universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Design and Implementation of a Machine Learning Workflow for the LigandScout API for the Analysis of Molecular Datasets with Pharmacophores“

verfasst von / submitted by

Daniel Rose, B.Sc. M.Sc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Thierry Langer

Acknowledgement

First and foremost, I want to thank my supervisor Univ.-Prof. Mag. Dr. Thierry Langer for welcoming me in his group and giving me the opportunity to collaborate with Inte:Ligand on this very exciting project. I also want to thank Assoc.-Prof. Dr. Johannes Kirchmair for the co-supervision of this thesis.

A huge thank you goes to Christian Permann, MSc for the truly amazing guidance throughout the project. This work is the product of lengthy discussions, detailed code-reviews, and invaluable advice, may it be on Java Programming, the LigandScout codebase, or computer science topics in general. Thank you for discussing the structure of the thesis, and the detailed proof-reading. I further want to thank Dipl.-Ing. Gökhan Ibis for all the code reviews and the feedback on the final API design.

Thanks to Dipl.-Ing. Dr. Oliver Wieder, Dipl.-Ing. Dr. Thomas Seidel, and Nedra Mekni, MSc for helpful tips, insightful discussions, and sharing your knowledge in the field of computational drug discovery with me. I want to thank the whole group for the warm welcome, the pleasant atmosphere, and for having a wonderful time during my master's thesis.

Last but not least, I want to thank my family and Roxane, for your support, your help, and your trust. You are the gradient descent to my optimization problem. This thesis could not have been written without you. Thank you a lot!

Contents

List of Figures

List of Tables

1 Introduction

1.1 Computer-Aided Drug Design

Computer-aided drug design (CADD) refers to the utilization of computer-assisted methods for the discovery, design, and optimization of compounds with potential biological activity as drugs. Among other things, CADD techniques are employed for the screening of large compound databases to identify subsets of potentially active compounds for experimental testing. They further assist in the optimization of lead compounds by predicting the potential increase in binding affinity induced by structural changes or estimating pharmacokinetic properties such as absorption, distribution, metabolism, excretion, and toxicity (ADMET). Moreover, CADD can also be utilized in the *de novo* design of bioactive compounds.[1,2]

CADD can be broadly categorized into two main classes, structure-based and ligand-based methods. Structure-based methods utilize the three-dimensional structure of a protein to compute, *e.g.*, binding free energies *via* statistical mechanics and molecular dynamics simulations. In contrast, ligand-based methods are typically employed in instances where no structural information of the target is available. They involve the knowledge of known active and inactive compounds to perform tasks such as chemical similarity searches or constructing a predictive quantitative structure-activity relationship (QSAR) model.[2]

Computational methods are often intended to supplement experimental techniques. For example, high-throughput screening (HTS) is an automated experimental workflow in which large collections of molecules are screened for those that exhibit a desired biological response. The hit rate for HTS is usually low, but can be increased by the use of computational methods. Its digital analogue, virtual high-throughput screening (vHTS), is an approach in which virtual compound libraries are queried to limit the number of compounds to be investigated in the HTS by prior assessment of their suitability in a computational model. Diminishing the quantity of compounds that require screening, while maintaining a comparable level of lead compounds reduces both the expenses and the workload associated with HTS.[2]

vHTS approaches encompass a variety of techniques, including structure-based docking of compounds to the protein target binding site, ligand-based similarity searches with structural fingerprints, or activity predictions with QSAR models. Pharmacophore mapping can be used in both ligand-based and structure-based contexts and will be discussed in more detail in Section

1.2. These techniques usually generate a ranking of the hit molecules in the virtual library with respect to a specific property. It should be noted that vHTS does not endeavor to identify compounds that are ready for clinical testing. Rather, it aims to uncover leads with chemotypes that have not yet been sufficiently investigated in the context of the target of interest. The use of computational methods therefore facilitates the coverage of a broader chemical space.[2]

The use of big data has become increasingly important in the recent years and has certainly not bypassed drug discovery. A vast amount of data, *e.g.*, originating from HTS, is stored in publicly accessible databases such as ChEMBL[3] and PubChem.[4] Machine learning (ML) algorithms can be applied to discover hidden patterns and non-linear relationships in large data sets, equipping the cheminformatician with a variety of novel tools. They have already been successfully used in QSAR modeling to evaluate binding affinity[5] or physicochemical properties such as lipophilicity, solubility, and acidity.[6,7] The prediction of the toxicity[8] and possible metabolites of molecules[9] also pose further challenges. Another exciting field is the *de novo* design of molecules, which can be tackled with generative neural networks. Moreover, molecular docking and virtual screening can also benefit from novel, machine-learned scoring functions.[10]

A widely applied approach is to represent molecules by fingerprints, serving as input for ML models such as Support Vector Machine (SVMs), Random Forest (RFs), or k-Nearest Neighbors (kNN) classifiers. Much research has been invested in the design of fingerprints that depict relevant chemical features and have a high information content. Recent advances utilize graph neural networks (GNNs) to train models directly on the molecular graph.[11] There are some difficulties associated with ML that should always be kept in mind. As their reliability is entirely based on the available data, careful creation of data sets and high data quality are crucial. Moreover, model interpretability is a part of on-going research, since the black-box character of most algorithms makes it hard to gain deeper insights into the studied problem.[10,12]

The current pace of advancements in ML for CADD is rapid, but there is also a significant potential for novel applications in this field. The interested reader is referred to the comprehensive review by Yang *et al.*,[12] which describes the concepts of artificial intelligence (AI) in CADD in great detail.

1.2 LigandScout

The IUPAC defines a pharmacophore as "the ensemble of steric and electronic features that is necessary to ensure the optimal supramolecular interactions with a specific biological target structure and to trigger (or to block) its biological response". It does not describe the underlying molecule itself, but rather provides an abstract description of the interaction, which usually includes H-bonding, hydrophobic, and electrostatic interaction sites. Pharmacophores can therefore be considered as the largest common denominator shared by a set of active molecules.[1]

LigandScout[13] by inte:ligand[14] is an established CADD software that provides a comprehensive toolbox for 3D pharmacophore modeling. It can be used to develop 3D interaction

feature models from the structure of a protein-ligand complex, a set of known ligands that bind to the same active site, or an active site where no ligands are present. These models can be applied in virtual screening experiments to identify new hit compounds, prioritize them for further synthesis and testing, as well as support lead optimization projects using 3D pharmacophore-based alignment experiments. LigandScout offers additional functionalities including docking functions, advanced filtering tools, fragment-based approaches, and the possibility to display molecular dynamics (MD) trajectories.[15]

LigandScout works per default with a basic pharmacophore feature set that encompasses hydrogen-bond acceptors (HBA), hydrogen-bond donors (HBD), aromatic moieties (AR), positive-ionizable groups (PI), negative-ionizable groups (NI), and hydrophobic features (H), although the use of other features is also possible. This default feature set, which is presented in Figure 1.1, was chosen with respect to a trade-off between generality and selectivity of the resulting pharmacophore representation.[15]

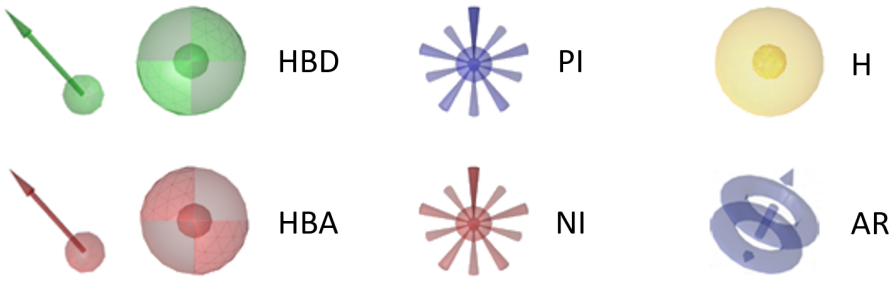

Figure 1.1: The default feature types in LigandScout and their graphical representation, *i.e.* hydrogen-bond acceptors (HBA), hydrogen-bond donors (HBD), aromatic moieties (AR), positive-ionizable groups (PI), negative-ionizable groups (NI), and hydrophobic features (H).[15]

It is possible to add metal-binding locations, or exclusion volumes for the representation of spatial constraints of the binding side. 3D pharmacophores are the spatial arrangement of a set of these features in the three dimensional Euclidean space, directed features are complemented by a 3D vector. A well-designed pharmacophore model captures the essence of the protein-ligand interaction, and screening a compound library with such pharmacophore as a query should therefore retrieve compounds exhibiting a similar feature pattern. Since pharmacophores do not rely on the underlying molecular structure, the approach enables scaffold hopping, which refers to the retrieval of a set of compounds with diverse molecular core-structures from the same query pharmacophore. The simplicity of the pharmacophore representation allows comparably fast screening of virtual screening databases, while additionally offering an easy interpretation of the obtained results. It is important to note that feature definition and placement are rule-based and handled differently among 3D pharmacophore modeling softwares, which influences the results of the retrieved hit list.[15–17]

The general virtual screening workflow in LigandScout is as follows: First, a pharmacophore model is created. This could be either structure-based from an available 3D structure of a

ligand-target complex, or ligand-based from a set of compounds that are known to bind to the same receptor site. An example for a structure-based pharmacophore is given in Figure 1.2, a ligand-based, merged pharmacophore is presented in Figure 1.3.

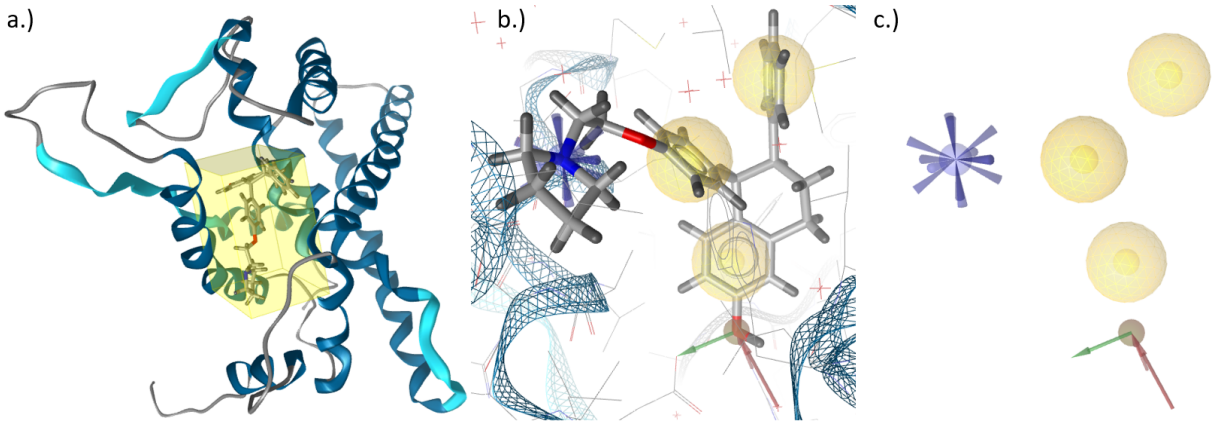

Figure 1.2: Generation of a structure-based pharmacophore from the PDB-entry 2OUZ. (**a.**) The crystal structure of the estrogen receptor α-lasofoxifene complex was deposited by Vajdos *et al.*[18] The protein is the target of the ESR1 dataset (Table 4.1) of the LIT-PCBA collection,[19] which was used for the benchmarking study in Section 4.1. (**b.**) Lasofoxifene in the binding-site, together with the structure-based pharmacophore, which was created with default settings in LigandScout 4. (**c.**) The structure-based pharmacophore.

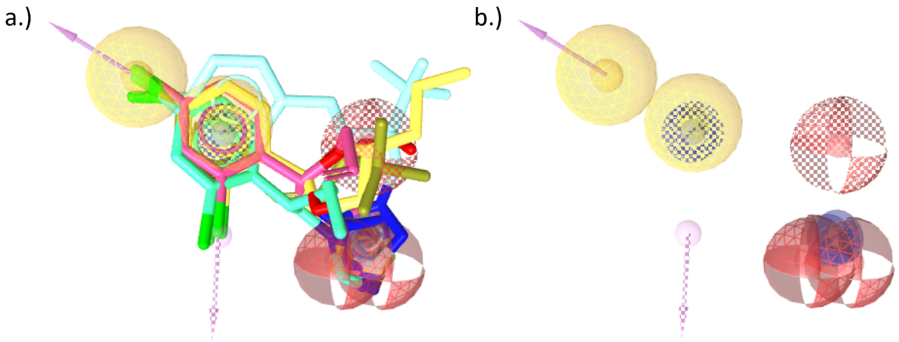

Figure 1.3: Generation of a ligand-based pharmacophore from the set of active compounds in the ESR1 dataset (Table 4.1) of the LIT-PCBA collection.[19] (**a.**) Active compounds were clustered with default settings in LigandScout 4. (**b.**) Ligand-based, merged pharmacophore.

