RF::XAR scoring model accuracy to size of training dataset and BLAST sequence similarity cutoff between binding sites of proteins in training and test complexes.

To summarize, imposing a sequence similarity cutoff between the binding sites of proteins in training and test set complexes has an expected adverse impact on the accuracy of all scoring models. However, increasing the number of training complexes helps improve accuracy for all similarity cutoffs. RF::XAR has the best accuracy considering the entire range of similarity cutoffs. The other generally-competitive machine-learning models (BRT, kNN, and SVM) may also provide comparable accuracy if parameter tuning is performed with respect to the training set being considered. The fact that the largest training sets that could be obtained from the 2007 and 2010 PDBbind refined sets (which is representative of PDB) at low similarity cutoffs were small is a reflection of real-world drug targets most often being similar to known proteins in the PDB [108].

3.9 Interpreting Scoring Models

Although predictive power determines success or failure of screening campaigns, it is still not the only characteristic of scoring function we desire to improve. In some cases, the descriptive power of a model is as important as its accuracy. Self-explanatory models enable drug designers, for instance, to gain deeper insights to interactions between a given protein and its ligand. Ensemble methods (BRT, RF, and NF) we employ in our study possess excellent tools that allow users to decipher much of the relationship between protein-ligand interactions and binding affinity. As pointed out in Section 2.5.2, these methods compute the relative variable importance and their marginal effects on the response variable. This descriptive information will be obtained using BRT and RF methods in the following subsections.

3.9.1 Relative Importance of Protein-Ligand Interactions on Binding Affinity

Since BRT and RF employ different schemes to calculate relative variable relevance, we here show the results of both techniques. Out of the 72 features of the dataset Sc_{XAR}, we will calculate the most important ten features as well as the least relevant five. Figure 3.18 shows estimation of the most important ten features using both RF (red, top) and BRT (blue, bottom) techniques. It is clear that the number of hydrophobic contacts has the strongest effect on binding affinity. It is interesting to see that this feature (the number of protein-ligand carbon-carbon contacts) is identified by both methods as the highest contributor to the change in binding affinity. That is the case even though BRT and RF estimate this contribution differently. In fact, they overlap in four other features which include van der Waals interactions between the protein and the ligand, the hydrophilicity of all protein interfacial atoms, and two hydrophobicity related features.

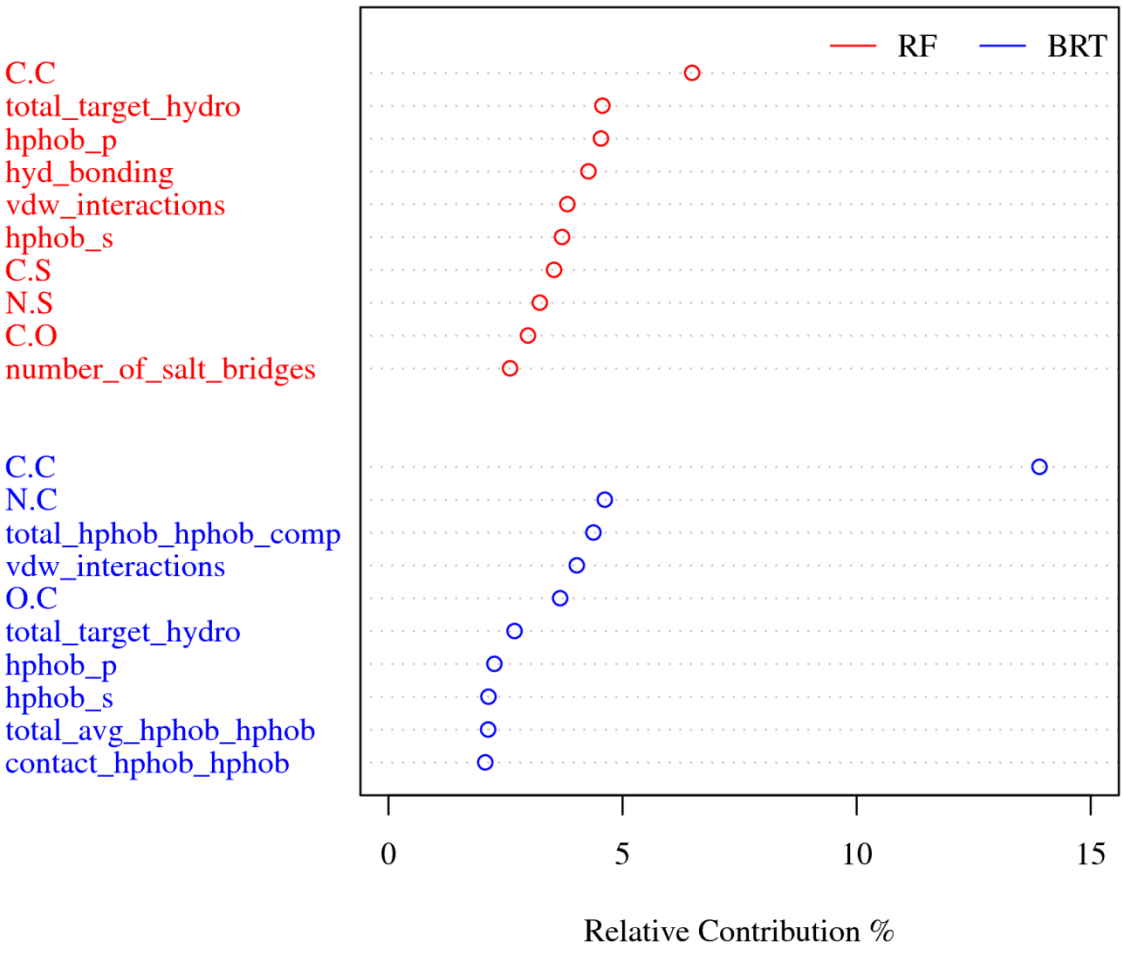


Figure 3.18 The most important ten features identified by RF and BRT algorithms. Refer to Table 2.1 for feature description.

Even more interestingly, the least five contributing features identified by BRT and RF are almost identical. As shown in Figure 3.19, the number of contacts between protein atoms (C, N, O, and S) and iodine in ligands has no effect (0%) on calculating the binding affinity.

