Fachhochschul-Bachelorstudiengang
MEDIZIN- UND BIOINFORMATIK
A-4232 Hagenberg, Austria

Predicting the activity of protein-ligand complexes

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science in Engineering

Eingereicht von

Lukas Fallmann

Begutachtet von Micha Birklbauer, M.Sc.

Hagenberg, Mai 2024

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

This printed thesis is identical with the electronic version submitted.

31.05.2024

Date

Signature

Contents

Abstract

The process of discovering drugs or any chemically active compounds is time-consuming and therefore also very expensive. For this reason it is of general interest to improve the efficiency of drug discovery and reduce the overall cost of this process. The identification of new protein targets or leads can be achieved using an *in-vitro* approach called high-throughput screening or a computational approach called virtual screening. This thesis is based on "Automatic identification of important interactions and interaction-frequency-based scoring in protein-ligand complexes" by Micha Birklbauer[1] and proposes a novel virtual screening approach to protein docking for drug design using various machine learning methods.
For this purpose the algorithms *k nearest neighbor*, *random forest*, and an artificial *neural network* approach have been implemented. The underlying data for the algorithm consists of protein-ligand complexes of five unique proteins sourced from molecules of interest and the *DUD-E* database[2]. To extract the interaction data the *PLIP Algorithm*[3] was used. In addition to the machine learning algorithms various feature engineering methods also have been implemented and tested.
When comparing the performance of the three algorithms across a multitude of feature engineering techniques the random forest classifier performs best with respect to all the metrics implemented. The implemented algorithms are able to achieve more accurate classification results than [1]. The results computed by the algorithmic approaches proposed in this thesis are capable of producing 5-10% more accurate results. Furthermore, the algorithms also where able to detect less false positives. This is a very important gain, as there is less time spent investigating false leads in the laboratory.
Machine learning definitely works for virtually screening protein ligand databases based on interaction data. This thesis serves as a proof of concept for its applicability. The performance of the mentioned methods may even be improved by using more problem-tailored algorithms. The source code for this thesis can be found on

https://github.com/michabirklbauer/activity_prediction.

Kurzfassung

Der Prozess der Entdeckung von Medikamenten oder chemisch aktiven Verbindungen ist zeitaufwendig und daher auch sehr teuer. Aus diesem Grund ist es von allgemeinem Interesse, die Effizienz der Medikamentenforschung zu verbessern und die Gesamtkosten dieses Prozesses zu senken. Die Identifizierung neuer Medikamente oder Wirkstoffkandidaten kann mit einem *in-vitro* Ansatz, dem sogenannten *high-throughput screening*, oder mit einem computergestützten Ansatz, dem *virtual screening*, erreicht werden. Diese Arbeit basiert auf „Automatic identification of important interaction- sand interaction-frequency-based scoring in protein-ligand complexes“ von Micha Birklbauer[1] und schlägt einen neuartigen *virtual screening*-Ansatz zur Evaluierung von Protein-Docking im Medikamentendesign unter Verwendung verschiedener Machine-Learning-Methoden vor.

Zu diesem Zweck wurden die Algorithmen *k nearest neighbor*, *random forest* und ein künstliches neuronales Netzwerk implementiert. Die zugrunde liegenden Daten für die Algorithmen bestehen aus Protein-Liganden-Komplexen von fünf medizinisch relevanten Proteinen mit Wirkstoffen und Molekülen aus der *DUD-E*-Datenbank[2]. Zur Extraktion der Interaktionsdaten wurde der *PLIP*-Algorithmus[3] verwendet. Zusätzlich zu den Machine-Learning-Algorithmen wurden auch verschiedene Feature-Engineering-Methoden implementiert.

Beim Vergleich der Leistung der drei Algorithmen unter der Verwendung verschiedener Feature-Engineering-Techniken schneidet der Random-Forest-Klassifikator in Bezug auf alle implementierten Metriken am besten ab. Die implementierten Algorithmen können genauere Klassifikationsergebnisse erzielen als [1]. Die Ergebnisse der Machine-Learning-Algorithmen, welche im Rahmen dieser Arbeit entwickelt wurden, liefern im Durchschnitt um 5-10% genauere Ergebnisse. Darüber hinaus erkennen die Algorithmen auch weniger falsch positive Proteinkomplexe. Dies ist ein entscheidender Vorteil, da weniger Zeit mit der Untersuchung inaktiver Protein-Ligand Komplexe im Labor verbracht wird.