There is always the possibility to customize the model manually. This model will be subjected to user-defined validation sets of active and inactive compounds. Common metrics for the evaluation of the screening are the enrichment factor, sensitivity, or the ROC-AUC metric. The user refines the model until they are satisfied with the validation results. The resulting pharmacophore hypothesis can then be used to filter a virtual library for screening hits. A screening database contains pre-generated conformations for each compound to account for the molecular flexibility. The pharmacophore representations of these conformations are aligned to the query and ranked according to a scoring function. Several pre-filtering steps, including pharmacophore feature counts and bloom filters, before the computationally demanding alignment-step reduce the overall

complexity of the screening. The obtained hit list can then be further prioritized for experimental assessment.[15–17,20]

There are similarities between the described virtual screening workflow and a ML pipeline. For example, a binary classifier could be similarly trained on data with labels indicating (in)activity towards a target protein. Since the pharmacophore model does describe the interaction between ligand and target, a ML model trained on pharmacophore input should be able to learn a correlation between the pharmacophore representation and the activity of a molecule that would match this pharmacophore. The validated model could then be used for further refinement of a hit list of a virtual screening experiment.

1.3 Aim of the Thesis

CADD tools are essential in assisting medicinal chemists in the drug discovery process. ML methods have gained popularity in recent years, and embedding ML in a CADD tool would provide practitioners with these expert methods for the analysis of chemical data. A lot of the currently reported ML models for biological activity prediction are trained on descriptors that are derived from the molecular structure, while pharmacophores are used less frequently for this purpose.

Pharmacophores are designed to describe the key properties of the protein-ligand interaction with only a small set of features, which can be interpreted as dimensionality reduction by feature selection from a ML point of view. If correctly applied, dimensionality reduction can help to reduce the amount of data needed for the creation of robust models, and improve generalization on unseen data.

The aim of this thesis is to design and implement a user-friendly ML pipeline that exploits the pharmacophore model for feature engineering. The pipeline will be integrated into the LigandScout software, which has an easy-to-use graphical user interface (GUI). The software also provides tested algorithms to extract pharmacophores from molecular data, making it an ideal platform for embedding the ML pipeline.

The package should be written in Java to maintain the cross-platform compatibility of the application. Since the application is to be kept lightweight, it is tried to avoid dependencies on external libraries. The utility of the pipeline shall be demonstrated by using features such as the LigandScout pharmacophores, the iConfGen[21] conformer generator, and the pharmacophore RDF spectrum.

2 Related Work

2.1 The Machine Learning Workflow

Nowadays, data scientists are confronted with a vast ecosystem of tools to choose from, beginning with the choice of the programming language. Python[22] is widely used in the field of data science due to its ease of use, large and active community, and extensive range of specialized libraries and tools for data analysis and machine learning. Popular Python libraries for machine learning include scikit-learn,[23] Tensorflow,[24] and PyTorch.[25] Java,[26] on the other hand, is a compiled language that offers high performance and scalability, making it a popular choice for developing complex systems that require robustness and security. While it may not be as commonly used in data science as Python, Java does have its advantages for large-scale applications. Java is inherently object-oriented, which makes it easy to create modular programs and reusable code. Because it is statically typed, it offers much more type safety than a dynamically typed language like Python. Its *"write once, run everywhere"* philosophy allows for high portability and platform-independent execution of code as long as a Java Virtual Machine (JVM) is installed. Data science libraries that are written in Java are, *e.g.*, Weka,[27] Elki,[28] and SparkML.[29]

The underlying workflow is essentially the same for most of the aforementioned libraries, a thorough analysis of the respective architectural design decisions is given by Zdun *et al.*[30] Typically, this process involves a series of steps, including:

- Data Collection: Data needs to be gathered from various sources.
- Data Preprocessing: This includes data cleaning, feature extraction, and feature selection. Data cleaning comprises removing or correcting any missing, duplicated, or erroneous data. Feature extraction is about identifying important features from the data, while feature selection involves selecting the most relevant features to improve model performance. This step often requires the transformation of data objects into a numerical representation.
- Model Selection: There are many machine learning algorithms to choose from. This step implies deciding on an algorithm that suits the problem at hand, and tuning its hyperparameters for optimal performance.
- Model Training: The selected model is trained with the preprocessed data to learn the patterns and relationships that may exist.

- Model Evaluation: The performance of the trained model needs to be assessed on a separate test dataset that was not used during the training phase. This is done to avoid overfitting, where the model performs well on the training data, but poorly on new, unseen data.
- Model Deployment: Once a satisfactory level of performance is achieved, the model can be deployed to make predictions on new, unseen data.

Synthesis of chemical compounds and the determination of their properties requires a lot of time, resources, expert-knowledge, and specialized instruments. Chemical datasets are therefore expensive to generate and comparably smaller than datasets in other areas of research. Furthermore, bioactivity data is generally highly imbalanced, simply because there exist much more inactive compounds than active ones, but it may also happen that only few actives and none of the inactive compounds were reported.

Molecules can vary in shape and size, however, most machine learning algorithms require a fixed dimensional input representation. There is a wide range of descriptors that can be used for feature engineering of molecules and picking the right representation can be challenging. Importantly, one must keep in mind that data splitting of molecular datasets is not as straightforward as for most other machine learning disciplines. Training and validation set should represent a similar structural diversity, which can not be guaranteed by simple randomized splitting.[31] The same arguments hold for the generation and representation of pharmacophore models, since they are directly derived from molecular data.

The upcoming section is meant to give a broad overview over relevant publicly available molecular benchmark datasets. Several use-cases of pharmacophores as machine learning input will be explored and the affiliated feature engineering techniques will be discussed in greater detail.

2.2 Open-Source Bioactivity Data

2.2.1 Molecular Databases

Training a machine learning model requires data, and there are a large number of open-source chemistry databases to choose from. In the following, a few popular database options will be introduced. All reported database sizes are as of April 2023.

PubChem is the largest, with 114 million compounds; it contains mainly small molecules organized into several data collections, with the PubChem BioAssay collection being particularly noteworthy in the context of bioactivity prediction. It contains 1.5 million test results from biological experiments along with a description provided by the submitter, totaling 300 million data points on biological activities.[4,32]

The ZINC archive is an open-access database of commercially available compounds suitable for virtual screening in drug discovery. The majority of compounds in the archive are referred to as "make-on-demand", which allows for the synthesis of novel molecules that have not been previously investigated, but can be readily accessed through contemporary synthetic methods.

ZINC offers multiple subsets of compounds, providing a means to narrow down the chemical space being searched. According to the ZINC homepage, it contains 230 million purchasable compounds in 3D formats, and over 750 million that could be searched for analogs in less than a minute.[33–35]

The Protein Data Bank (PDB) is described as a global repository for experimentally determined 3D structures of biological macromolecules and their complexes. It was established in 1971 and currently contains approximately 200,000 entries. It stores experimental data, associated metadata, and 3D atomic level structural models from crystallography, nuclear magnetic resonance spectroscopy, and electron microscopy.[36,37] The PDBbind database is a collection of experimentally measured binding affinity data for a subset of the protein-ligand complexes available in the PDB. It thus provides a link between the structural information of biomolecular complexes and their energetic properties.[38]

ChEMBL is a bioactivity database that contains binding, functional, and ADMET information for a large number of drug-like bioactive molecules. The data is manually abstracted from the primary published literature and contains 20 million bioactivity measurements for more than 2.3 million compounds and 15,100 protein targets.[3,39]

Querying these databases allows practitioners to assemble datasets. For example, a structure-based machine learning task could involve data from the PDBbind database to map the structure of the protein-ligand complex to an activity value *via* regression. In a ligand-based approach, one could collect molecules known to be active or inactive with respect to a given target, and train a binary classifier to separate them as accurately as possible.

2.2.2 Benchmark Datasets for Virtual Screening and Machine Learning

There exists a manifold of virtual screening methods and machine learning tools. Choosing the right method usually depends on the underlying benchmark study, which highlights the advantages and drawbacks of the respective method. It appears quite often that new methodologies are studied on self-compiled datasets, but this has the drawback that it does not allow a fair comparison with other methodologies.

The Directory of Useful Decoys (DUD) is a compilation of 2,950 literature-known ligands for 40 different proteins, which was designed for benchmarking of molecular docking experiments. To each ligand, 36 structurally distinct decoy molecules from the ZINC archive were added, which were expected to have similar physical properties. The database thus comprises a total of 98,266 compounds.[40] Nevertheless, actives and decoys differed in some properties. For example, a difference in the molecular net-charge was noticed in some data sets, which already allowed easy separation. In addition, not enough emphasis was placed on structural diversity in the selection of active compounds, which partially resulted in closely related derivatives. Furthermore, it should be noted that six co-crystal structures of the docking targets in DUD had a diffraction precision index of 1.5 Å or more, an uncertainty in the atomic coordinates that must be taken into account when interpreting the docking results.[41]

To address these issues, the DUD was expanded to include a larger number of active compounds. Prior to their selection, the active compounds were clustered to ensure diversity of chemotypes, and also the previously unevenly distributed net-charge property was considered. The revised Database of Useful Decoys: Extended (DUD-E) contained a total of 102 protein targets, 22,886 active ligands from the ChEMBL database, and 50 decoys for each ligand.[42] Despite the cautious selection of compounds, further correlations between molecular properties and the performance of the virtual screening program were found in a follow-up study by Chaput *et al.* In particular, their study found a biasing distribution of the polar surface area, number of hydrogen donors, and embranchement count. They also reported structural similarities between the molecules in the active set and the ligand of the co-crystallized structure.[43]

The Maximum Unbiased Validation (MUV) datasets are based on PubChem bioactivity data and are also frequently used benchmarking datasets for virtual screening and machine learning models. The ligands for these datasets were selected by a refined nearest neighbor analysis to achieve a spatially unbiased distribution of actives and decoys.[44]

One shall keep in mind that high performance of a machine learning model on a given dataset does not necessarily guarantee general applicability on unseen data, since good evaluation scores could also stem from overfitting. The asymmetric validation embedding (AVE) method by Wallach *et al.* aims at measuring the data redundancy within the training and validation set for ligand-based classification problems, which could induce undetected overfitting. They found several benchmark datasets for virtual screening and classification, including the DUD-E and MUV, to be biased and rewarding memorization rather than generalization.[45]

In an attempt to create an unbiased dataset for machine learning and virtual screening, Rognan *et al.* had developed the LIT-PCBA dataset. It was compiled with experimentally determined bioactivity data from the PubChem BioAssay database (PCBA). The authors had used the aforementioned AVE procedure for unbiasing of the data and assembled a dataset consisting of 15 targets, 7,761 actives, and 382,674 inactives. They also provided a training and validation split for machine learning methods. Table 2.1 gives an overview over the available targets and the number of actives and decoys that are contained in the LIT-PCBA dataset.[19] The experimental validation of the inactive compounds is a huge asset, since artificially generated decoys might in fact be "falsely" active towards the target. It should be noted that most of the datasets in LIT-PCBA are highly imbalanced and contain only few active compounds, which might be acceptable for virtual screening benchmarks, but is not ideal for machine learning.

Last but not least, MoleculeNet[46] is another popular molecular benchmark for the assessment of machine learning models. It is part of the DeepChem[47] package, an open-source Python library that aims for easy accessibility of deep learning techniques in the cheminformatics community. MoleculeNet contains information on 700,000 compounds and their properties in total. The data can be further divided into four categories, *i.e.*, quantum mechanics, physical chemistry, biophysics, and physiology. These categories are intended as levels of detail ranging from the molecular level to the macroscopic effects on the human body. The MoleculeNet

Table 2.1: Number of active and inactive compounds in the LIT-PCBA datasets. AID corresponds to the PubChem Assay ID of the respective targets. All datasets are split into a training and validation split following an 80/20 ratio. Chemical compounds are represented with their corresponding SMILES strings. For each target, there is at least one PDB file of a co-crystallized structure available.[19]

AID	Set	Target	Ligands	Actives	Inactives
492947	ADRB2	Beta2 adrenergic receptor	Agonists	17	311748
1030	ALDH1	Aldehyde dihydrogenase 1	Inhibitors	5363	101874
743075	ESR_ago	Estrogen receptor α	Agonists	13	4378
743080	ESR_antago	Estrogen receptor α	Antagonists	88	3820
588795	FEN1	FLAP Endonuclease	Inhibitors	360	350718
2101	GBA	Glucocerebrosidrase	Inhibitors	163	291241
602179	IDH1	Isocitrate dihydrogenase	Inhibitors	39	358757
504327	KAT2A	Histone acetyltransferase KAT2A	Inhibitors	194	342729
995	MAPK1	Mitogen-activated protein kinase 1	Inhibitors	308	61567
493208	MTORC1	Mechanistic target of rapamycin	Inhibitors	97	32972
1777	OPRK1	Kappa opioid receptor	Agonists	24	269475
1631	PKM2	Pyruvate kinase muscle isoform 2	Inhibitors	546	244679
743094	PPARG	Peroxisome proliferator-activated receptor γ	Inhibitors	24	4071
651631	TP53	Cellular tumor antigen p53	Inhibitors	64	3345
504847	VDR	Vitamin D receptor	Inhibitors	655	262648

package further contains a library for splitting of chemical data, featurization methods, and a series of benchmark results of different machine learning models using several featurization and splitting techniques. The benchmarks are further annotated with an ideal evaluation metric. The transparency of the MoleculeNet benchmarks allows practitioners a comparison with their own methodologies.[31]

2.3 Feature Engineering of Pharmacophore Data

One of the main features of LigandScout is to generate 3D pharmacophore models that can be utilized as filters in virtual screening. However, if these pharmacophores are to be utilized as input for machine learning models, it is necessary to undergo feature engineering in order to transform them into a suitable format. The following section will look at some of the previous efforts to perform feature engineering with pharmacophores, and how they have been used in a machine learning context.