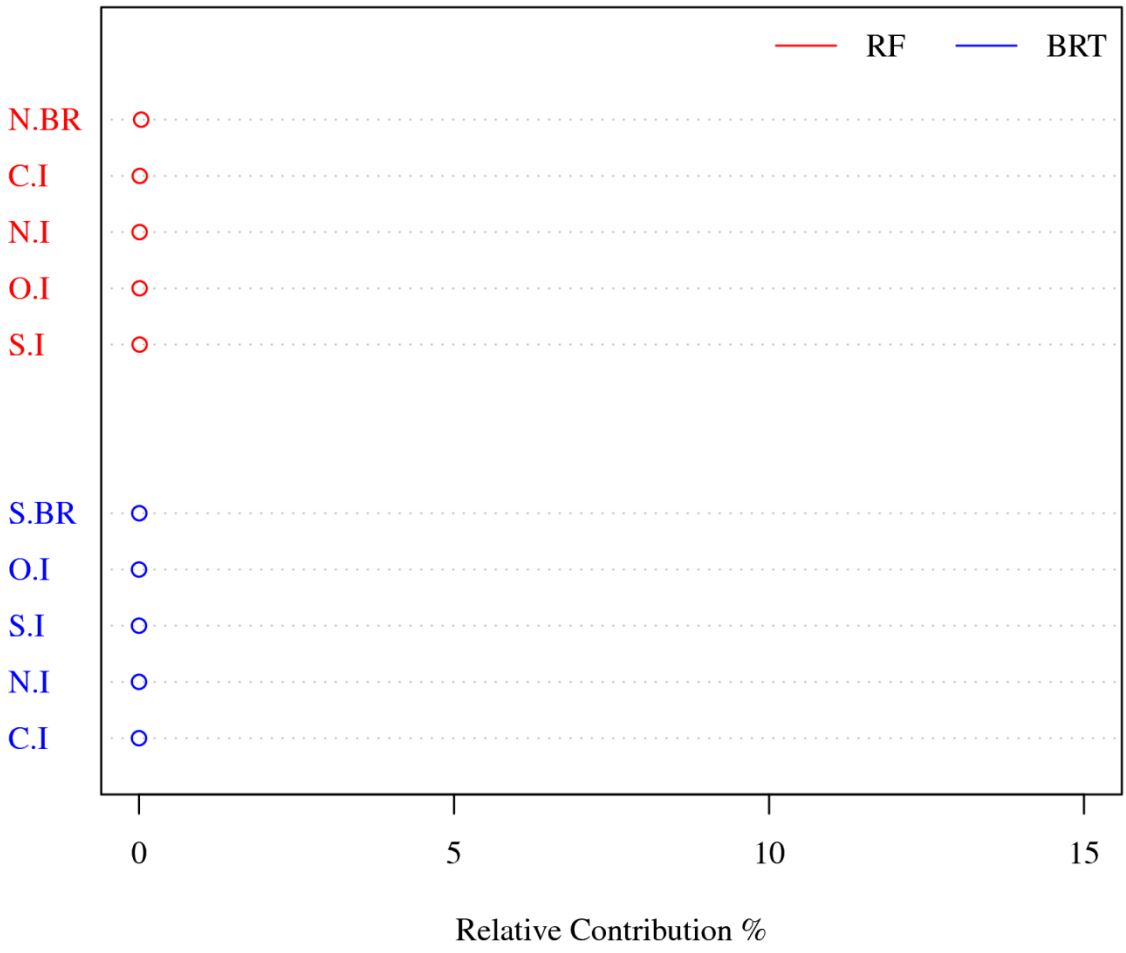


Figure 3.19 The least important five features identified by RF and BRT algorithms. Refer to Table 2.1 for feature description.

3.9.2 Partial Dependence plots

We used here the BRT tool to produce the partial dependence plots of the most relevant features identified in the previous section. Figure 3.20 shows the marginal plots of the most important four features that were seen overlapping between RF and BRT. For each variable, the effect of the other 71 features was averaged out according to Equation 2.18. It is apparent that the binding affinity increases by increasing the number of hydrophobic contacts and van der Waals interactions between proteins and ligands.

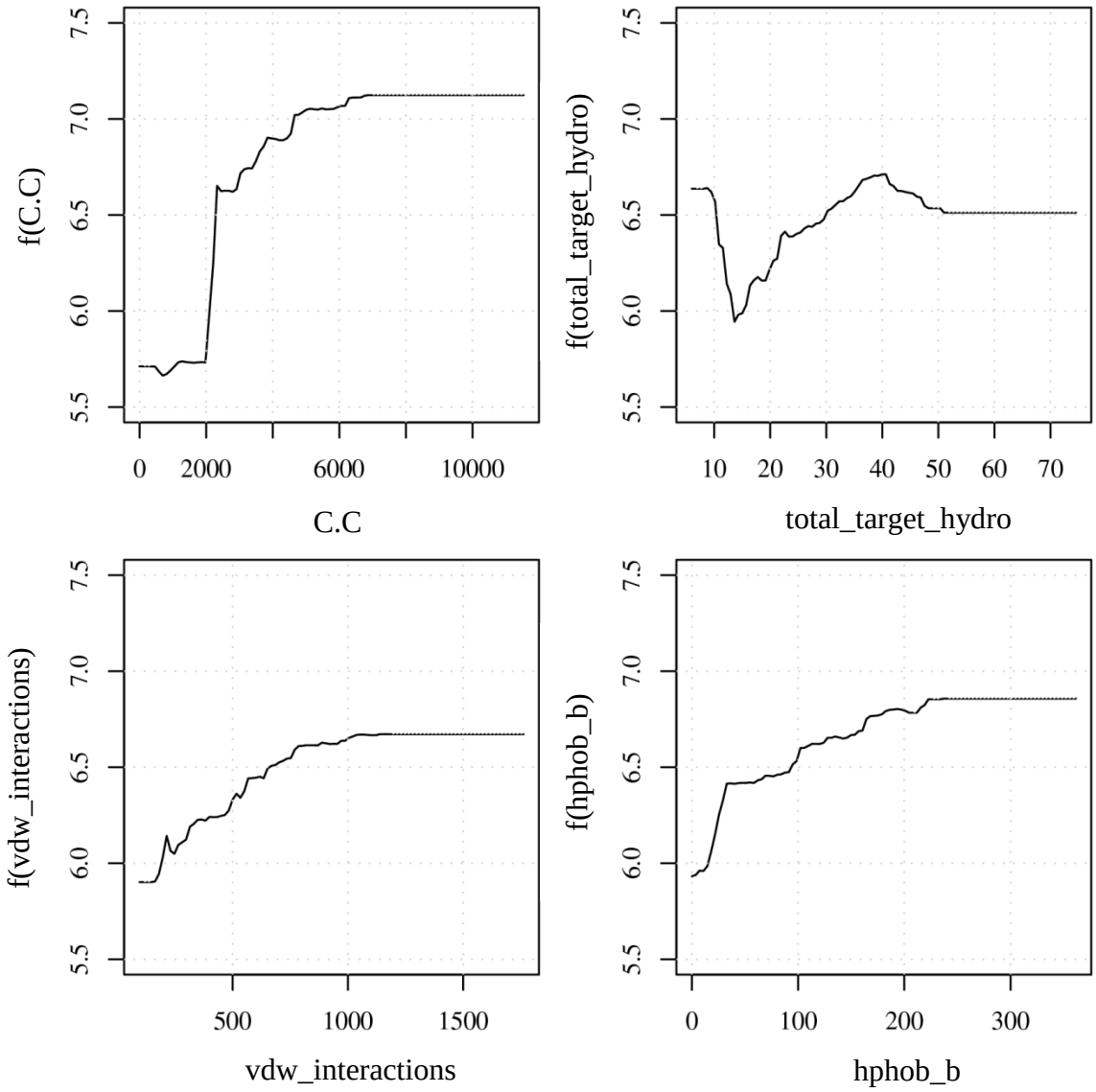


Figure 3.20 Marginal dependence plots of most relevant four features.

On the other hand, the marginal effect of hydrophilic interfacial atoms of proteins is less clear. The binding affinity sharply declines when the hydrophilicity starts to increase from 10. Once it reaches 15, the binding affinity rises almost linearly to peak at hydrophilicity of 40 before it declines again.

The marginal plots of the least influential features corroborate the results we obtained in the previous subsection. Figure 3.21 shows that no feature among the least important four has any effect on estimating binding affinity.