Diese Arbeit zeigt die Anwendbarkeit von Machine Learning für das interaktionsbasierte virtuelle Screening von Protein-Liganden-Komplexen. Die Qualität der genannten Methoden kann eventuell durch das weitere Anpassen der Algorithmen noch verbessert werden.

Der Quellcode, welcher für diese Arbeit implementiert wurde, ist auf https://github.com/michabirklbauer/activity_prediction zu finden.

Acronyms

ACC Accuracy. 15, 17, 34
AcH Acetylcholine. 5
AChE Acetylcholinesterase. iv, 5, 17, 23, 33
AUC Area under the curve. 16

BCE binary cross-entropy. 11

COX1 Cyclooxygenase 1. 5
COX2 Cyclooxygenase 2. 5

DPP4 Dipeptidyl peptidase IV. 5

EF enrichment factor. 16

FPR False positive Rate. 15

HTS high-throughput screening. 1

KNN K nearest neighbor. 8, 18, 30, 31

LBVS ligand basedvirtual screening. 1

MAOB Monoamine oxidase B. 6
MDI Mean Decrease in Impurity. 12

NN neural networks. 9

PCA Principal component analysis. iii, 14
PLIP protein ligand interaction profiler. 2

REF relative enrichment factor. 16, 34
ReLU rectified linear unit. 12

SMOTE Synthetic Minority Over-sampling Technique. 14, 17, 19, 33
SVD singular value decomposition. 14

VS virtual screening. 1

Ya Yield of actives. 16

Chapter 1

Introduction

The discovery of new drugs or any chemically active compounds for that matter is an expensive and time-consuming process. It has been estimated, that it takes about 14 Years from the initial discovery of a promising new compound to the release of a marketable drug[4]. In addition to that the price of this drug-discovery circle ranges up to 800 Million Dollars[5]. All techniques which aim to improve the efficiency of drug discovery can generally be categorized as one of two methods. These two are called high-throughput screening (HTS) and virtual screening (VS)[4].
When using an HTS-approach there are many compounds which are tested against some type of target protein. Target proteins are usually proteins which are of general interest for medical use. During testing, it is measured whether a certain compound biochemically interacts with a protein. Those interacting combinations are considered active and are marked by researchers as hits. To improve the performance of HTS there are a number of factors to consider. Through miniaturization, it is possible to investigate more compounds at the same time. With a higher throughput quality-control is more time-consuming and leads to an overall more expensive process. For this reason HTS is most efficient, when analyzing a small set of compounds as the technology is not suitable for large datasets[6].
In contrast to the in vitro approach of HTS, VS is a theoretical in silico approach. To save resources in the laboratory the activity of certain compounds is predicted using a preexisting library of small molecules. The activity can be predicted using the ligands of a compound and their respective binding sites or the 3D structure of a compound. The key idea behind the ligand based approach (LBVS) is that similar compounds have similar chemical properties. Therefore, the goal of LBVS is to find molecules which have similar or identical chemical properties as the sample compound[7]. Structure-based VS uses the 3D structure of a compound to predict which molecules from the dataset will bind to the provided sample. Each molecule of a certain database subset is fitted (docked) to the sample. Hereby it is important to differentiate between rigid and flexible docking[8].
In rigid docking the dataset sample is rotated and translated in a six-dimensional space in order to fit the sample protein. For each fitted molecule a score is calculated based on how well the molecule fits to the sample[9]. Although this algorithm often predicts actual possible binding sites and bound proteins, there is no guarantee that this compound will actually bind in vitro. Therefore, predicted interactions should be seen as a hypothesis.