2.3.1 Pharmacophore Fingerprints

Most of the molecular fingerprints were originally developed for similarity searches of large screening databases. The idea of a molecular fingerprint is to represent the molecular structure as a bit sequence that allows easy comparison of two molecules. A commonly used measure of the

Table 2.2: MoleculeNet datasets ordered according to the respective subcategory. All datasets are split into training, validation, and test set following an 80/10/10 ratio. The number of tasks refers to the number of output variables. Except in QM7b, the molecules of all datasets are represented as SMILES strings. The QM datasets and the PDBbind set provide 3D coordinates of the atomic locations. Metric refers to the recommended evaluation metric for the specific dataset, *i.e.* mean absolute error (MAE), root mean squared error (RMSE), and the area-under-curve(AUC) metric of the precision-recall curve (PRC) or receiver-operating curve (ROC). PRC-AUC is recommended for classification datasets with less than 2% positive rates, ROC-AUC is recommended otherwise.[31]

Category	Dataset	Tasks		Compounds	Metric
Quantum mechanics	QM7	1	Regression	7165	MAE
	QM7b	14	Regression	7211	MAE
	QM8	12	Regression	21786	MAE
	QM9	12	Regression	133885	MAE
Physical chemistry	ESOL	1	Regression	1128	RMSE
	FreeSolv	1	Regression	643	RMSE
	Lipophilicity	1	Regression	4200	RMSE
Biophysics	PCBA	128	Classification	439863	PRC-AUC
	MUV	17	Classification	93127	PRC-AUC
	HIV	1	Classification	41913	ROC-AUC
	PDBbind	1	Regression	11908	RMSE
	BACE	1	Classification	1522	ROC-AUC
Physiology	BBBP	1	Classification	2053	ROC-AUC
	Tox21	12	Classification	8014	ROC-AUC
	ToxCast	617	Classification	8615	ROC-AUC
	SIDER	27	Classification	1427	ROC-AUC
	ClinTox	2	Classification	1491	ROC-AUC

similarity of two sets is the Jaccard coefficient, usually referred to as the Tanimoto coefficient in cheminformatics.[48] Molecular fingerprints come in several flavours, most of which are based on the molecular graph. Substructure key-based fingerprints such as the MACCS fingerprint encode the presence of a fixed set of molecular substructures.[49] The Daylight fingerprint is an example for a pathway-based fingerprint, encoding molecular fragments by following a path up to a specific number of bonds. The fingerprint is then created by subsequent hashing of these paths.[50] Circular fingerprints also employ hashing, but instead of paths, they consider the neighborhood of an atom within a certain radius. The extended-connectivity fingerprint (ECFP) is based on Morgan's algorithm and is often used in QSAR modeling.[51,52] A variation of the ECFP is the functional-class fingerprint (FCFP), which uses the functional group affiliation of the atoms instead of the atom-type for the hashing procedure.[51]

It is also possible to create pharmacophore fingerprints, which are usually based on the pharmacophore features of a molecule and the relative distances between them. The ChemAxon PF fingerprint, for example, uses the underlying molecular graph to represent the distance between

pairs of pharmacophore features, whereby the distance corresponds to the number of bonds between a given pair of features. For each possible combination of feature pairs, a histogram is created by binning the occurrences in a molecule into predefined distance ranges. The concatenation of these histograms results in the fingerprint representation of the pharmacophore.[53]

The RDKit pharmacophore fingerprint also uses the topological distance between pharmacophore features, but additionally includes feature triplets. The occurrence of pairs (AA, AB, BB) or triplets (AAA, AAB, ABB, BBB) of pharmacophore features of type A and B within a certain distance are denoted in a bit set. The distance with respect to a triplet is defined as the sum of the pairwise distances of the triplet features. Unlike the ChemAxon fingerprint, the frequency of occurrences is not included. RDKit provides the user with the ability to define the number and size of distance bins individually. The default pharmacophore feature definition in RDKit is described in its `BaseFeatures.fdef` module. An option to use the alternative feature definition from the publication by Gobbi and Poppinger is also provided.[54,55]

The aforementioned fingerprints are based on the underlying molecular graph, but it is also feasible to create fingerprints of 3D pharmacophores. These methods most often rely on relative Euclidean distances between pharmacophore features of pairs or triplets in 3D space.[56–58] For example, Wood *et al.* had used feature types and relative Euclidean distances to encode signatures of 2-point and 3-point pharmacophores, which they used for the evaluation of subpockets of binding sites.[59] Kutlushina *et al.* considered the 3D pharmacophore as a complete graph, where the nodes are labeled with the feature type and the edges are labeled with the rounded relative Euclidean distance between features. For each quadruplet within the graph, a signature was created from the feature types, their distances, and the tetrahedral stereoconfiguration. A graph signature was created from the sorted quadruplet signatures and hashed using an md5-hashing method. They used their method for fast 3D pharmacophore matching without prior alignment.[60] In a very recent study by Tsuda *et al.*, the authors used a fingerprint for 3D pharmacophores for similarity search, which is comparable to the ChemAxon fingerprint in that it uses the relative Euclidean distances of pharmacophore feature pairs to generate histograms. The fingerprint is, in turn, the concatenation of these histograms.[61]

2.3.2 Structure-based Learning with Pharmacophores

Several approaches have already been developed to train machine learning models on structure-based pharmacophores. For example, the pharmacophore interaction fingerprint (Pharm-IF) developed by Sato *et al.* is based on the pairwise relative distances between pharmacophore features of the ligand in the binding pocket. Their datasets were generated from known active compounds in the StARLite database (predecessor of ChEMBL) and randomly selected decoys from the PubChem database for five different protein targets. To include structurally diverse compounds in the datasets, they clustered the actives based on their structural fingerprints and selected those compounds with the highest activity level from each cluster. If experimentally determined crystal structures were not available, the binding poses of the ligands were

determined by docking using the SCHRÖDINGER[62] software. They then created structure-based pharmacophore models using the MOLECULAR OPERATING ENVIRONMENT (MOE).[63] For each feature-pair combination, they created a vector of length 12 to represent relative spatial distances from 0 to 12 Å, and assigned the pairs to the respective entry of these vectors. The concatenated vector was finally used as input for training binary classification models. The best prediction results in terms of the ROC-AUC score and 10%-enrichment factor were obtained using an SVM with radial basis function kernel. It is important to note that their approach requires at least one experimentally determined crystal structure.[64]

One study that is also often mentioned in the context of machine learning with structure-based pharmacophores is the Hot-Spots-Guided Receptor-Based Pharmacophores (HS-PHARM) approach by Rognan *et al.* They trained machine learning algorithms with atom-based fingerprints of known binding pockets to prioritize cavity atoms that might be relevant for ligand binding. Their approach aimed to reduce the number of features of apo-site-generated pharmacophore models.[65]

The goal of the DEEPSITE study by Jimenez *et al.* was to train convolutional neural networks (CNNs) for the prediction of the druggability of protein binding pockets. Voxel representations are a common input for CNNs, which in simple terms are 3D images with RGB values. Analogous to RGB channels, they used pharmacophore channels to create a voxel representation of the binding site.[66] In a follow-up study, they had used their approach to predict the binding affinity of small molecules towards a respective binding site, and suggested that their model could be used as a scoring function.[67]

There also exist some generative approaches that employ structure-based pharmacophores for *de novo* drug design. Skalic *et al.* did rely on a voxel representation of the molecule. They used a seed compound, its three-dimensional shape, and its pharmacophoric features to yield a perturbed 3D representation, which was subsequently used by a system of convolutional and recurrent neural networks to generate a sequence of SMILES tokens.[68] Yoshimori *et al.* used reinforcement learning to generate novel compounds that match a structure-based pharmacophore. They trained a SMILES-generating agent by exploiting the LIGANDSCOUT pharmacophore scoring function as a penalty function.[69]

2.3.3 Ligand-based Learning with Pharmacophores

One of the first attempts to correlate ligand-based pharmacophores with activity values was approached using CATALYST's HYPOGEN tool. It creates a pharmacophore hypothesis from the common features of the most active compounds in the dataset and deletes the features that are common to the set of inactive compounds. The hypothesis is optimized by simulated annealing, in which small perturbations are introduced to the feature coordinates, and only those changes that led to an improvement in the model are retained. The resulting pharmacophore hypothesis could be used to correlate the geometric fit of input molecules with their activity values.[70]

PHASE by SCHRÖDINGER, which is implemented in MAESTRO,[62] uses the 3D conformation

of molecules, aligns them, and generates pharmacophore fields from the input molecules. A pharmacophore field is simply a vectorized box, and can be used in combination with Partial Least Squares (PLS) to predict the activity of input molecules.[71]

Kohlbacher *et al.* proposed a different workflow that did not rely on the underlying input molecule or the selection of the most active compounds in the training set. Their algorithm generates a merged consensus pharmacophore from the training samples, to which the input pharmacophores are aligned. A vector is generated from the inverse relative positions of the features of each aligned pharmacophore with respect to the consensus pharmacophore. This feature representation was used as input to classical machine learning, primarily a random forest model, and correlated with biological activity values. They tested the algorithm on the PHASE study datasets and a set of self-generated benchmark datasets from the ChEMBL database, and showed that even small datasets with 15-20 training samples could already be sufficient to generate robust models. They emphasized the advantage of using pharmacophores, as the coarse level description could facilitate generalization by avoiding bias towards over-represented functional groups in small datasets.[72]

A recent study of Warszycki *et al.* uses a combinatorial 2- and 3-point fingerprint of 2D pharmacophores as input for various machine learning models like (linear) SVMs, logistic regression, and neural networks to discriminate between active and inactive ligands. Their fingerprint is based on the RDKit pharmacophore fingerprint. The authors optimized the respective distance bins and reduced the dimensionality of the resulting vector using an autoencoder. They compared its performance with 11 other fingerprints, including the ChemAxon PF fingerprint, the ECFP and the FCFP fingerprint, and tested their method on several datasets using known active compounds from the ChEMBL database and putative inactive compounds from the ZINC database. Their fingerprint showed slightly better performance than the ECFP4 and FCFP4 fingerprint with respect to the Matthew's correlation coefficient (MCC) metric.[73]

3 API Design

3.1 Preliminaries

The goal of this work was to integrate a machine learning workflow into the LigandScout code base and to comply with the existing internal coding standards. Accordingly, several requirements had to be considered. To ensure the reusability of the code and its readability, clear structuring, descriptive variable and function names, as well as self-explanatory method signatures were essential. Efficiency was considered without compromising the code readability. Java was chosen as the programming language, since the entire code base of LigandScout was implemented with it and the advantage of platform-independence was to be maintained. Carefully selected dependencies were used to keep the application light-weight.

Rather than relying on existing machine learning libraries that could introduce a large amount of transitive dependencies, a custom application programming interface (API) was developed to retain full control and customization, especially when integrating with existing LigandScout tools. This API should achieve an abstract description that embeds the logic of a machine learning workflow. Much inspiration was taken from the API logic of the scikit-learn library,[23] a popular framework for machine learning with Python.

Interfaces and (abstract) classes with the following properties were designed:

- A dataset abstraction to represent collections of data objects and their respective labels
- A featurizer interface to transform data objects into numerical feature vectors for use in machine learning
- Preprocessing tools for shuffling and splitting datasets, and normalizing or standardizing data
- Supervised machine learning models for classification and regression tasks, optionally with probabilistic output and/or online learning capabilities
- Evaluation metrics for assessing model performance
- Hyperparameter tuning *via* cross-validated grid-search

With these abstractions in mind, the following goals should be achieved:

- Adding new models should be easy by implementing the respective interface. This was

demonstrated by implementing an online SVM classifier.

- Using the interface as a facade to wrap existing JAVA class files should also be possible. This strategy was used to incorporate the LIBSVM library,[74] an open-source library for support vector machines.
- It should be further possible to employ existing fingerprints in LIGANDSCOUT to prepare molecules or pharmacophores as input for machine learning models. One advantage of the API is that LIGANDSCOUT tools such as the conformer generator and the pharmacophor generator can be used to create custom pharmacophore fingerprints. This was demonstrated by using the pharmacophore RDF spectrum as machine learning input.

With a working command line application at hand, the workflow should be easy to embed into the LIGANDSCOUT GUI.

3.2 Design Decisions

3.2.1 Package Structure & Hierarchy

Figure 3.1 provides an overview of the organizational structure of the implemented `ml` machine learning package, which consists of three sub-packages: `data`, `models`, and `utils`. The `data` package encompasses data handling and storage, including abstraction and implementation of datasets, feature engineering techniques, and data loading utilities. The `models` package contains abstraction interfaces for supervised learning models and implementing classes for classification and regression models. It also includes the class files of the LIBSVM library and a wrapper class that adapts it to the API logic. The `utils` package includes abstraction interfaces and implementation classes for evaluation metrics for classification and regression models, as well as a hyperparameter tuning package that implements k-fold cross validation techniques. Additionally, it includes implementing classes for data preprocessing techniques, such as the `StandardScaler`.