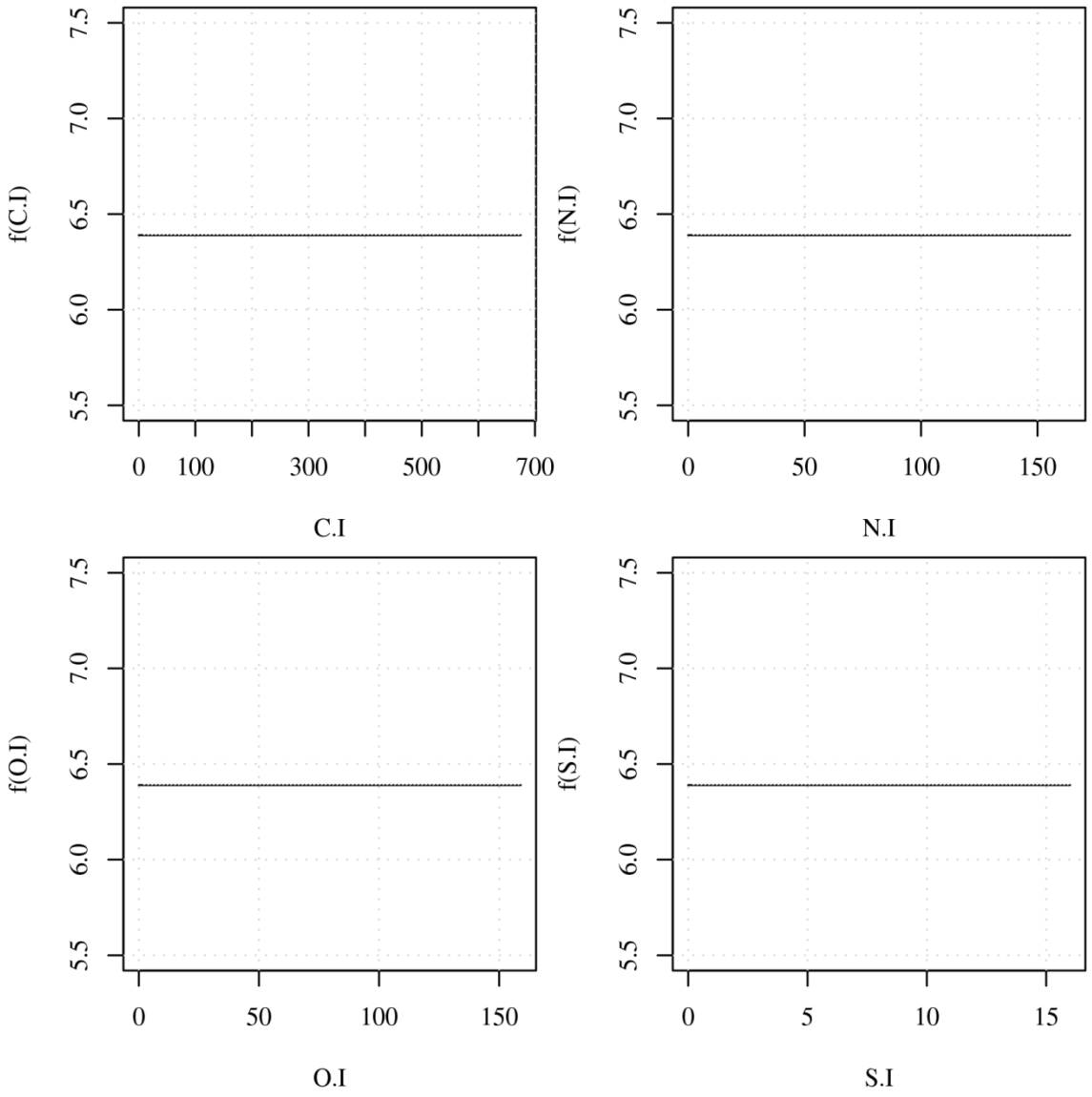


Figure 3.21 Marginal dependence plots of least relevant four features.

The question may arise: how much change in performance of the RF::XAR scoring function will occur when only the most important features are taken into account? To answer this

question, we built 71 versions of the RF::XAR scoring function. The first version considered only the most two contributing features (the number of protein-ligand carbon-carbon contacts and total hydrophilicity of all protein interfacial atoms.) The three most important input features were accounted for in the second version and we proceeded in this fashion until all the 72 features were considered in the seventy-first scoring function (i.e., regular RF::XAR). Each resultant version of the scoring function was evaluated on the core test set Cr_{XAR} and the corresponding Pearson's correlation coefficient between the predicted binding affinity and the measured values was recorded. These statistics are represented by the blue curve in Figure 3.22. The red curve shows the relative influence of the least important feature taken into account to build each version of the seventy-one models.

It is clear from the plots that adding more features to RF models renders them more accurate. The improvement is very noticeable in the early versions where the performance (in terms of R_P) increases by more than 20% when eighteen more features are added to the model that accounts for just the most important two. The improvement trend still holds (but with smaller slope) even when the least influential features (that have virtually zero influence) are considered in the later RF::XAR versions. Since accounting for such features does not harm the scoring function's performance, we recommend keeping them as long as they are relevant to estimating the binding affinity. In fact, these features may have some contribution to the overall binding affinity when they interact with other features. We cannot elucidate the interaction effects between the input features (if there are any) from the results depicted in Figure 3.20 and Figure 3.21. Instead, techniques other than those discussed in Section 2.5.2 can be used to analyze and visualize the interplay between the input variables. Such techniques will be investigated in future work.

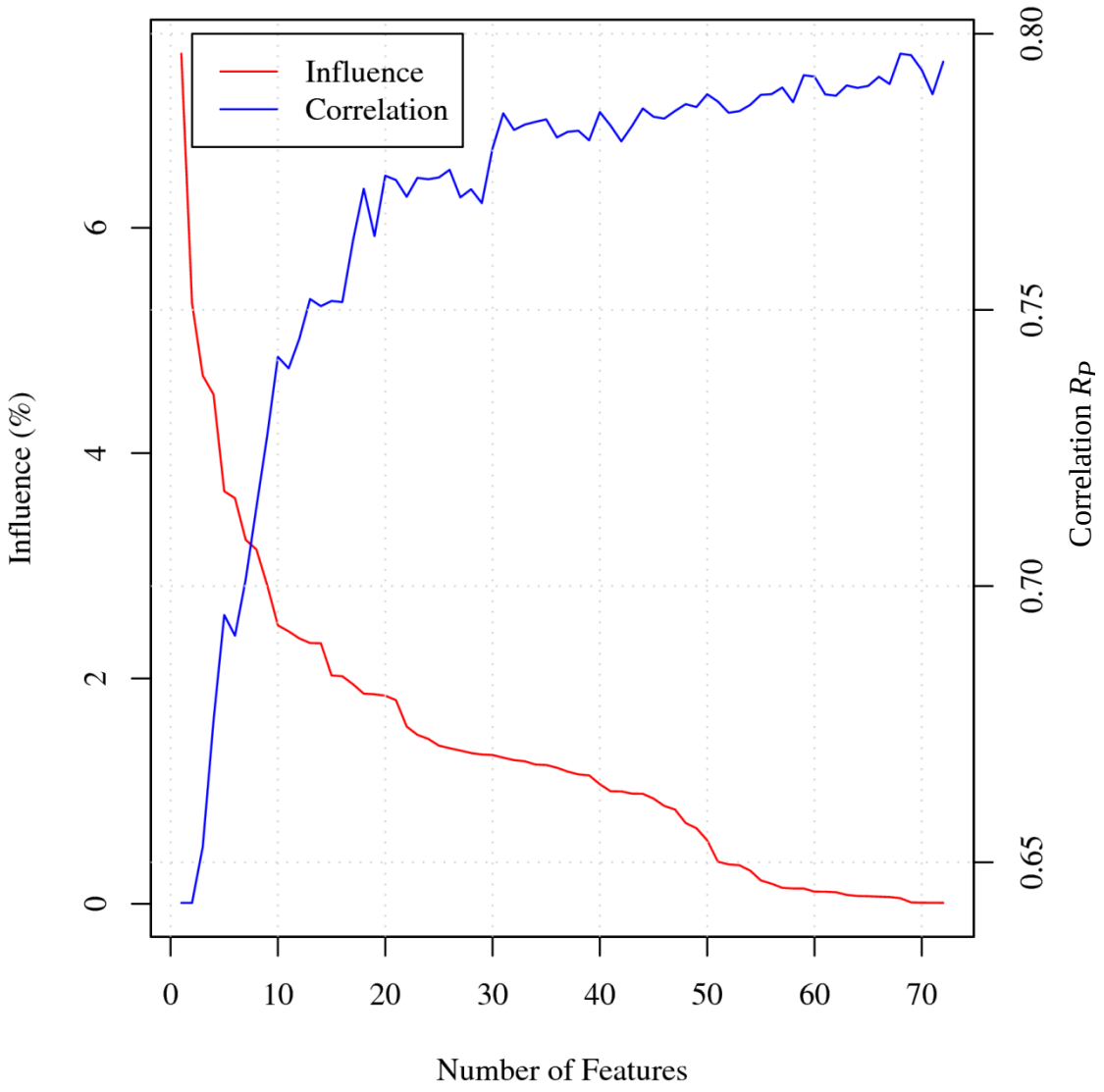


Figure 3.22 The RF::XAR scoring model's performance as a function of inclusion of increasing numbers of the most important features.