3.2.2 Dataset Abstraction & Implementation

A dataset is a collection of data objects with corresponding labels and metadata about the class. The `LabeledDataSet` class represents such an instance and is initialized with a `DataObjectArray` and a `LabelArray` (Figure 3.2). Both classes are abstract and contain a generic array to represent the data objects or labels, respectively. Their implementing child classes are parameterized with the desired data type. The `LabeledDataSet` class offers the ability to merge datasets into one and split it into subsets, which is essential for model training and evaluation.

The LIGANDSCOUT software is focused on molecular data and allows the import of ligand sets from *e.g.* `.smi`-, or `.sdf`-files *via* the IDBGEN generator, which creates a `ScreeningDatabase` to store collections of molecules, and their corresponding ligand-based pharmacophores. The

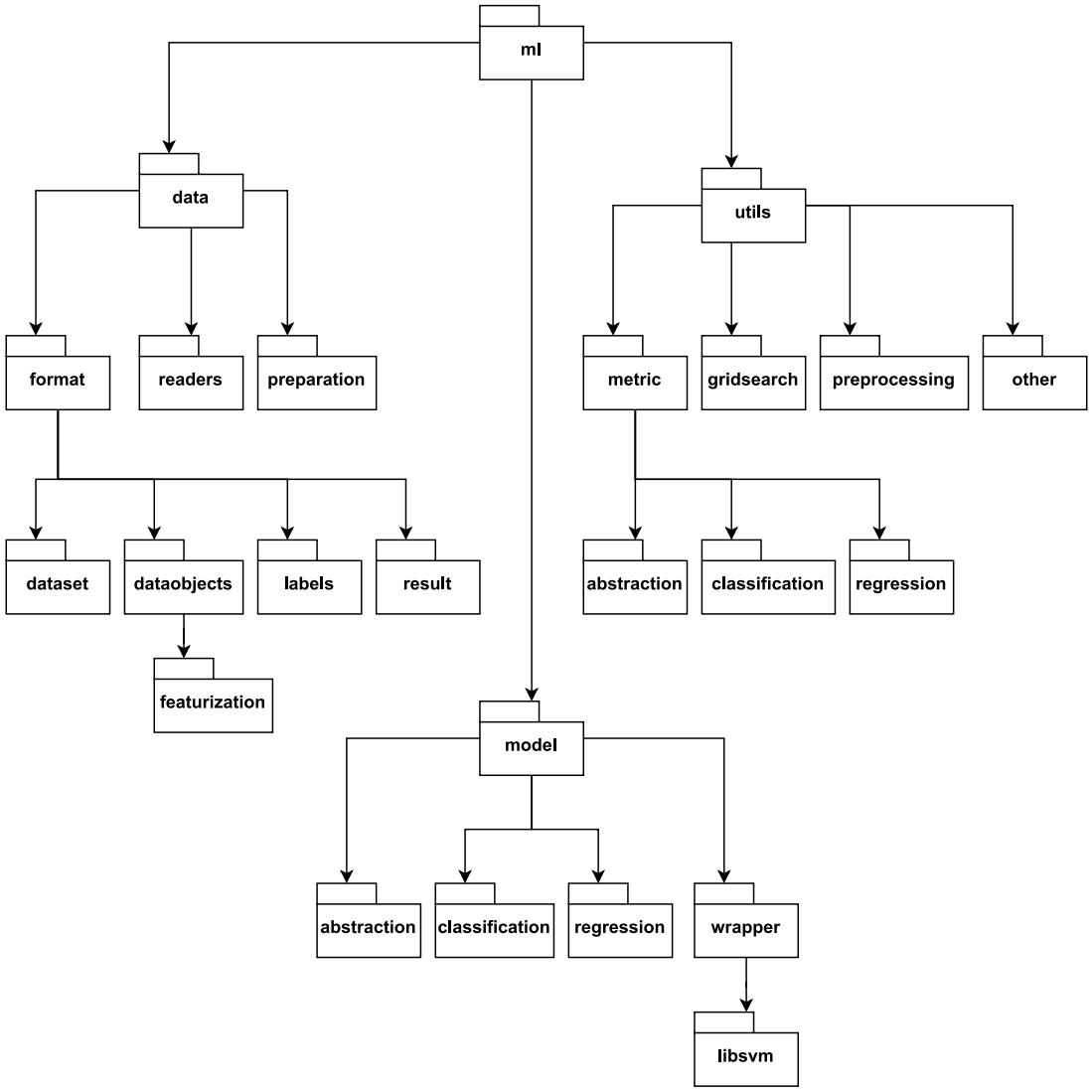

Figure 3.1: The hierarchy of the implemented machine learning package.

iConfGen[21] conformer generator samples the 3D conformational space of the molecules, storing up to 25 conformations per molecule.

However, machine learning models typically require numerical data represented in matrix format. The `FeatureMatrix64` class offers a dense matrix for this general purpose representation. The `LibSvmMatrix64` class is necessary for the `LibSvm` class, which was incorporated with a wrapper. This matrix represents data objects row-wise like the standard feature matrix, but it stores non-zero values only, together with their respective column indices.

Feature engineering is a crucial step in the machine learning pipeline. Transforming the `ScreeningDatabase` into a `FeatureMatrix64` falls under this umbrella, and is represented by the `Featurizer` interface. Two implementing child classes are offered for this purpose. The `MoleculeFingerprinter` class uses featurization techniques based on the molecular graph, such as the ECFP fingerprint, or a simple pharmacophore feature count. The

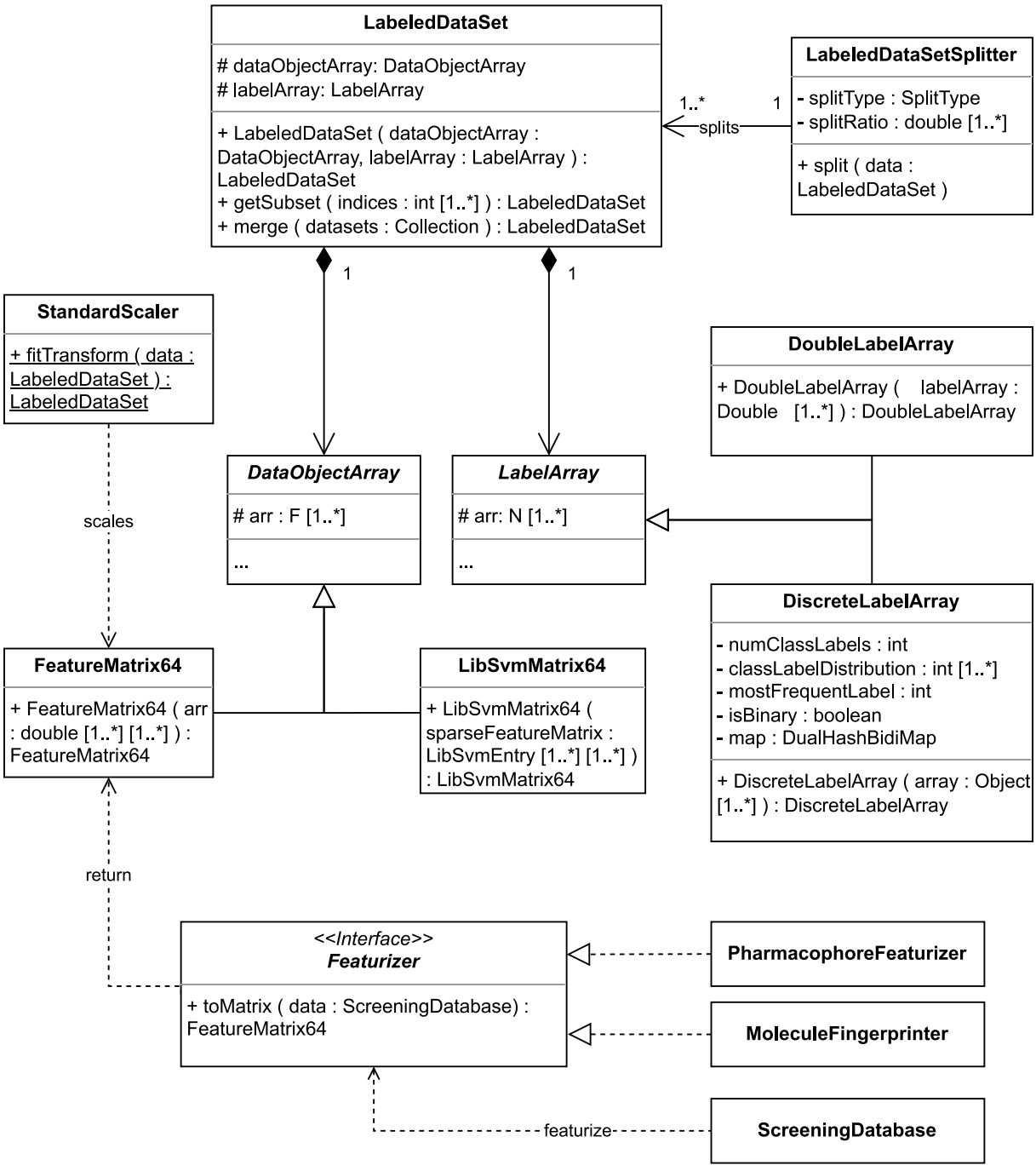

Figure 3.2: UML Diagram of the dataset implementation details.

`PharmacophoreFeaturizer` class uses the pharmacophores of one or more conformations, *e.g.* for the creation of RDF spectra as ML input.

Some machine learning models benefit from preprocessing techniques, such as scaling with a `StandardScaler`, which is shown to improve performance for SVMs. Scaling works only on numerical data.

Supervised learning can be divided into classification and regression tasks. For this reason,

the abstract `LabelArray` class has two implementing child classes. The `DoubleLabelArray` represents continuous `Double` labels for regression tasks, while the `DiscreteLabelArray` is parameterized with the `Integer` type and represents the labels for classification tasks. It represents N labels internally as integers from 0 to $N-1$, and stores a map to transform the original labels to the internal integer representation and back.

To train and evaluate models, it is common to split the data into training, validation, or test sets. The `LabeledDataSetSplitter` facilitates this process and allows stratified splitting to account for the unbalanced nature of many chemical datasets.

3.2.3 Machine Learning Model Abstraction & Implementation

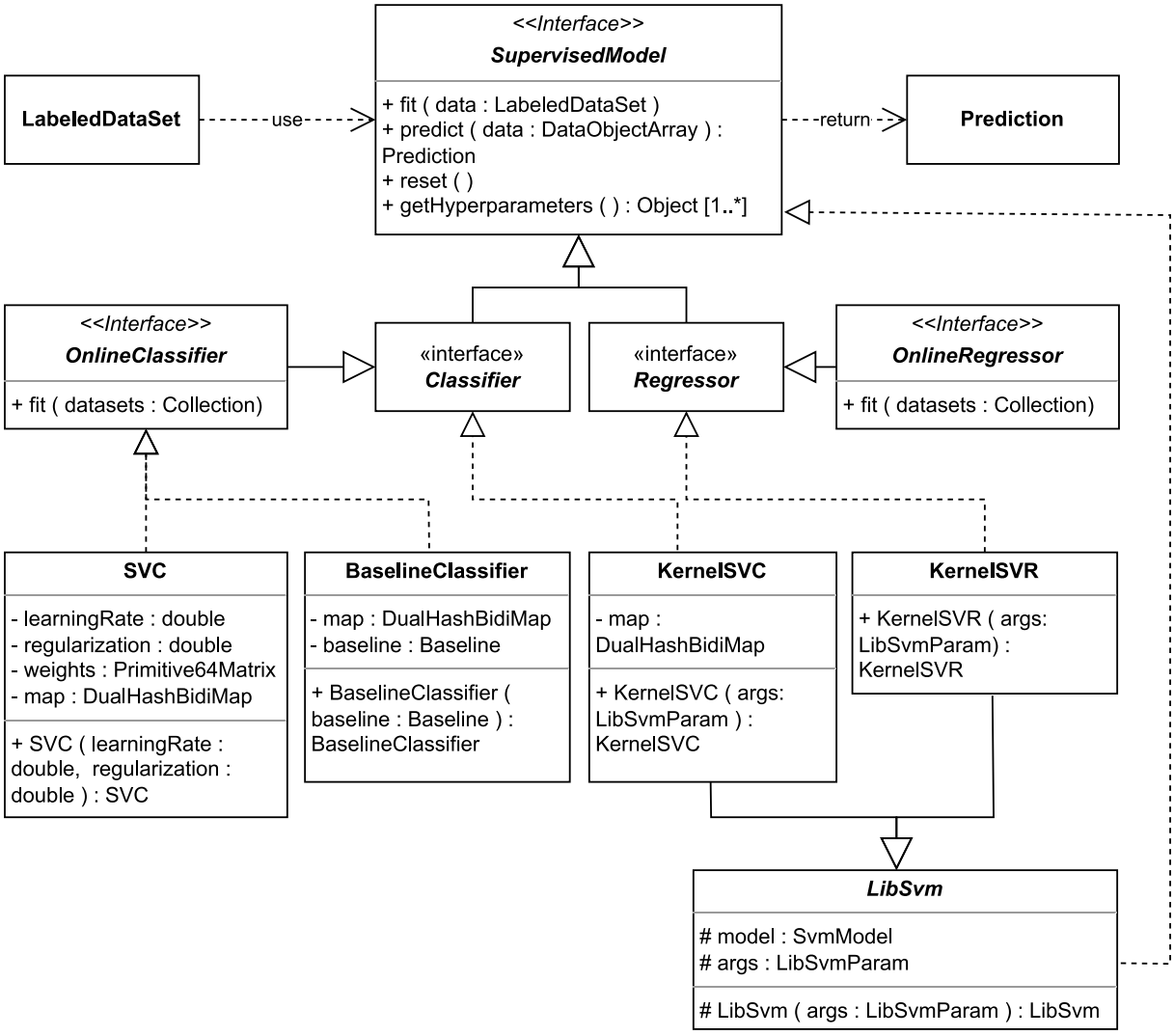

Figure 3.3: UML Diagram of the machine learning model implementation details.

The `SupervisedModel` interface defines the general behavior of a machine learning model, which is depicted in Figure 3.3. A model is trained by fitting it to a `LabeledDataSet` object, and once trained, it can be used to predict labels for a given `DataObjectArray` object. The `predict`

method returns a `Prediction` object that contains the set of predicted labels. Each supervised model has a set of hyperparameters, which are set during the constructor call and can be retrieved as an `Object[]`. Resetting the model keeps the initial hyperparameters, but deletes the internal weights that were determined during the training process.

The `Classifier` and `Regressor` interfaces are subinterfaces of the `SupervisedModel` interface. Classifiers deal with data that has discrete labels and are parameterized with `Integer`, while regressors deal with continuous labels and are parameterized with `Double`.

The machine learning models can be classified as ordinary or online models, and there are different interfaces provided for each. The ordinary model interface offers a `default fit(Collection<LabeledDataSet>)` method that accepts a collection of datasets as input. It merges the datasets into one and calls its `fit` method implementation on the merged dataset. The online model interface overrides this signature, since it can call `fit` multiple times whenever new data arrives, and does not require merging all the training data in advance.

However, extending the `OnlineLearner` interface for online models directly from the `SupervisedModel` interface could result in ambiguous method signatures for those classes that implement the `OnlineLearner` together with the `Classifier` / `Regressor` interface. To avoid this problem, the `Classifier` interface was extended with the `OnlineClassifier` interface and the `Regressor` interface with the `OnlineRegressor` interface. This adds two interfaces to the class hierarchy, but it avoids any ambiguity in method signatures.

The focus of this work has been mainly on binary classification tasks. Two implementing child classes of the `OnlineClassifier` interface are provided. The `BaselineClassifier` is a classifier that predicts the most frequent or random labels, while the `SVC` implementation is a linear support vector machine that uses the regularized hinge-loss formulation as the loss function and stochastic gradient descent as the optimizer.

A `LibSvm` class is also provided, which is a wrapper for the LibSvm library.[74] LibSvm is an open-source library for support vector machines that was originally implemented in C++. It offers SVM implementations for classification, regression, and one-class SVM, as well as several kernel functions. The Java version comes in the form of several Java class files and is intended as a console application. To make LibSvm compatible with our library, an abstract `LibSvm` wrapper class was created that implements the `SupervisedModel` interface. A `LibSvm` object is initialized by passing a `LibSvmParameters` object, which represents the collection of settings for the SVM experiment. Instead of passing settings *via* the console, the user changes a setting by calling the respective setter methods of this object. A `LibSvm` model can perform both classification and regression tasks. To make this behavior conform with the API, `LibSvm` was extended with two children classes: `KernelSVC`, which implements the `Classifier` interface, and `KernelSVR`, which implements the `Regressor` interface. The children classes offer the parameterized `fit` and `predict` methods.

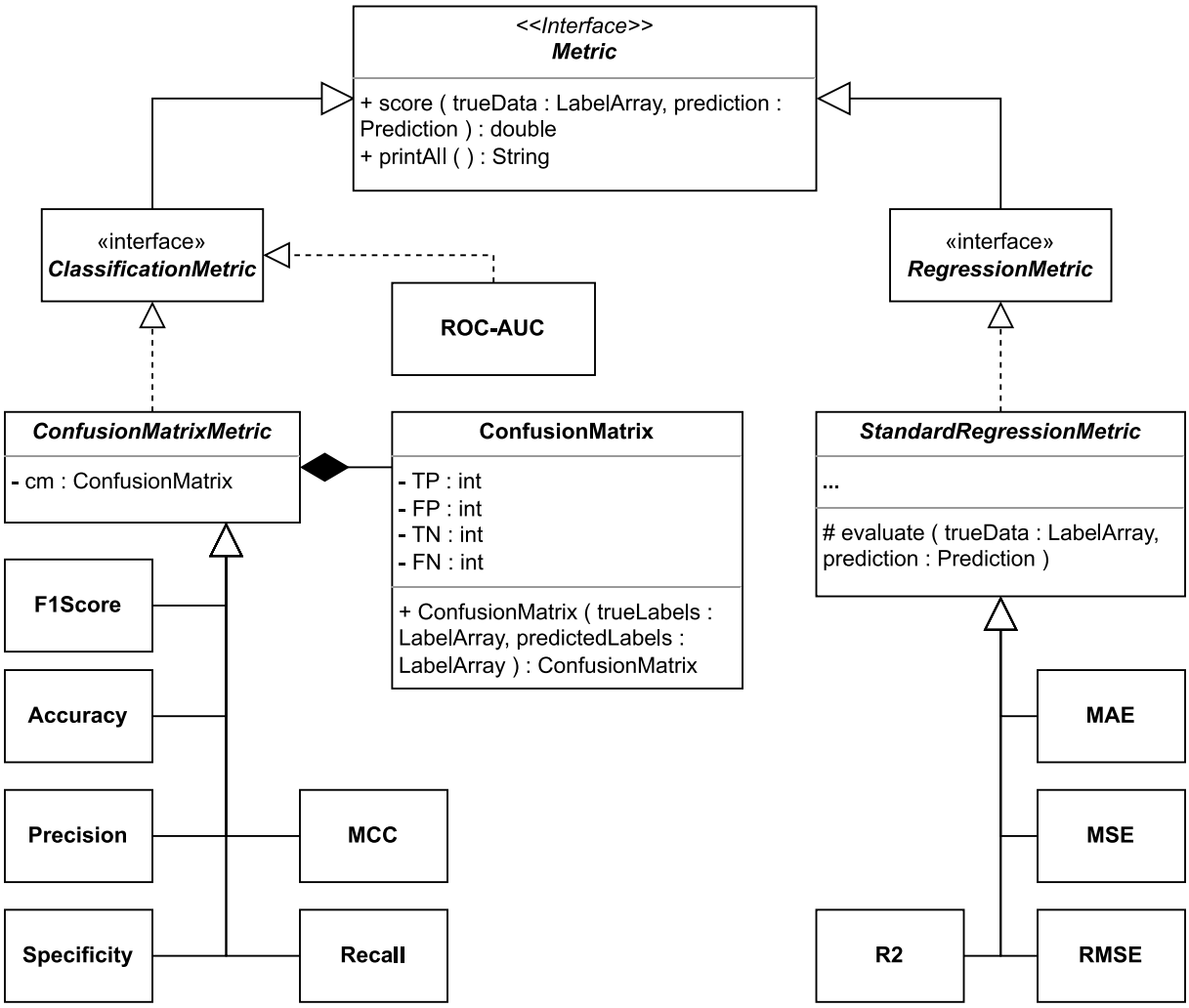

Figure 3.4: UML Diagram of the evaluation metric implementation details.

3.2.4 Evaluation Metric Abstraction & Implementation

In machine learning, a metric is a measure that quantifies the quality of a model's predictions on a given dataset. It takes both the true labels and the predicted labels as inputs and outputs a numerical score that reflects the model's performance. The `Metric` interface (Figure 3.4) encapsulates this functionality. This interface defines the basic behavior of a metric and is extended by the `ClassificationMetric` and `RegressionMetric` interfaces. These interfaces provide additional parameterization for computing scores in classification and regression tasks, respectively.

The abstract `ConfusionMatrixMetric` class is used to represent metrics that are based on the confusion matrix. This class stores the confusion matrix as a field and is extended by implementing child classes such as `Accuracy`, `Precision`, `Recall`, `Specificity`, `MCC`, and `F1Score`. It should be noted that at this point in the project, binary classification metrics have been implemented only. However, the aforementioned metrics can be extended for the multiclass case by using a multiclass confusion matrix. The `ROC-AUC` metric is a probabilistic metric that

can only be calculated for prediction results that contain class probabilities. If the prediction result does not contain probabilistic information, an error is thrown.

In the context of regression tasks, several metrics are available to evaluate the performance of a model, such as Mean Absolute Error (`MAE`), Mean Squared Error (`MSE`), Root Mean Squared Error (`RMSE`), and Coefficient of Determination (`R2`). These metrics are calculated based on the discrepancy between the predicted and true values of the target variable. The abstract `StandardRegressionMetric` class defines their behaviour, which includes a general method to evaluate these differences. The specific formula for each metric is then implemented in the respective child classes, which also provide the `score` method signature.

3.2.5 Model Selection by Grid Search

The process of model selection involves combining several design decisions previously described. The `GridSearch` object requires initialization with a `CombinatorialModelCreator` object, a chosen metric, and the number of folds k (Figure 3.5).

The `CombinatorialModelCreator` is initialized with a `ParameterGrid` and a function for model creation. The `ParameterGrid` contains an `Object[][]` that includes hyperparameter grid points to be screened. The `ParameterGrid` implements the `Iterable` interface from the `Java.lang` package and provides an `iterator()` method. This iterator enables iterating over all possible combinations of the parameter grid, returning an `Object[]` for each parameter combination. The `CombinatorialModelCreator` also implements the `Iterable` interface itself. Its iterator creates `SupervisedModels` iteratively from the `ParameterGrid` . The user-defined model creation function of the `CombinatorialModelCreator` takes the `Object[]` of the parameter grid point as input and returns a `SupervisedModel` instance.

When the `search` method of the `GridSearch` is called on a given `LabeledDataSet` , it performs k-fold cross-validation iteratively on each of the screened Supervised Models. To do so, the `CrossValidation` class is initialized with a `SupervisedModel` instance, a metric, and a number of folds k. The `fitPredict` method is then called on the `LabeledDataSet` instance, which splits the dataset object into k folds. The model is trained k times on the merged $(k-1)$ folds, predicts the labels on the remaining fold, and returns the k evaluation results as a `CrossValidationResult` object. This object provides methods to calculate the mean evaluation score, its standard deviation, and also stores the fit and predict durations.

The `GridSearch` stores a `CrossValidationResult` for each of the cross-validated `SupervisedModels` in a `List` . Upon evaluating all possible hyper parameter combinations, the list of results is ranked with respect to the mean score, and this finishes the `search` method. The `bestParameters` method returns the hyper parameters of the model with the highest-ranked `CrossValidationResult` object. If the user is satisfied with the evaluation results of the `GridSearch` , they can use the obtained best hyper parameters to retrain the model on the full training set and test its performance on a separate test set.

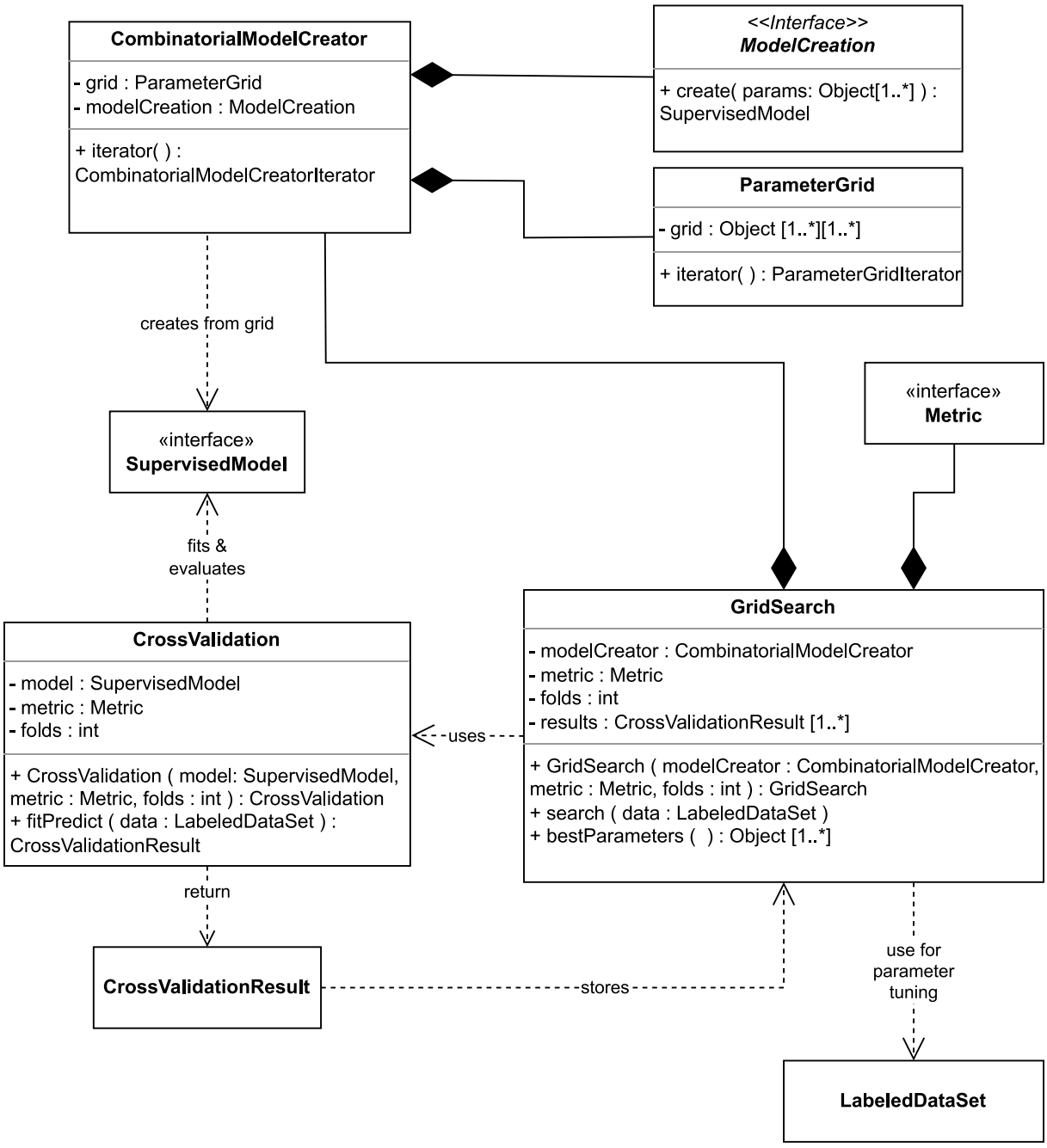

Figure 3.5: UML Diagram of the model selection *via* grid search..