3.10 Scoring Throughput

After isolating it from docking, a scoring function's throughput depends on two factors. The first, which has the highest contribution, is feature extraction. The time required for calculating features typically used by empirical scoring functions is the bottleneck of scoring speed. This conclusion comes from an experiment we conducted whose results are listed in Table 3.9. In this experiment, we recorded the average time taken to calculate the features of each protein-ligand complex in the primary training set *Pr* and the core test set *Cr* (1278 complexes). Extraction tools X-Score v1.2, SLIDE v3.3.5, and RF-Score were executed on a 2.53 GHz-PC running Linux-Ubuntu 10.04. Clearly, X-Score is the slowest in that it spends 27 seconds on average to compute its six features (listed in Table 2.1) for a protein-ligand complex. It is about 77 times slower than AffiScore which also utilizes empirical energy terms.

Table 3.9 Feature extraction time (in seconds) and performance (R_P) of some prediction techniques fitted to these features.

Features	Time in sec.	MLR	RF	BRT	NF
X-Score	27	0.644	0.727	0.706	0.751
AffiScore (raw)	0.35	0.663	0.755	0.770	0.771
AffiScore (subset)	0.35	0.563	0.708	0.689	0.766
RF-Score	0.18	0.590	0.763	0.741	0.751

Interestingly, AffiScore is not only faster, scoring functions fitted to its features are also more accurate. That is true only if the raw features computed by AffiScore are used to construct the predictive model. The linear regression model fitted to Pr_A (i.e., the primary training complexes described by AffiScore raw features) achieves an accuracy of 0.663 in terms of Pearson correlation coefficient (R_P). The original implementation of AffiScore scoring function that uses

rather a subset of the raw features yields an accuracy of only R_P = 0.563. This latter scoring function is inferior to X-Score v1.2 whose correlation R_P is 0.644. AffiScore (raw) features are better than their X-Score counterparts not only for linear models, but also for the best performing ML techniques as shown in the last three columns of Table 3.9.

As pointed out in Section 2.2, the RF-Score tool employs the simplest algorithm to generate features. That explains why RF-Score's 36 features were the fastest to calculate. Generating them took about half the time that the 30 features of AffiScore required. Performance-wise, RF-Score based models lag slightly behind those fitted to AffiScore raw features (except in the case of RF) as shown in Table 3.9 above.

The second contributor to scoring throughput is prediction time. The generated features discussed here and in Section 2.2 are merely fed to a prediction machine to calculate the final binding affinity. This computation is instantaneous for almost all the machine-learning tools we investigate in this study. The third column of Table 3.10 shows the time spent in predicting the binding affinity of protein-ligand complexes in Cr_{XAR} test set by all methods. The only method that showed nonzero time in prediction is kNN. As discussed in Section 2.5.1, this can be attributed to the "lazy learning" that kNN performs during prediction. kNN's time (0.6 seconds) was for performing predictions for the 192 complexes in Cr_{XAR}, which would be almost zero if a per-complex time was recorded instead.

Unlike prediction time, the training time of some prediction schemes can be quite long. The second column in Table 3.10 lists the time elapsed (in seconds) to train each method on the primary training set Pr_{XAR} (1086 complexes and 72 features). Optimal parameters listed in

Table 3.1 were used to construct the machine-learning models. The numbers show that ensemble techniques are relatively slow to learn the unknown function. NF is the slowest; in fact, BRT is about 180 times faster than NF. This is understandable since each neural network of the 1000 grown in NF models requires hundreds of epochs for their weights to converge. This time, however, can be reduced significantly if the NF algorithm is slightly modified to run on a multi-processor system. Remember that each network in NF is completely independent from the rest of the committee. This parallelization is also applicable to RF models whose learning time is already relatively small. For the other techniques, we notice that MARS is not as rapid as SVM and MLR. It could be even slower if more exhaustive approaches were used during the addition/elimination of the model's terms in forward/backward passes. Again, kNN does not have the notion of training, so its training time is not reported in the table.

Table 3.10 Training and prediction times of different scoring techniques.

Method	**Training Time (in secs.) - Pr_{XAR}**	**Prediction Time (in secs.) - Cr_{XAR}**
NF	9000	0.0
BRT	48.7	0.0
RF	16.7	0.0
MARS	2.2	0.0
kNN	-	0.6
SVM	0.9	0.0
MLR	0.1	0.0

In summary, training a scoring function is not an issue in virtual screening since they are typically constructed offline. The online part on the other hand is mixed. We saw that the time spent in prediction is almost zero. Feature extraction, however, is the main factor in determining scoring speed. Therefore, we recommend using only features that can be rapidly calculated, such as those provided by SLIDE (AffiScore – raw) and RF-Score. It turns out that there is no big difference in performance between the scoring functions BRT::XAR (i.e., X-Score features

included) and BRT::AR (only AffiScore and RF-Score features considered). In terms of numbers (R_P), compare 0.797 for BRT::XAR to 0.795 achieved by BRT::AR.

CHAPTER 4

Concluding Remarks and Future Work

4.1 Thesis Summary

The aim of this thesis was to empirically compare machine-learning-based scoring functions to their conventional counterparts. Machine-learning schemes investigated were random forests (RF), neural forests (NF), boosted regression trees (BRT), support vector machines (SVM), k-Nearest neighbors (kNN), multivariate adaptive regression splines (MARS), and multiple linear regression (MLR). A diverse database of protein-ligand complexes was used. In addition, a diverse set of features were extracted to characterize each complex in this database. The machine-learning models were then trained on this data and their predictive ability were compared to 16 state-of-the-art scoring functions. We next discuss our main findings and some areas for future work.

4.2 Findings and Suggestions

We found that ensemble prediction methods in general outperform others in many performance metrics. If the task is to score or rank ligands for almost all proteins (except HIV protease), then ensemble-based scoring functions should be employed. When large numbers of features are available and need to be exploited, again ensemble models are most suitable. That is due to their excellent immunity to overfitting. If the collected features are not noisy and have adequate relevance to binding affinity, then kNN-based scoring functions can be considered as well. SVM could also perform well given its performance on other experiments when its parameters were carefully tuned.

Empirical- and MARS-based scoring functions as well as RF and kNN may be used when the target is a protein that is not present in the training dataset used to build the scoring model. In such scenarios, simple models can compete with sophisticated methods. This was obvious when we attempted to predict the binding affinity of ligands interacting with HIV protease, trypsin, carbonic anhydrase, thrombin (see Table 3.7). For novel targets with low binding site sequence similarity with the binding sites of proteins in training complexes, RF::XAR, kNN::XAR and MLR::X were found to be the best choices (see Table 3.8).

Ensemble techniques were found not only to be accurate prediction methods, but also excellent descriptive tools. Variable importance calculation and marginal plots generation are valuable visualization utilities. These features changed BRT, RF, and NF from being black box tools to methods that enable specialists to decipher the influence of different protein-ligand interactions on modeling the binding energies. Computationally, ensemble techniques were shown to be as fast as other predictive methods in calculating binding affinity. Training

ensemble-based scoring functions is, however, relatively slower than their counterparts based on single models.

Our findings agree with previous work in that there is no single scoring function that consistently outperforms others in all scoring aspects. However, we can conclude that BRT, RF, and NF are the best performers on the whole. Even in scenarios when these ensemble models do not take the lead, their performance statistics are not far from those of the top scoring functions.

4.3 Looking Ahead

Based on the experiments we conducted in this work, machine-learning based algorithms have been very effective in predicting protein-ligand binding affinity. Accuracy-wise, BRT, RF, and NF are the best performers on average. Their descriptive power is also valuable in virtual screening. Another powerful ensemble method that could be explored in the future is rule ensembles. It is known that rule-based models are very intuitive to interpret. In addition, a study conducted by Friedman and colleagues suggests that the rule ensembles model surpasses other competitive ensemble techniques in both classification and regression problems [109]. Their proposed model, RuleFit, is a set of rules extracted from an ensemble of trees. In addition to the high accuracy they reported for RuleFit and the inherent ease of rule interpretation, it also has the capability of automatic identification of important variables. Despite these features, applying RuleFit to our problem did not take the first spot from BRT, NF, and RF models. In terms of numbers, when the RuleFit model was trained on Pr_{XAR} dataset, it yielded a Pearson correlation coefficient of 0.75 on the Cr_{XAR} test set. Nevertheless, this was a preliminary experiment that did not involve any model improvements or thorough parameter tuning. In the future, we aim to enhance the model accuracy and present its powerful descriptive capabilities, as well as compare it to other competitive techniques.