3.3 Where to go from here: A Workflow scenario

LigandScout includes a pre-existing workflow to evaluate a pharmacophore hypothesis against a collection of known compounds, distinguishing between active and inactive ones. The user can upload datasets into the program and designate them as active or inactive. LigandScout then conducts a pharmacophore alignment of the query against the added datasets.

This existing workflow could be adapted to include a binary classification task within LigandScout. The user could add datasets of known active and inactive compounds against a specific

target, as is already provided by the software. Rather than conducting a pharmacophore alignment, the user could access a machine learning menu. The user would then be able to select the train/test split ratio, the featurization method, the machine learning model, the evaluation metric, and the hyperparameter specifications through one or multiple dialogs. To assist less experienced users, standard settings could be provided for each model and featurization method.

Upon completing parameter optimization and model selection, the user would receive a report containing evaluation metrics for their dataset. The trained model and featurization settings could be saved for future use. The model could then be used either to refine the hit list from a previous screening run, or even as an alternative approach to virtual screening.

4 Results & Discussion

4.1 Benchmark Studies

Several benchmark studies were performed to demonstrate the performance of the workflow on binary classification tasks for bioactivity prediction. The datasets were selected from the MoleculeNet[31] and the LIT-PCBA[19] benchmark publications. These collections contain datasets ranging in size from a few thousand to several hundred thousand compounds. To avoid unreasonably long run-times, it was decided to initially limit the benchmark studies to the smaller datasets. The datasets each consist of active and inactive compounds with respect to one target protein. Table 4.1 provides information about the number of compounds per training and test set, together with a short description of the objective. The TP53 and ESR1 datasets are more imbalanced with less known active compounds than the BACE dataset. Training and test data splits were used as provided by the authors. Note that none of the datasets have the same class ratio within the training and test partitioning. This is due to the respective splitting procedures, which aimed at equal coverage of the chemical space rather than a stratified label distribution.

Table 4.1: Bioactivity datasets used for benchmarking purposes in this study.

Target	Training Set		Test Set		Short Description
	active	inactive	active	inactive	
BACE[31]	515	695	176	127	Activity data for a series of 1,513 compounds in relation to inhibition of human β-secretase 1 (BACE-1).[75] Training and test data were split according to the scaffold split in the MoleculeNet publication.[31]
TP53[19]	48	2,550	16	795	3,409 compounds tested for inhibition of the cellular tumor antigen p53. The training and test data were split according to the AVE unbiasing procedure of Wallach *et al.*[45]
ESR1[19]	63	3,026	25	794	3,908 tested agonists for the estrogen receptor α. Training and test data were partitioned according to the AVE unbiasing procedure of Wallach *et al.*[45]

Each dataset consists of four `.smi`-files, *i.e.* `active_train.smi`, `inactive_train.smi`,

`active_test.smi`, and `inactive_test.smi`. These files contain chemical compounds as SMILES-strings. From each of these files a `ScreeningDatabase` was generated, which contains 25 conformations per molecule and the respective ligand-based 3D pharmacophores. The `ScreeningDatabase` is based on a Nitrite database[76] backend and can be stored as a `.ldb2`-file. For this purpose, the LigandScout iDbGen generator with default setting was employed, which internally uses the iConfGen[21] conformer generator.

The datasets were subjected to different featurization techniques to compare their impact on the performance of the machine learning model. The fingerprints used, which are based on the molecular graph, are the ECFP,[51] the FCFP,[51] and the Daylight fingerprint.[50] Each of these fingerprints was created with 1024 bits.

The above-mentioned techniques were compared with fingerprints based on LigandScout pharmacophores. The simplest of these fingerprints is the pharmacophore feature count (FC), which works as follows: The standard pharmacophore feature set consists of six different feature types. The fingerprint has a length of 60 bits, of which ten bits are reserved for each type. For small molecule ligands, an upper limit of ten bits per feature type should usually be sufficient. Each time a feature occurs in a pharmacophore, a corresponding bit is set to `True`.

A more sophisticated representation is the pharmacophore radial distribution function (RDF) spectrum. A pharmacophore consists of a set of features A with cardinality N. A subset $A_m \subseteq A$ contains features of the same type m from the set of feature types $\{1, \ldots, M\}$ and has cardinality N_m. The spectrum $g(r)$ is the concatenation of the subspectra $g_m(r)$ of the respective feature types m. They are computed from a distance parameter r, which is increased incrementally from 0 to 15 Å in steps of 0.1 Å, and the relative distances between the features of a subset A_m and the entire feature set A:

$$g_m(r) = \sum_{i=1}^{N_m} \sum_{\substack{j=1, \\ i \neq j}}^{N} \exp(-\beta(r - r_{ij})^2) \tag{4.1}$$

where r_{ij} is the distance between feature $i \in A_m$ and feature $j \in A$, and β is a smoothing factor. The RDF spectrum is thus a 1D vector representation of a 3D pharmacophore that contains information about its features and their spatial distribution. LigandScout uses the cosine similarity of pharmacophore RDF spectra for hierarchical clustering in the generation process of ligand-based, merged pharmacophores. It could therefore be interesting to find out to what extent this representation is suitable for supervised learning.

Two approaches were tried as part of the benchmark experiments. In the first approach, the pharmacophore of the conformation with the lowest energy of each ligand was used to generate the corresponding spectrum. In the remainder of this paper, this procedure will be referred to as RDF_{min}. In the second approach, the RDF spectra of the pharmacophores of all conformations of a ligand were used. The input for the machine learning is the element-wise average of the respective vectors. In the following, this procedure will be referred to as RDF_{avg}.

It was decided to evaluate model performance with the MCC,[77] following Chicco's[78] recommendation for evaluating binary classification problems in machine learning. The MCC is calculated as follows:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{4.2}$$

where TP is the number of true positive, TN the number of true negative, FP the number of false positive, and FN the number false negative samples. The MCC is the equivalent of the Pearson correlation coefficient for binary classification problems, and thus represents the correlation between true and predicted labels. It ranges from -1 to $+1$, where -1 represents disagreement between true and predicted labels, 0 indicates random prediction, and $+1$ represents correct prediction results. The MCC has the advantage over the accuracy- and the F1-measure that it takes the entire confusion matrix into account, and is therefore particularly useful for working with imbalanced data sets.

The machine learning experiments were performed with the SVM classifier implementation of the LibSvm library.[74] The `KernelSVC`, which extends the `LibSvm` wrapper class, is equipped with various kernel functions, *i.e.*, a linear, a polynomial, a radial basis function (RBF), and a sigmoid kernel. Through a preliminary screening experiment, it was decided to perform all benchmark experiments with the RBF kernel, since classifiers achieved the comparatively highest evaluation scores with it. In addition, preprocessing the data *via* standard scaling was omitted because it did not appear to have any effect on the results.

Hyperparameter selection was performed for all models *via* a 5-fold cross-validated grid search. The optimized hyper parameter was the regularization parameter C, which was selected from $\{0.1, 1, 10, 50, 100\}$. A smaller value of C implies a stronger regularization and thus a less flexible decision boundary, a reduction of C can therefore be advantageous in the case of an overfitted model. The k-fold cross validation works internally with a stratified splitting procedure. The hyperparameters of the top ranked model with respect to the mean evaluation metric were used to retrain the model on the entire training set. The predictive performance of the model was evaluated on the test set and reported without further adjustments.

The results of the benchmark experiments are depicted in Figure 4.1. Note that standard deviations of the evaluation metrics are only given for the validation procedure, which refers to deviations from the mean evaluation score in the 5-fold cross validation. The evaluation of the performance on the test set was performed only once, and therefore does not exhibit a standard deviation.

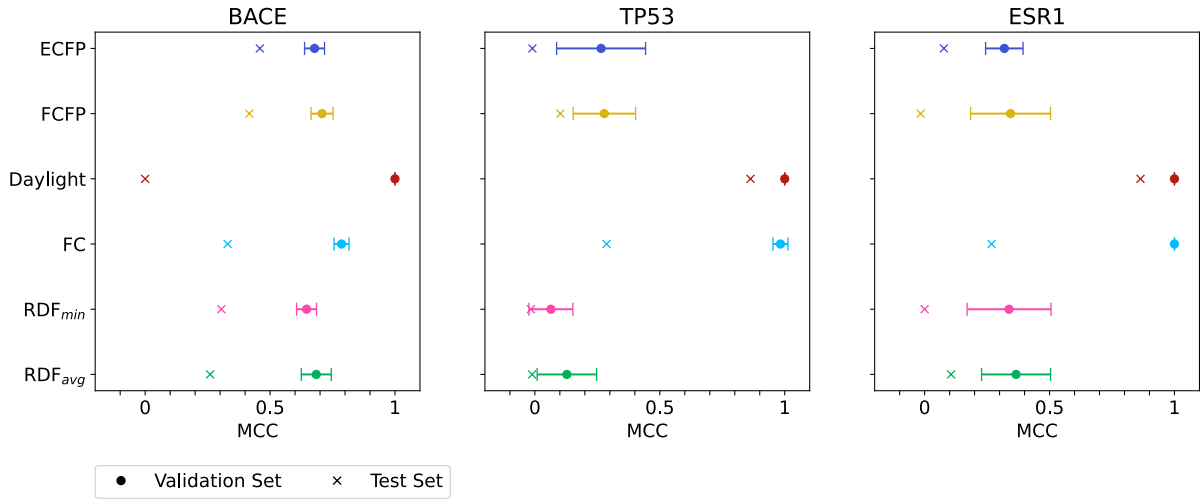

Figure 4.1: Validation and test set evaluation of the BACE, TP53, and ESR1 datasets *w.r.t.* the MCC metric. A SVC classifier with RBF kernel was used in all cases. Please note that the MCC metric ranges from -1 to 1.

4.2 Discussion & Future Work

The molecular fingerprints currently outperform the pharmacophore fingerprints regarding all three datasets. The best results for the MoleculeNet BACE dataset were obtained with circular fingerprints. The Daylight fingerprint performs very well regarding the validation set, but has no predictive power on the test set. This behaviour is reversed in the TP53 and ESR1 datasets of LIT-PCBA. In both cases, the Daylight fingerprint performs best, while test set performance with the circular fingerprints is random. The reason for this behavior is not clear, but could be related to the different data processing and splitting methods of the MoleculeNet and LIT-PCBA publications.

Recall that pharmacophores are defined by a set of features and their spatial orientation. Interestingly, counting the pharmacophore features already seems to be sufficient to achieve a high evaluation score in each of the validation sets and a reasonable score in the respective test sets. This confirms the importance of the pharmacophore feature types and the number of their occurrences for ligand-based activity prediction. However, the significant difference between the validation and test results indicates a strong overfitting. A pharmacophore descriptor containing also spatial information could lead to a classifier that generalizes better on unseen data.

The RDF spectrum fingerprint takes into account the relative Euclidean distance between features of 3D pharmacophores. However, this did not lead to an improvement in the evaluation results. The performance is comparable to the feature count in the BACE dataset and only random in the TP53 and ESR1 datasets. The difficulty of the RDF spectrum is that it relies on conformer sampling of the underlying ligand, which introduces additional problems, most importantly that the active conformation of the ligand in the binding pocket is not known. Choosing the conformation with the lowest energy does not guarantee the selection of the bioactive con-

formation at the binding site, and by taking the average of all conformations, the information about the active site can be superimposed by the signal of all other conformations.

Fingerprints that are based on the pharmacophore RDF spectrum might work better in conjunction with prior alignment to select a suitable molecular conformation. However, this would require finding an appropriate template structure, similar to the approach that was described by Kohlbacher *et al.*[72] Prior docking of the ligands to select a conformation might also be possible, as was done in the study by Sato *et al.*, but would require at least one crystal structure. Further, it should be kept in mind that the quality of docking experiments relies on the underlying docking function. In any case, both alignment and docking would increase the computational cost.

The feature count works reasonably well, but tends to overfit, while RDF spectrum based fingerprints seem to be too sophisticated, and additionally bear the problem of choosing a suitable conformation. Graph-based pharmacophore fingerprints might offer a good compromise. Such a fingerprint could be implemented similar to the RDKit pharmacophore fingerprint that was described earlier in Section 2.3.1, by binning of 2-point and 3-point pharmacophores based on the topological distance that is defined by the underlying molecular graph. LigandScout has methods for the generation of 2-point and 3-point pharmacophores available, which would make it easy to implement such an interaction fingerprint. The approach would however be restricted to pharmacophores that were directly derived from molecules. Implementing such a fingerprint might be part of future investigations.

The generation of 3D pharmacophore fingerprints involves conformation sampling prior to the RDF calculation, which adds an additional computational cost. Runtime experiments were not yet performed, but shall be carried out in future benchmark studies to complement the prediction performance benchmarks of the respective classifiers.

The benchmarks show that the choice of the right fingerprint depends strongly on the underlying data set. Practitioners should therefore always try the range of available fingerprints to find the one best suited to their problem. In turn, it will be important to test newly developed fingerprints with many different datasets to draw a good picture of the general predictive capabilities of the underlying classifier.

A major drawback of the current state of the machine learning workflow is that it does not provide a scaffold-based splitting method. Currently, users need to provide pre-split datasets, if they want to consider this aspect in their experiments. In this benchmark study, the data splits were already available from the respective publications. If this had not been the case, an open-source software like DataWarrior[79] could have been used for the purpose of a diverse subset selection. Adding this functionality to the workflow would have its benefits, since k-fold cross validation relies on the underlying splitting procedure. Moreover, an automated splitting procedure would allow to run benchmark experiments multiple times with several random seeds, which would add a standard deviation to the test set evaluation scores, and therefore enhance the statistical relevance of the results.

The benchmarks were performed with medium-sized data sets. In practice, however, it is often the case that only very few active compounds are known. Therefore, it would be interesting to investigate the influence of the dataset size on the classifier performance. This could give an indication of how many compounds are needed to create a robust classifier. Training machine learning models on very large datasets presents further challenges. The computational complexity of kernelized SVMs scales roughly between $O(M \cdot N^2)$ and $O(M \cdot N^3)$, where M is the number of features and N is the number of samples, which can quickly become computationally infeasible for a large number of data points. For the training with big data, algorithms with a complexity close to linear are preferred.

At the current stage of the project, experiments were only conducted with SVMs, disregarding RF and kNN classifiers, which are also routinely used in CADD. Neural networks are powerful tools, but their training requires a large number of samples. They may therefore be less suitable for experiments, where only few active compounds are known. Nevertheless, given the recent advances in the use of GNNs for molecular datasets, it may still be interesting to see how well they work with pharmacophores as input. One could also investigate completely different featurization methods, such as point cloud descriptors or voxel representations. Moreover, there exist kernel functions that were specifically designed for pharmacophores,[80] which could in turn be utilized for the customization of SVMs.

The present workflow is designed for ligand-based tasks, but future steps could be to extend the workflow for structure-based pharmacophores. In addition, it may be of interest to expand the scope from single-target classification to multi-target classification or regression. And last but not least, the graphical user interface has to be developed to make all this available, not only as a console application, but as a novel tool for LigandScout users.

5 Conclusion

The aim of the present work was to develop and implement a machine learning workflow for the analysis of molecular data with pharmacophores. Pharmacophores are an abstract description of the ligand-target interactions, making them an interesting input for machine learning to train bioactivity prediction models. The workflow was integrated into the LigandScout API, a pharmacophore modeling software that provides a set of methods for processing chemical data and creating 3D pharmacophores.

Interfaces and abstract classes for all necessary steps for training machine learning models were designed and implemented. The starting point of the workflow is the creation of a LigandScout screening library of known active and inactive molecules for a given target protein. A featurization step converts this library into a format suitable for machine learning. Both molecular and pharmacophore fingerprints can be used for this purpose. The featurized data is used for training machine learning models. Currently, only SVMs are integrated, but it is also possible to include other models. The workflow provides methods for model selection, using k-fold cross-validated grid search.

The presented workflow was accompanied by benchmark experiments to investigate the predictive performance of the integrated machine learning models with the existing fingerprinting methods. Several molecular and pharmacophore fingerprints were compared in combination with the implemented SVM models. Their performance was evaluated with three benchmark data sets from MoleculeNet and LIT-PCBA. Although molecular fingerprinting provided the better prediction results, pharmacophore fingerprinting still offers much room for improvement.

There are a lot of ways to extend the workflow that would have been beyond the scope of this thesis. In the benchmark experiments, only existing fingerprints were used that were not specifically developed or modified for machine learning. On the one hand, the existing pharmacophore fingerprint methods could be refined, on the other hand, completely new fingerprints could be implemented. Moreover, other machine learning models could be added to the workflow like RF or kNN classifiers. Furthermore, the workflow could be extended from ligand-based to structure-based methods.

The proposed workflow could be integrated into the GUI of LigandScout and used for training of machine learning models on biactivity data and subsequent screening of molecular databases or the refinement of hit lists. This contribution might add machine learning models as an alternative method to the toolbox of LigandScout users.

Bibliography

[1] C. G. Wermuth, C. R. Ganellin, P. Lindberg, and L. A. Mitscher, "Glossary of terms used in medicinal chemistry (iupac recommendations 1998)," *Pure Appl. Chem.*, vol. 70, no. 5, pp. 1129–1143, 1998.

[2] G. Sliwoski, S. Kothiwale, J. Meiler, and E. W. Lowe, "Computational methods in drug discovery," *Pharmacol. Rev.*, vol. 66, no. 1, pp. 334–395, 2014.

[3] "Chembl." `https://www.ebi.ac.uk/chembl/`. Accessed: 2023-04-17.

[4] "Pubchem." `https://pubchem.ncbi.nlm.nih.gov/`. Accessed: 2023-04-17.

[5] E. N. Muratov, J. Bajorath, R. P. Sheridan, I. V. Tetko, D. Filimonov, V. Poroikov, T. I. Oprea, I. I. Baskin, A. Varnek, A. Roitberg, O. Isayev, S. Curtalolo, D. Fourches, Y. Cohen, A. Aspuru-Guzik, D. A. Winkler, D. Agrafiotis, A. Cherkasov, and A. Tropsha, "Qsar without borders," *Chem. Soc. Rev.*, vol. 49, pp. 3525–3564, 6 2020.

[6] F. Mayr, M. Wieder, O. Wieder, and T. Langer, "Improving small molecule pk a prediction using transfer learning with graph neural networks," *Front. Chem.*, vol. 10, 2022.

[7] O. Wieder, M. Kuenemann, M. Wieder, T. Seidel, C. Meyer, S. D. Bryant, and T. Langer, "Improved lipophilicity and aqueous solubility prediction with composite graph neural networks," *Molecules*, vol. 26, no. 20, p. 6185, 2021.

[8] L. Zhang, H. Zhang, H. Ai, H. Hu, S. Li, J. Zhao, and H. Liu, "Applications of machine learning methods in drug toxicity prediction," *Curr. Top. Med. Chem.*, vol. 18, no. 12, pp. 987–997, 2018.

[9] T. Fox and J. M. Kriegl, "Machine learning techniques for in silico modeling of drug metabolism," *Curr. Top. Med. Chem.*, vol. 6, no. 15, pp. 1579–1591, 2006.

[10] L. Zhao, H. L. Ciallella, L. M. Aleksunes, and H. Zhu, "Advancing computer-aided drug discovery (cadd) by big data and data-driven machine learning modeling," *Drug Discov. Today*, vol. 25, no. 9, pp. 1624–1638, 2020.

[11] O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, and T. Langer, "A compact review of molecular property prediction with graph neural networks," *Drug Discov. Today Technol.*, vol. 37, pp. 1–12, 2020.

[12] X. Yang, Y. Wang, R. Byrne, G. Schneider, and S. Yang, "Concepts of artificial intelligence for computer-assisted drug discovery," *Chem. Rev.*, vol. 119, no. 18, pp. 10520–10594, 2019.

[13] G. Wolber and T. Langer, "Ligandscout: 3-d pharmacophores derived from protein-bound

ligands and their use as virtual screening filters," *J. Chem. Inf. Model.*, vol. 45, no. 1, pp. 160–169, 2005.

[14] "Inte:ligand: Your partner for in-silico drug discovery." `https://www.inteligand.com/`. Accessed: 2022-10-6.

[15] T. Seidel, S. D. Bryant, G. Ibis, G. Poli, and T. Langer, "3d pharmacophore modeling techniques in computer-aided molecular design using ligandscout," *Tutorials in Chemoinformatics*, pp. 279–309, 2017.

[16] G. Wolber, T. Seidel, F. Bendix, and T. Langer, "Molecule-pharmacophore superpositioning and pattern matching in computational drug design," *Drug Discov. Today*, vol. 13, no. 1-2, pp. 23–29, 2008.

[17] T. Seidel, O. Wieder, A. Garon, and T. Langer, "Applications of the pharmacophore concept in natural product inspired drug design," *Mol. Inform.*, vol. 39, 2020.

[18] F. F. Vajdos, L. R. Hoth, K. F. Geoghegan, S. P. Simons, P. K. LeMotte, D. E. Danley, M. J. Ammirati, and J. Pandit, "The 2.0 å crystal structure of the erα ligand-binding domain complexed with lasofoxifene," *Protein Sci.*, vol. 16, no. 5, pp. 897–905, 2007.

[19] V.-K. Tran-Nguyen, C. Jacquemard, and D. Rognan, "Lit-pcba: an unbiased data set for machine learning and virtual screening," *J. Chem. Inf. Model.*, vol. 60, no. 9, pp. 4263–4273, 2020.

[20] C. Permann, T. Seidel, and T. Langer, "Greedy 3-point search (g3ps)—a novel algorithm for pharmacophore alignment," *Molecules*, vol. 26, no. 23, p. 7201, 2021.

[21] G. Poli, T. Seidel, and T. Langer, "Conformational sampling of small molecules with icon: Performance assessment in comparison with omega," *Front. Chem.*, vol. 6, p. 229, 2018.

[22] "Python." `https://www.python.org/`. Accessed: 2023-05-05.

[23] "scikit-learn." `https://scikit-learn.org/stable/`. Accessed: 2023-05-05.

[24] "Tensorflow." `https://www.tensorflow.org/`. Accessed: 2023-05-05.

[25] "Pytorch." `https://pytorch.org/`. Accessed: 2023-05-05.

[26] "Java." `https://www.java.com/en/`. Accessed: 2023-05-05.

[27] "Weka 3." `https://www.cs.waikato.ac.nz/ml/weka/`. Accessed: 2023-05-05.

[28] "Elki." `https://elki-project.github.io/`. Accessed: 2023-05-05.

[29] "Sparkml." `https://spark.apache.org/docs/latest/ml-guide.html`. Accessed: 2023-05-05.

[30] S. J. Warnett and U. Zdun, "Architectural design decisions for the machine learning workflow," *Computer*, vol. 55, pp. 40–51, 2022.

[31] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: a benchmark for molecular machine learning," *Chem. Sci.*, vol. 9,

no. 2, pp. 513–530, 2018.

[32] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B. A. Shoemaker, P. A. Thiessen, B. Yu, *et al.*, "Pubchem 2023 update," *Nucleic Acids Res.*, vol. 51, no. D1, pp. D1373–D1380, 2023.

[33] T. Sterling and J. J. Irwin, "Zinc 15–ligand discovery for everyone," *J. Chem. Inf. Model.*, vol. 55, no. 11, pp. 2324–2337, 2015.

[34] J. J. Irwin, K. G. Tang, J. Young, C. Dandarchuluun, B. R. Wong, M. Khurelbaatar, Y. S. Moroz, J. Mayfield, and R. A. Sayle, "Zinc20—a free ultralarge-scale chemical database for ligand discovery," *J. Chem. Inf. Model.*, vol. 60, no. 12, pp. 6065–6073, 2020.

[35] "Zinc20." `https://zinc.docking.org/`. Accessed: 2023-04-17.

[36] S. K. Burley, H. M. Berman, G. J. Kleywegt, J. L. Markley, H. Nakamura, and S. Velankar, "Protein data bank (pdb): the single global macromolecular structure archive," *Protein crystallography: methods and protocols*, pp. 627–641, 2017.

[37] "Pdb." `https://www.rcsb.org/`. Accessed: 2023-04-17.

[38] Z. Liu, Y. Li, L. Han, J. Li, J. Liu, Z. Zhao, W. Nie, Y. Liu, and R. Wang, "Pdb-wide collection of binding data: current status of the pdbbind database," *Bioinformatics*, vol. 31, no. 3, pp. 405–412, 2015.

[39] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte, *et al.*, "The chembl database in 2017," *Nucleic Acids Res.*, vol. 45, no. D1, pp. D945–D954, 2017.