According to results collected from the features size experiment in Section 3.6, we find that some machine-learning techniques, BRT and RF in particular, are very immune to overfitting. Moreover, their accuracy tends to improve by increasing the number of features relevant to binding affinity. This finding should encourage us to pursue the path of collecting as many important features as possible and utilizing them in building future scoring engines.

Collecting more training data is also shown to be important to enhance the performance of scoring functions. In the future, we will consider more than one source for protein-ligand complexes to expand our training and test datasets. In fact, there are many publicly available databases of protein-ligand complexes such as LPDB [110], PLD [111], Binding DB [112], PDBbind [31], [104], Binding MOAD [113], and AffinDB [114]. Complexes to be collected from these archives should undergo strict filtering criteria to insure only complexes with high quality structures and reliable measured binding affinities are used.

We pointed out earlier that BRT, NF and RF yield best performing scoring functions. Besides combining a set of trees to constitute one ensemble model, one could fuse different predictive techniques to form a consensus scoring function. Different machine-learning methodologies tend to generate diverse prediction surfaces for the same data patterns. The average of these surfaces may be closer to the surface of the unknown function. That may explain why prediction performance of combined models sometimes surpasses those of the model components when they are applied alone.

We conducted a preliminary experiment to explore the potential of consensus schemes. In this experiment, we averaged all possible combinations of RF, NF, BRT, SVM, kNN, MARS, and MLR predictions on Cr_{XAR} test set. The results are depicted in Figure 4.1, where the x-axis shows the number of combined models, while the vertical axis shows each combination's scoring power. The combinations are identified by the initials of the constituting components, except for MLR which is represented by the letter L. The figure shows only the models whose Pearson correlation coefficient exceeds 0.80. Note that BLNRS consensus scoring function (which is a combination of BRT, MLR, NF, RF, and SVM) outperforms the best of single models (BRT);

the difference is very modest though. It is clear that the best consensus scoring functions have always at least two ensemble models included. One area of future work regarding consensus scoring is to investigate the optimal techniques that could be used to combine several scoring functions. Here we merely take the simple average; more sophisticated schemes such as weighed aggregation should be probed.

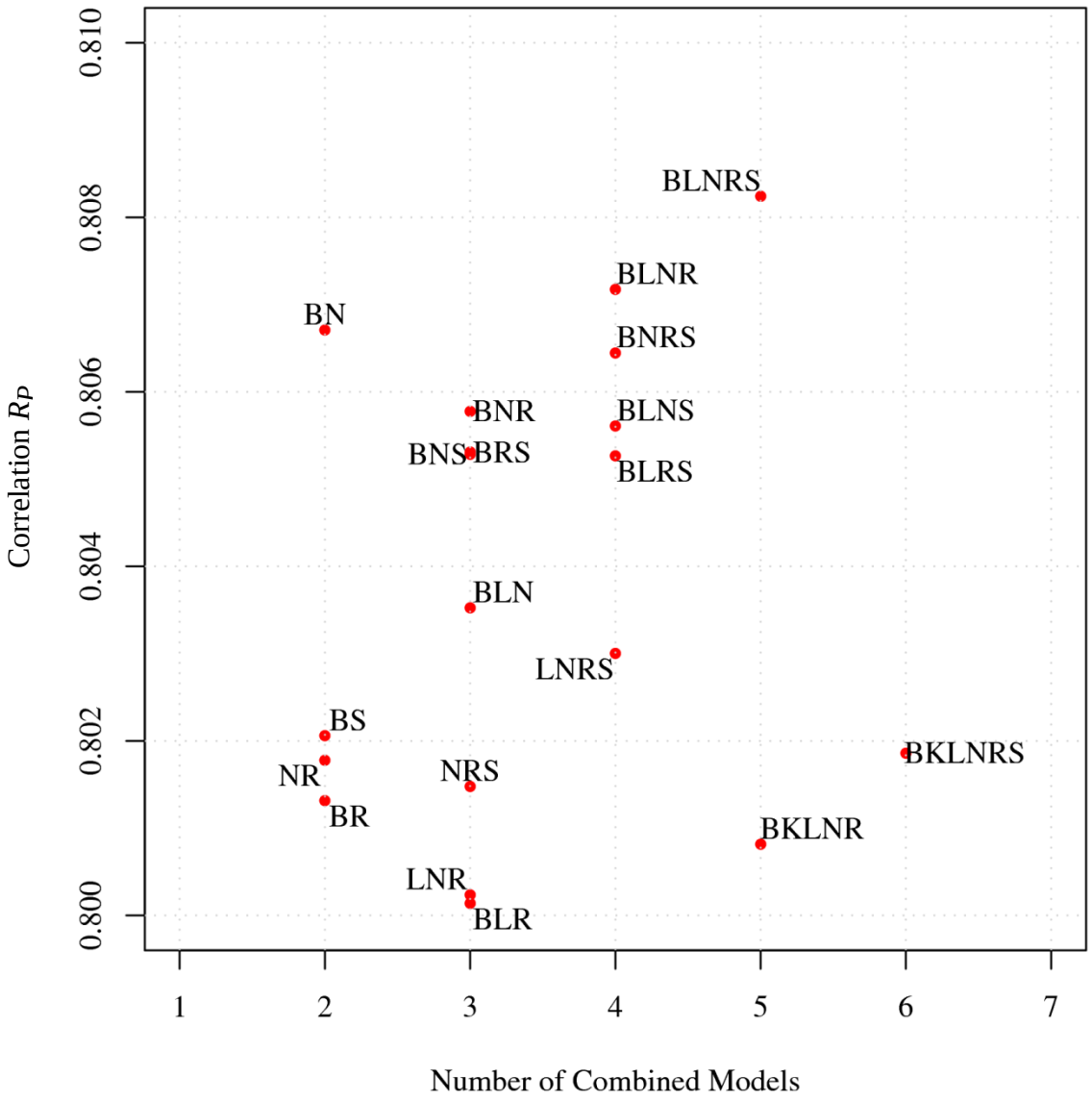


Figure 4.1 Performance of consensus scoring functions on the core test set *Cr*.

Finally, we showed in Section 3.4 that ranking power of scoring functions is inferior to the scoring power although both statistics are related. Therefore, one could put more emphasis on ranking power when scoring functions are to be designed in the future.

In conclusion, we believe that there is good room for further improvement in the performance of scoring functions. In addition to the high performance profile we already obtained using machine-learning techniques, we think that our enhancement plans outlined above could result in steadily superior scoring functions.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] P. D. Lyne, "Structure-based virtual screening: an overview," *Drug Discovery Today*, vol. 7, no. 20, pp. 1047-1055, 2002.

[2] M. P. Mathieu, *PARAXEL's Pharmaceutical R&D Statistical Sourcebook*. PAREXEL International Corp., 2001.

[3] L. Michielan and S. Moro, "Pharmaceutical Perspectives of Nonlinear QSAR Strategies," *Journal of chemical information and modeling*, vol. 50, no. 6, pp. 961-978, 2010.

[4] A. M. Davis, S. J. Teague, and G. J. Kleywegt, "Application and limitations of X-ray crystallographic data in structure-based ligand and drug design," *Angewandte Chemie International Edition*, vol. 42, no. 24, pp. 2718-2736, 2003.

[5] G. Pintacuda, M. John, X. Su, and G. Otting, "NMR Structure Determination of Protein−Ligand Complexes by Lanthanide Labeling," *Accounts of Chemical Research*, vol. 40, no. 3, pp. 206-212, 2007, PMID: 17370992.