[40] N. Huang, B. K. Shoichet, and J. J. Irwin, "Benchmarking sets for molecular docking," *J. Med. Chem.*, vol. 49, no. 23, pp. 6789–6801, 2006.

[41] P. C. Hawkins, G. L. Warren, A. G. Skillman, and A. Nicholls, "How to do an evaluation: pitfalls and traps," *J. Comput. Aided Mol. Des.*, vol. 22, no. 3-4, pp. 179–190, 2008.

[42] M. M. Mysinger, M. Carchia, J. J. Irwin, and B. K. Shoichet, "Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking," *J. Med. Chem.*, vol. 55, no. 14, pp. 6582–6594, 2012.

[43] L. Chaput, J. Martinez-Sanz, N. Saettel, and L. Mouawad, "Benchmark of four popular virtual screening programs: construction of the active/decoy dataset remains a major determinant of measured performance," *J. Cheminformatics*, vol. 8, no. 1, pp. 1–17, 2016.

[44] S. G. Rohrer and K. Baumann, "Maximum unbiased validation (muv) data sets for virtual screening based on pubchem bioactivity data," *J. Chem. Inf. Model.*, vol. 49, no. 2, pp. 169–184, 2009.

[45] I. Wallach and A. Heifets, "Most ligand-based classification benchmarks reward memorization rather than generalization," *J. Chem. Inf. Model.*, vol. 58, no. 5, pp. 916–932, 2018.

[46] "Moleculenet." `https://moleculenet.org/`. Accessed: 2023-04-17.

[47] "Deepchem." `https://deepchem.io/`. Accessed: 2023-04-17.

[48] A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé, and G. Pujadas, "Molecular fingerprint similarity search in virtual screening," *Methods*, vol. 71, pp. 58–63, 2015.

[49] J. L. Durant, B. A. Leland, D. R. Henry, and J. G. Nourse, "Reoptimization of mdl keys for use in drug discovery," *J. Chem. Inf. Comput. Sci.*, vol. 42, no. 6, pp. 1273–1280, 2002.

[50] "Daylight theory: Fingerprints." `https://www.daylight.com/dayhtml/doc/theory/theory.finger.html`. Accessed: 2023-05-05.

[51] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *J. Chem. Inf. Model.*, vol. 50, no. 5, pp. 742–754, 2010.

[52] H. L. Morgan, "The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service.," *J. Chem. Doc.*, vol. 5, no. 2, pp. 107–113, 1965.

[53] O. F. Güner, *Pharmacophore perception, development, and use in drug design*, vol. 2. Internat'l University Line, 2000.

[54] A. Gobbi and D. Poppinger, "Genetic optimization of combinatorial libraries," *Biotechnol. Bioeng.*, vol. 61, no. 1, pp. 47–54, 1998.

[55] "Rdkit: Open-source cheminformatics.." `https://www.rdkit.org`. Accessed: 2023-05-03.

[56] M. J. McGregor and S. M. Muskal, "Pharmacophore fingerprinting. 1. application to qsar and focused library design," *J. Chem. Inf. Comput. Sci.*, vol. 39, no. 3, pp. 569–574, 1999.

[57] F. Bonachéra, B. Parent, F. Barbosa, N. Froloff, and D. Horvath, "Fuzzy tricentric pharmacophore fingerprints. 1. topological fuzzy pharmacophore triplets and adapted molecular similarity scoring schemes," *J. Chem. Inf. Model.*, vol. 46, no. 6, pp. 2457–2477, 2006.

[58] S. Askjaer and M. Langgård, "Combining pharmacophore fingerprints and pls-discriminant analysis for virtual screening and sar elucidation," *J. Chem. Inf. Model.*, vol. 48, no. 3, pp. 476–488, 2008.

[59] D. J. Wood, J. d. Vlieg, M. Wagener, and T. Ritschel, "Pharmacophore fingerprint-based approach to binding site subpocket similarity and its application to bioisostere replacement," *J. Chem. Inf. Model.*, vol. 52, no. 8, pp. 2031–2043, 2012.

[60] A. Kutlushina, A. Khakimova, T. Madzhidov, and P. Polishchuk, "Ligand-based pharmacophore modeling using novel 3d pharmacophore signatures," *Molecules*, vol. 23, no. 12, p. 3094, 2018.

[61] F. Berenger and K. Tsuda, "3d-sensitive encoding of pharmacophore features," *J. Chem. Inf. Model.*, 2023.

[62] "Maestro | schrödinger." `https://www.schrodinger.com/products/maestro`. Accessed: 2023-05-05.

[63] "Molecular operating environment (moe)." `https://www.chemcomp.com/Products.htm`. Accessed: 2023-05-05.

[64] T. Sato, T. Honma, and S. Yokoyama, "Combining machine learning and pharmacophore-based interaction fingerprint for in silico screening," *J. Chem. Inf. Model.*, vol. 50, pp. 170–185, 2010.

[65] C. Barillari, G. Marcou, and D. Rognan, "Hot-spots-guided receptor-based pharmacophores (hs-pharm): a knowledge-based approach to identify ligand-anchoring atoms in protein cavities and prioritize structure-based pharmacophores," *J. Chem. Inf. Model.*, vol. 48, no. 7, pp. 1396–1410, 2008.

[66] J. Jiménez, S. Doerr, G. Martínez-Rosell, A. S. Rose, and G. De Fabritiis, "Deepsite: protein-binding site predictor using 3d-convolutional neural networks," *Bioinformatics*, vol. 33, no. 19, pp. 3036–3042, 2017.

[67] J. Jiménez, M. Skalic, G. Martinez-Rosell, and G. De Fabritiis, "K deep: protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks," *J. Chem. Inf. Model.*, vol. 58, no. 2, pp. 287–296, 2018.

[68] M. Skalic, J. Jiménez, D. Sabbadin, and G. De Fabritiis, "Shape-based generative modeling for de novo drug design," *J. Chem. Inf. Model.*, vol. 59, no. 3, pp. 1205–1214, 2019.

[69] A. Yoshimori, E. Kawasaki, C. Kanai, and T. Tasaka, "Strategies for design of molecular structures with a desired pharmacophore using deep reinforcement learning," *Chem. Pharm. Bull.*, vol. 68, no. 3, pp. 227–233, 2020.

[70] H. Li, J. Sutter, and R. Hoffmann, "Hypogen: an automated system for generating 3d predictive pharmacophore models," *Pharmacophore perception, development, and use in drug design*, vol. 2, p. 171, 2000.

[71] S. L. Dixon, A. M. Smondyrev, E. H. Knoll, S. N. Rao, D. E. Shaw, and R. A. Friesner, "Phase: a new engine for pharmacophore perception, 3d qsar model development, and 3d database screening: 1. methodology and preliminary results," *J. Comput. Aided Mol. Des.*, vol. 20, pp. 647–671, 2006.

[72] S. M. Kohlbacher, T. Langer, and T. Seidel, "Qphar: quantitative pharmacophore activity relationship: method and validation," *J. Cheminformatics*, vol. 13, 2021.

[73] D. Warszycki, Ł. Struski, M. Smieja, R. Kafel, and R. Kurczab, "Pharmacoprint: A combination of a pharmacophore fingerprint and artificial intelligence as a tool for computer-aided drug design," *J. Chem. Inf. Model.*, vol. 61, no. 10, pp. 5054–5065, 2021.

[74] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[75] G. Subramanian, B. Ramsundar, V. Pande, and R. A. Denny, "Computational modeling of β-secretase 1 (bace-1) inhibitors using ligand based approaches," *J. Chem. Inf. Model.*,

vol. 56, no. 10, pp. 1936–1949, 2016.

[76] "Nitrite database." `https://www.dizitart.org/nitrite-database.html`. Accessed: 2023-15-05.

[77] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochim. Biophys. Acta - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[78] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Min.*, vol. 10, no. 1, p. 35, 2017.

[79] "Datawarrior." `https://openmolecules.org/datawarrior/`. Accessed: 2023-19-05.

[80] P. Mahé, L. Ralaivola, V. Stoven, and J.-P. Vert, "The pharmacophore kernel for virtual screening with support vector machines," *J. Chem. Inf. Model.*, vol. 46, no. 5, pp. 2003–2014, 2006.

Appendices

Abstract

Pharmacophores provide an abstract description of the ligand-target interaction. They are an established tool in computer-aided drug design (CADD) and have been used in virtual screening for many years. Machine learning (ML) is commonly used in drug design, *e.g.* for the prediction of the bioactivity of small molecules. For this purpose, ML models are often trained on molecular structures; using the pharmacophore representation is less common. However, since pharmacophores inherently describe interactions, they should present a useful input to train ML models for bioactivity prediction.

The goal of this work was to develop a framework that would provide ML methods for CADD in a user-friendly and accessible way. LigandScout is a computational drug design software that employs 3D pharmacophore modeling for screening large molecular databases. Herein, the design and implementation of a Java module for integrating a ML workflow into the LigandScout API is presented. The workflow includes featurization of chemical datasets, training of ML models, and tuning of hyper parameters for model selection.

The package currently contains algorithms for support vector machines, but can easily be extended with additional models. In addition, it offers featurization techniques based on both the molecular graph and LigandScout pharmacophores. The integrated workflow was tested by training binary classification models on three different bioactivity datasets. Although the molecular fingerprints outperformed the pharmacophore fingerprints, there appeared to be much potential for further improvements. The presented workflow could provide LigandScout users with an additional tool for virtual screening and the refinement of virtual screening hit lists.

Zusammenfassung

Pharmacophore bieten eine abstrakte Beschreibung der Interaktion zwischen Liganden und deren Zielprotein. Sie sind ein etabliertes Werkzeug im Computer-gestützten Wirkstoffdesign (CADD) und werden seit vielen Jahren im virtuellen Screening eingesetzt. Maschinelles Lernen (ML) wird regelmäßig im Wirkstoffdesign genutzt, um zum Beispiel die Bioaktivität von kleinen Molekülen hervorzusagen. Hierfür werden ML-Modelle in der Regel auf molekularen Strukturen trainiert, die Pharmacophordarstellung wird hierfür seltener verwendet. Da Pharmacophore für die Beschreibung von Interaktionen konzipiert sind, sollten sie einen nützlichen Input für ML-Modelle zur Bioaktivitätvorhersage darstellen.

Das Ziel dieser Arbeit war die Entwicklung eines Frameworks, um ML-Methoden für CADD benutzerfreundlich und leicht zugänglich anzubieten. LigandScout ist eine Software für computergestütztes Wirkstoffdesign und nutzt 3D Pharmacophormodellierung für das Durchsuchen großer Moleküldatenbanken. Im Folgenden wird das Design und die Implementierung eines Java Moduls vorgestellt, um einen ML-Workflow in die LigandScout API einzubetten. Der Workflow beinhaltet Methoden zur Featuredarstellung chemischer Datensätze, für das Trainieren von ML-Modellen und das Einstellen derer Hyperparameter.

Aktuell enthält der Workflow Algorithmen für Support Vector Machines, eine Erweiterung mit weiteren Modellen ist jederzeit möglich. Die Methoden zur Featuredarstellung basieren sowohl auf dem Molekülgraphen als auch auf LigandScout Pharmacophoren. Der eingebettete Workflow wurde getestet, indem binäre Klassifikationsmodelle auf verschiedenen Bioaktivitätsdatensätzen trainiert wurden. Auch wenn die molekularen Fingerprintmethoden die Pharmacophor-basierten derzeit übertreffen, scheint es viel Potential für deren Verbesserung zu geben. Der vorgestellte Workflow könnte LigandScout Nutzern ein zusätzliches Werkzeug für das Durchsuchen von Moleküldatenbanken und die Verfeinerung von Hitlisten bieten.

Abbreviations

3D	Three Dimensional
ADMET	Absorption, Distribution, Metabolism, Excretion, and Toxicity
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AR	Aromatic
AUC	Area Under Curve
AVE	Asymmetric Validation Embedding
CADD	Computer Aided Drug Design
DUD	Directory of Useful Decoys
DUD-E	Database of Useful Decoys: Extended
ECFP	Extended-Connectivity Fingerprint
FCFP	Functional-Class Fingerprint
FN	False Negative
FP	False Positive
GNN	Graph Neural Network
GUI	Graphical User Interface
H	Hydrophobic
HBA	Hydrogen Bond Acceptor
HBD	Hydrogen Bond Donor
HTS	High-Throughput Screening
IUPAC	International Union of Pure and Applied Chemistry
JVM	Java Virtual Machine
kNN	k-Nearest Neighbours
MACCS	Molecular Access System

MAE	Mean Absolute Error
MCC	Matthew's Correlation Coefficient
ML	Machine Learning
MOE	Molecular Operating Environment
MSE	Mean Squared Error
MUV	Maximum Unbiased Validation
NI	Negative Ionizable
PCBA	PubChem BioAssay
PDB	Protein Data Bank
PI	Positive Ionizable
PRC	Precision Recall
QPhAR	Quantitative Pharmacophor-Activity Relationship
QSAR	Quantitative Structure-Activity Relationship
R2	Coefficient of Determination
RBF	Radial Basis Function
RDF	Radial Distribution Function
RF	Random Forest
RMSE	Root Mean Squared Error
ROC	Receiver Operating Curve
SMILES	Simplified Molecular Input Line Entry Specification
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UML	Unified Modeling Language
vHTS	Virtual High-Throughput Screening