[6] A. Evers and G. Klebe, "Successful Virtual Screening for a Submicromolar Antagonist of the Neurokinin-1 Receptor Based on a Ligand-Supported Homology Model," *Journal of Medicinal Chemistry*, vol. 47, no. 22, pp. 5381-5392, 2004.

[7] A. Evers and T. Klabunde, "Structure-based Drug Discovery Using GPCR Homology Modeling: Successful Virtual Screening for Antagonists of the Alpha1A Adrenergic Receptor," *Journal of Medicinal Chemistry*, vol. 48, no. 4, pp. 1088-1097, 2005, PMID: 15715476.

[8] F. Sousa, M. Cerqueira, A. Fernandes, and J. Ramos, "Virtual Screening in Drug Design and Development," *Combinatorial Chemistry & High Throughput Screening*, vol. 13, no. 5, pp. 442-453, 2010.

[9] W. Schaal, A. Karlsson, A. Göran, J. Lindberg, H. Andersson, U. Danielson, and B. Classon, T. Unge, B. Samuelsson, J. Hultén, A. Hallberg, and A. Karlén, "Synthesis and Comparative Molecular Field Analysis (CoMFA) of Symmetric and Nonsymmetric Cyclic Sulfamide HIV-1 Protease Inhibitors," *Journal of Medicinal Chemistry*, vol. 44, no. 2, pp. 155-169, 2001.

[10] T. J. Ewing, S. Makino, A. Skillman, and I. D. Kuntz, "DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases," *Journal of Computer-Aided*

Molecular Design, vol. 15, no. 5, pp. 411-428, 2001.

[11] M. I. Zavodszky, P. C. Sanschagrin, L. A. Kuhn, and R. S. Korde, "Distilling the essential features of a protein surface for improving protein-ligand docking, scoring, and virtual screening," *Journal of Computer-Aided Molecular Design*, vol. 16, pp. 883-902, 2002, 10.1023/A:1023866311551.

[12] R. A. Friesner, J. L. Banks, R. B. Murphy, T. A. Halgren, J. J. Klicic, D. T. Mainz, M. P. Repasky, E. H. Knoll, M. Shelley, J. K. Perry, D. E. Shaw, P. Francis, and P. S. Shenkin, "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy," *Journal of Medicinal Chemistry*, vol. 47, no. 7, pp. 1739-1749, 2004, PMID: 15027865.

[13] S. Y. Huang, S.Z. Grinter, and X. Zou, "Scoring functions and their evaluation methods for protein--ligand docking: recent advances and future directions," *Physical chemistry chemical physics: PCCP*, vol. 12, pp. 12899-12908, 2010.

[14] D. B. Kitchen, H. Decornez, J. R. Furr, and J. Bajorath, "Docking and scoring in virtual screening for drug discovery: methods and applications," *Nature reviews Drug discovery*, vol. 3, no. 11, pp. 935-949, 2004.

[15] I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, and T. E. Ferrin, "A geometric approach to macromolecule-ligand interactions* 1," *Journal of Molecular Biology*, vol. 161, no. 2, pp. 269-288, 1982.

[16] R. L. DesJarlais, R. P. Sheridan, G. L. Seibel, J. S. Dixon, I. D. Kuntz, and R. Venkataraghavan, "Using shape complementarity as an initial screen in designing ligands for a receptor binding site of known three-dimensional structure," *Journal of medicinal chemistry*, vol. 31, no. 4, pp. 722-729, 1988.

[17] D. A. Gschwend and I. D. Kuntz, "Orientational sampling and rigid-body minimization in molecular docking revisited: on-the-fly optimization and degeneracy removal," *Journal of computer-aided molecular design*, vol. 10, no. 2, pp. 123-132, 1996.

[18] D. S. Goodsell and A. J. Olson, "Automated docking of substrates to proteins by simulated annealing," *Proteins: Structure, Function, and Bioinformatics*, vol. 8, no. 3, pp. 195-202, 1990.

[19] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson, "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function," *Journal of Computational Chemistry*, vol. 19, no. 14, pp. 1639-1662, 1998.

[20] H. J. Böhm, "The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure," *Journal of Computer-Aided Molecular Design*, vol. 8, no. 3, pp. 243-256, 1994.

[21] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe, "A fast flexible docking method using an incremental construction algorithm," *Journal of Molecular Biology*, vol. 261, no. 3, pp. 470-489, 1996.

[22] P. D. Thomas and K. A. Dill, "An iterative method for extracting energy-like quantities from protein structures," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 21, p. 11628, 1996.

[23] W. Koppensteiner and M. Sippl, "Knowledge-based potentials--back to the roots.," *Biochemistry. Biokhimii□a*, vol. 63, no. 3, p. 247, 1998.

[24] I. Muegge, and Y. Martin, "A General and Fast Scoring Function for Protein−Ligand Interactions: A Simplified Potential Approach," *Journal of Medicinal Chemistry*, vol. 42, no. 5, pp. 791-804, 1999, PMID: 10072678.

[25] I. Muegge, "A knowledge-based scoring function for protein-ligand interactions: Probing the reference state," *Perspectives in Drug Discovery and Design*, vol. 20, pp. 99-114, 2000, 10.1023/A:1008729005958.

[26] H. Gohlke, M. Hendlich, and G. Klebe, "Knowledge-based scoring function to predict protein-ligand interactions," *Journal of Molecular Biology*, vol. 295, no. 2, pp. 337-356, 2000.

[27] I. Muegge, "Effect of ligand volume correction on PMF scoring," *Journal of Computational Chemistry*, vol. 22, no. 4, pp. 418-425, 2001.

[28] H. F. Velec , H. Gohlke, and G. Klebe, "DrugScoreCSD Knowledge-Based Scoring Function Derived from Small Molecule Crystal Data with Superior Recognition Rate of Near-Native Ligand Poses and Better Affinity Prediction," *Journal of Medicinal Chemistry*, vol. 48, no. 20, pp. 6296-6303, 2005.

[29] I. Muegge, "PMF Scoring Revisited," *Journal of Medicinal Chemistry*, vol. 49, no. 20, pp. 5895-5902, 2006, PMID: 17004705.

[30] P. Ballester and J. Mitchell, "A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking," *Bioinformatics*, vol. 26, no. 9, p. 1169, 2010.

[31] R. Wang, X. Fang, Y. Lu, and S. Wang, "The PDBbind Database: Collection of Binding Affinities for Protein−Ligand Complexes with Known Three-Dimensional Structures," *Journal of Medicinal Chemistry*, vol. 47, no. 12, pp. 2977-2980, 2004, PMID: 15163179.

[32] T. Cheng, X. Li, Y. Li, Z. Liu, and R. Wang, "Comparative assessment of scoring functions on a diverse test set," *Journal of chemical information and modeling*, vol. 49, no. 4, pp. 1079-1093, 2009.

[33] R. Wang, Y. Lu, X. Fang, S. Wang, "An Extensive Test of 14 Scoring Functions Using the PDBbind Refined Set of 800 Protein−Ligand Complexes," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 6, pp. 2114-2125, 2004, PMID: 15554682.

[34] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5-32, 2001, 10.1023/A:1010933404324.

[35] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: a classification and regression tool for compound classification and QSAR modeling," *J. Chem. Inf. Comput. Sci*, vol. 43, no. 6, pp. 1947-1958, 2003.

[36] S. Doniger, T. Hofmann, and J. Yeh, "Predicting CNS permeability of drug molecules: comparison of neural network and support vector machine algorithms," *Journal of computational biology*, vol. 9, no. 6, pp. 849-864, 2002.

[37] W. Tong, H. Hong, H. Fang, Q. Xie, and R. Perkins, "Decision Forest: Combining the Predictions of Multiple Independent Decision Tree Models," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 2, pp. 525-531, 2003, PMID: 12653517.

[38] J. E. Penzotti, M. L. Lamb, L. Michelle, E. Evensen, and J. D. Peter, "A Computational Ensemble Pharmacophore Model for Identifying Substrates of P-Glycoprotein," *Journal of Medicinal Chemistry*, vol. 45, no. 9, pp. 1737-1740, 2002.

[39] G. A. Bakken, and P. C. Jurs, "Classification of Multidrug-Resistance Reversal Agents Using Structure-Based Descriptors and Linear Discriminant Analysis," *Journal of Medicinal Chemistry*, vol. 43, no. 23, pp. 4534-4541, 2000.

[40] P. J. Gilligan, G. A. Cain, T. E. Christos, L. Cook, S. Drummond, A. L. Johnson, A. A. Kergaye, J. F. McElroy, and K. W. Rohrbach, "Novel piperidine .sigma. receptor ligands as potential antipsychotic drugs," *Journal of Medicinal Chemistry*, vol. 35, no. 23, pp. 4344-4361, 1992.

[41] G. W. Kauffman and P. C. Jurs, "QSAR and k-Nearest Neighbor Classification Analysis of Selective Cyclooxygenase-2 Inhibitors Using Topologically-Based Numerical Descriptors," *Journal of Chemical Information and Computer Sciences*, vol. 41, no. 6, pp. 1553-1560, 2001.

[42] C. H. Andrade, K. F. Pasqualoto, E. I. Ferreira, A. J. Hopfinger, "4D-QSAR: Perspectives in Drug Design," *Molecules*, vol. 15, no. 5, pp. 3281-3294, 2010.

[43] C. Andersson, B. Chen, and A. Linusson, "Multivariate assessment of virtual screening experiments," *Journal of Chemometrics*, vol. 24, no. 11-12, pp. 757-767, 2010.

[44] D. Plewczynski, S. Spieser, and U. Koch, "Assessing different classification methods for virtual screening," *J. Chem. Inf. Model*, vol. 46, no. 3, pp. 1098-1106, 2006.

[45] MDL Information Systems. MACCS Drug Data Report (MDDR). Database, 2006.

[46] M. Fatemi and S. Gharaghani, "A novel QSAR model for prediction of apoptosis-inducing activity of 4-aryl-4-H-chromenes based on support vector machine," *Bioorganic & Medicinal Chemistry*, vol. 15, no. 24, pp. 7746-7754, 2007.

[47] M. Goodarzi, M. Freitas, and R. Jensen, "Feature Selection and Linear\Nonlinear Regression Methods for the Accurate Prediction of Glycogen Synthase Kinase-3β Inhibitory Activities," *Journal of Chemical Information and Modeling*, vol. 49, no. 4, pp. 824-832, 2009.

[48] H. Tang, X. S. Wang, X. Huang, B. L. Roth, K. V. Butler, A. P. Kozikowski, M. Jung, and A. Tropsha, "Novel Inhibitors of Human Histone Deacetylase (HDAC) Identified by QSAR Modeling of Known Inhibitors, Virtual Screening, and Experimental Validation," *Journal of Chemical Information and Modeling*, vol. 49, no. 2, pp. 461-476, 2009.

[49] H. Liu, E. Papa, J. Walker, and P. Gramatica, "In silico screening of estrogen-like chemicals based on different nonlinear classification models," *Journal of Molecular Graphics and Modelling*, vol. 26, no. 1, pp. 135-144, 2007.

[50] H. Lin, L. Han, C. Yap, Y. Xue, X. Liu, F. Zhu, and Y. Chen, "Prediction of factor Xa inhibitors by machine learning methods," *Journal of Molecular Graphics and Modelling*, vol. 26, no. 2, pp. 505-518, 2007.

[51] D. S. Chekmarev, V. Kholodovych, K. V. Balakin, Y. Ivanenkov, S. Ekins, and W. J. Welsh, "Shape Signatures: New Descriptors for Predicting Cardiotoxicity In Silico," *Chemical Research in Toxicology*, vol. 21, no. 6, pp. 1304-1314, 2008, PMID: 18461975.

[52] R. Hu, J. Doucet, M. Delamar, and R. Zhang, "QSAR models for 2-amino-6-arylsulfonylbenzonitriles and congeners HIV-1 reverse transcriptase inhibitors based on linear and nonlinear regression methods," *European Journal of Medicinal Chemistry*, vol. 44, no. 5, pp. 2158-2171, 2009.

[53] N. Hernández, R. Kiralj, M. Ferreira, and I. Talavera, "Critical comparative analysis, validation and interpretation of SVM and PLS regression models in a QSAR study on HIV-1 protease inhibitors," *Chemometrics and Intelligent Laboratory Systems*, vol. 98, no. 1, pp. 65-77, 2009.

[54] M. Goodarzi, P. Duchowicz, C. Wu, F. Fernández, and E. Castro, "New Hybrid Genetic Based Support Vector Regression as QSAR Approach for Analyzing Flavonoids-GABA(A) Complexes," *Journal of Chemical Information and Modeling*, vol. 49, no. 6, pp. 1475-1485, 2009, PMID: 19492793.

[55] X. Yang, D. Chen, M. Wang, Y. Xue, and Y. Chen, "Prediction of antibacterial compounds by machine learning approaches," *Journal of Computational Chemistry*, vol. 30, no. 8, pp.

1202-1211, 2009.

[56] Y. Cong, X. Yang, W. Lv, and Y. Xue, "Prediction of novel and selective TNF-alpha converting enzyme (TACE) inhibitors and characterization of correlative molecular descriptors by machine learning approaches," *Journal of Molecular Graphics and Modelling*, vol. 28, no. 3, pp. 236-244, 2009.

[57] R. Wang, L. Lai, and S. Wang, "Further development and validation of empirical scoring functions for structure-based binding affinity prediction," *Journal of Computer-Aided Molecular Design*, vol. 16, pp. 11-26, 2002, 10.1023/A:1016357811882.

[58] V. Schnecke and L. A. Kuhn, "Virtual screening with solvation and ligand-induced complementarity," in *Virtual Screening: An Alternative or Complement to High Throughput Screening?*, G. Klebe, Ed. Springer Netherlands, 2002, pp. 171-190, 10.1007/0-306-46883-2_10.

[59] M. Zavodszky and L. Kuhn, "Side-chain flexibility in protein–ligand binding: The minimal rotation hypothesis," *Protein Science*, vol. 14, no. 4, pp. 1104-1114, 2005.

[60] M. Tonero, M. Zavodszky, J. Van Voorst, L. He, S. Arora, S. Namilikonda, and L. Kuhn, "Effective Scoring Functions for Predicting Ligand Binding Mode and Affinity in Docking and High-Throughput Screening"," *in preparation*.

[61] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The Protein Data Bank," *Nucleic Acids Research*, vol. 28, no. 1, pp. 235-242, 2000.

[62] K. Raha, and K. Merz, "Large-Scale Validation of a Quantum Mechanics Based Scoring Function: Predicting the Binding Affinity and the Binding Mode of a Diverse Set of Protein−Ligand Complexes," *Journal of Medicinal Chemistry*, vol. 48, no. 14, pp. 4558-4575, 2005, PMID: 15999994.

[63] S. Huang and X. Zou, "An iterative knowledge-based scoring function to predict protein–ligand interactions: II. Validation of the scoring function," *Journal of Computational Chemistry*, vol. 27, no. 15, pp. 1876-1882, 2006.

[64] A. Ruvinsky, "Calculations of protein-ligand binding entropy of relative and overall molecular motions," *Journal of Computer-Aided Molecular Design*, vol. 21, pp. 361-370, 2007, 10.1007/s10822-007-9116-0.

[65] A. Ruvinsky, "Role of binding entropy in the refinement of protein–ligand docking predictions: Analysis based on the use of 11 scoring functions," *Journal of Computational Chemistry*, vol. 28, no. 8, pp. 1364-1372, 2007.

[66] J. Lee, and C. Seok, "A statistical rescoring scheme for protein–ligand docking: Consideration of entropic effect," *Proteins: Structure, Function, and Bioinformatics*, vol.

70, no. 3, pp. 1074-1083, 2008.

[67] Tripos Inc. The Sybyl Software. Software, 2006.

[68] T. Madden, "The BLAST sequence analysis tool," *The NCBI Handbook [Internet]*. *National Library of Medicine (US), National Center for Biotechnology Information, Bethesda, MD*, 2002.

[69] Y. Cheng and W. Prusoff, "Relationship between the inhibition constant (Ki) and the concentration of inhibitor which causes 50% inhibition (IC50) of an enzymatic reaction," *Biochem. Pharmacol*, vol. 22, no. 23, pp. 3099-3108, 1973.

[70] R. A. Copeland, D. Lombardo, J. Giannaras, and C. P. Decicco, "Estimating KI values for tight binding inhibitors from dose-response plots," *Bioorganic & Medicinal Chemistry Letters*, vol. 5, no. 17, pp. 1947-1952, 1995.

[71] J. H. Friedman, "Multivariate Adaptive Regression Splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1-67, 1991.

[72] R Core Development Team. R: A Language and Environment for Statistical Computing. Software, 2010.

[73] S. Milborrow and R. Tibshirani, "earth: Multivariate Adaptive Regression Spline Models," 2010.

[74] K. Schliep, "kknn: Weighted k-Nearest Neighbors," 2010.

[75] K. Hechenbichler and K. Schliep, "Weighted k-nearest-neighbor techniques and ordinal classification," Citeseer, 2004.

[76] V. Vapnik, *Statistical learning theory*. Wiley, New York, 1998.

[77] X. J. Yao, A. Panaye, J. P. Doucet, R. S. Zhang, H. F. Chen, M. Liu, Z. D. Hu, and B. T. Fan, "Comparative Study of QSAR/QSPR Correlations Using Support Vector Machines, Radial Basis Function Neural Networks, and Multiple Linear Regression," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 4, pp. 1257-1266, 2004.

[78] C. C. Chang and C. J. Lin, "LIBSVM: a library for support vector machines," 2001.

[79] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel, "e1071: Misc Functions of the Department of Statistics (e1071), TU Wien," 2010.

[80] L. K. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993-1001, 1990.

[81] Y. Amit and D. Geman, "Shape Quantization and Recognition with Randomized Trees," *Neural Computation*, vol. 9, no. 7, pp. 1545-1588, 1997.

[82] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, no. 2, pp. 139-157, 2000.

[83] M. R. Segal, "Machine learning benchmarks and random forest regression," 2004.

[84] D. Plewczynski, M. Grotthuss, L. Rychlewski, and K. Ginalski, "Virtual High Throughput Screening Using Combined Random Forest and Flexible Docking," *Combinatorial Chemistry & High Throughput Screening*, vol. 12, no. 5, pp. 484-489, 2009, Not Available Yet.

[85] A. Liaw and M. Wiener, "Classification and Regression by randomForest," *R News*, vol. 2, no. 3, pp. 18-22, 2002.

[86] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge Univ Pre, 2005.

[87] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*. Springer, 2002.

[88] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367-378, 2002.

[89] G. Ridgeway, "gbm: Generalized Boosted Regression Models," 2010.

[90] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, pp. 1189-1232, 2001.

[91] Accelrys Software Inc. The Discovery Studio Software. Software, 2001.

[92] A. Krammer, P. Kirchhoff, X. Jiang, C. Venkatachalam, and M. Waldman, "LigScore: a novel scoring function for predicting binding affinities," *Journal of Molecular Graphics and Modelling*, vol. 23, no. 5, pp. 395-407, 2005.

[93] D. K. Gehlhaar, G. M. Verkhivker, P. A. Rejto, C. J. Sherman, D. R. Fogel, L. J. Fogel, and S. T. Freer, "Molecular recognition of the inhibitor AG-1343 by HIV-1 protease: conformationally flexible docking by evolutionary programming," *Chemistry & Biology*,

vol. 2, no. 5, pp. 317-324, 1995.

[94] A. L. Parrill and M. R. Reddy, *Rational Drug Design: novel methodology and practical applications*. An American Chemical Society Publication, 1999.

[95] A. N. Jain, "Scoring noncovalent protein-ligand interactions: A continuous differentiable function tuned to compute binding affinities," *Journal of Computer-Aided Molecular Design*, vol. 10, pp. 427-440, 1996, 10.1007/BF00124474.

[96] H. Böhm, "Prediction of binding constants of protein ligands: A fast method for the prioritization of hits obtained from de novo design or 3D database search programs," *Journal of Computer-Aided Molecular Design*, vol. 12, pp. 309-309, 1998, 10.1023/A:1007999920146.

[97] G. Jones, P. Willett, and R. C. Glen, "Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation," *Journal of molecular biology*, vol. 245, no. 1, pp. 43-53, 1995.

[98] G. Jones, P. Willett, R. Glen, A. Leach, and R. Taylor, "Development and validation of a genetic algorithm for flexible docking," *Journal of Molecular Biology*, vol. 267, no. 3, pp. 727-748, 1997.

[99] M. D. Eldridge, C. W. Murray, T. R. Auton, G. V. Paolini, and P. P. Mee, "Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes," *Journal of Computer-Aided Molecular Design*, vol. 11, pp. 425-445, 1997, 10.1023/A:1007996124545.

[100] C. A. Baxter, C. W. Murray, D. E. Clark, D. R. Westhead, and M. D. Eldridge, "Flexible docking using tabu search and an empirical estimate of binding affinity," *Proteins: Structure, Function, and Bioinformatics*, vol. 33, no. 3, pp. 367-382, 1998.

[101] W. Mooij, and M. Verdonk, "General and targeted statistical potentials for protein-ligand interactions.," *Proteins*, vol. 61, no. 2, p. 272, 2005.

[102] R. Friesner, R. Murphy, M. Repasky, L. Frye, J. Greenwood, T. Halgren, P. Sanschagrin, and D. Mainz, "Extra precision glide: docking and scoring incorporating a model of hydrophobic enclosure for protein- ligand complexes," *J. Med. Chem*, vol. 49, no. 21, pp. 6177-6196, 2006.

[103] Schrodinger, LLC. The Schrodinger Software. Software, 2005.

[104] R. Wang, X. Fang, Y. Lu, C. Yang, and S. Wang, "The PDBbind Database: Methodologies and Updates," *Journal of Medicinal Chemistry*, vol. 48, no. 12, pp. 4111-4119, 2005, PMID: 15943484.

[105] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Citeseer*, vol. 14, 1995, pp. 1137-1145.

[106] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. 2001.

[107] C. J. Lin, C. Chang, and C. Hsu, "A practical guide to support vector classification," *National Taiwan University*, 2004.

[108] J. Overington, B. Al-Lazikani, and A. Hopkins, "How many drug targets are there?," *Nature reviews Drug discovery*, vol. 5, no. 12, pp. 993--996, 2006.

[109] J. H. Friedman and B. E. Popescu, "Predictive learning via rule ensembles," *Annals*, vol. 2, no. 3, pp. 916-954, 2008.

[110] O. Roche, R. Kiyama, and C. L. Brooks, "Ligand−Protein DataBase: Linking Protein−Ligand Complex Structures to Binding Data," *Journal of Medicinal Chemistry*, vol. 44, no. 22, pp. 3592-3598, 2001, PMID: 11606123.

[111] D. Puvanendrampillai and B. John, "Protein Ligand Database (PLD): additional understanding of the nature and specificity of protein–ligand complexes," *Bioinformatics*, vol. 19, no. 14, pp. 1856-1857, 2003.

[112] T. Liu, Y. Lin, X. Wen, R. Jorissen, M. Gilson, and K. Michael , "BindingDB: a web-accessible database of experimentally determined protein–ligand binding affinities," *Nucleic Acids Research*, vol. 35, no. suppl 1, pp. D198-D201, 2006.

[113] M. L. Benson, R. D. Smith, N. A. Khazanov, B. Dimcheff, J. Beaver, P. Dresslar, J. Nerothin, and H. Carlson, "Binding MOAD, a high-quality protein–ligand database," *Nucleic Acids Research*, vol. 36, no. suppl 1, pp. D674-D678, 2008.

[114] P. Block, C. Sotriffer, I. Dramburg, and G. Klebe, "AffinDB: a freely accessible database of affinities for protein–ligand complexes from the PDB," *Nucleic Acids Research*, vol. 34, no. suppl 1, pp. D522-D526, 2006.