Still, rigid docking provides a great baseline at a comparatively low cost[7]. The low accuracy of rigid docking is due to the nature of biochemical substances as samples in a database can only provide a snapshot of a sample[8]. With flexible docking it is possible to simulate moving binding sites, where the flexibility can be introduced at different stages. Implicit flexibility is achieved by smoothing protein surfaces and therefore allowing room for interpretation when docking. Cross- or Ensemble docking can be done by repeating the docking process with different conformation and explicit flexibility is reached through allowing side-chain flexibility. Most commonly utilized is the approach where the ligand is flexible, and the receptor is rigid. Even though this approach does provide better and more accurate results it takes considerably longer to compute[8].
Regardless of the docking type the score should reflect which pose between a protein and a ligand is most likely to exist. In addition to that, the score also should reflect whether a protein-ligand complex is considered active. There are a lot of different scoring functions which can be grouped into four categories: physics-based, empirical, knowledge-based, and machine learning-based[10].
The focus of this work is on implementing a machine-learning based scoring approach. Machine-learning based scoring functions work by training on labeled data and finding the best model for predicting future data. To accurately and efficiently train a model crucial binding sites need to be identified beforehand. The basis of this thesis is the master thesis of Birklbauer Micha[1]. In his thesis a selection of eleven proteins from the directory of useful decoys[2] have been selected to be analyzed. For the selected proteins all possible interactions have been analyzed by *PLIP*, which is an algorithm designed to discover various interactions based on the physical properties of a compound[3]. Based on the interaction-data a few basic scoring functions have been implemented. The direct result of that thesis are protein-ligand complexes and the frequency of their interactions. Since this work aims to implement different machine learning algorithms for use in drug discovery the state of the art is described in the following.

1.1 Machine Learning in drug design and activity prediction

The following chapter summarizes the recent developments in drug design using various machine learning techniques.

Today there exist a multitude of machine learning approaches in the field of drug design and activity prediction. As a result of various AI breakthroughs in recent years there have been numerous research projects regarding the usability of artificial intelligence in various bioinformatic domains. One area where machine learning can be applied is quality assessment. *SVMQA* utilizes support vector machines to assess the quality of structural protein models. The algorithm works by constructing a feature vector for each prediction based on physical and statistical properties. Based on this score the algorithm predicts a numerous quality-assessment scores[11]. Support Vector machines have also been used for a *DeNovo* algorithm to detect protein-virus interactions. The goal of the *DeNovo* implementation is to identify protein-protein interactions without any interaction data. This is achieved by learning the primary interaction points of the host proteins[12]. AI has also been used to successfully identify drug responsive biomarkers in pre-clinical data using regression algorithms[13]. In the field of synthesis-prediction AI has largely replaced the rule- and heuristic-based systems in place since the 1960s[14].

Due to developments in the field of deep learning, this technology has found numerous applications in biochemistry[15]. One of which is *deepDTnet*, which is a deep learning based algorithm used to identify new targets and repurpose existing drugs in a drug-gene-disease environment. This is done by embedding already existing interaction profiles into low dimensional vector spaces. For two potentially interacting proteins a deep learning algorithm is used to determine whether they would interact based on their vector representations [16]. *NeuroCADR* is another approach for drug-repurposing, as drug repurposing using machine-learning is cheaper than the traditional drug discovery approaches. This paper discusses the use of random-forest and k nearest neighbor as a way to repurpose existing drugs for neurological diseases. The software developed for this paper was used to define new possible drugs for the treatment of epilepsy.[17] Deep learning also has its applications in the classification and segmentation of microscopic imagery. With the use of a combination of multiple instance learning and convolutional neural networks it is possible to classify and segment microscopic imagery simultaniously[18]. *MolDesigner* is a software, which implements a *human-in-loop* strategy. This means that a human expert is designing a drug within a web-interface and the numerous deep-learning networks provide feedback on how well the current design would work as a potential drug. The base data for the deep learning networks is sourced from state-of-the-art interaction databases[19]. *MILCDock* uses the Output of five traditional Scoring Functions as input for a neural Network. The input for the neural network comes from the tools *LeDock*[20], *Autodock Vina*[21], *PLANTS*[22], *Autodock4*[23], and *rDock*[24]. This technique has a slight performance benefit when compared to traditional scoring functions. The results are achieved by implementing a basic class balancing framework, as the majority of the data provided for the different algorithms is very unbalanced[25].

1.2 Goals

The goals of this thesis are twofold:

1. Evaluate common machine learning approaches for interaction-based activity prediction and compare results with current literature.
2. Evaluate the results posed by various feature engineering techniques and investigate the possible performance benefits for the implemented ML approaches.

The second goal can be viewed as an extension of the first one since its primary aim is to improve the results achieved while pursuing the first goal.

Chapter 2

Methods

2.1 Data description

The following chapter is dedicated to explaining the data used for this thesis. This includes detailed descriptions of the protein complexes as well as their interaction types.

2.1.1 Proteins

The following five proteins have been used as grounds for this thesis:

Acetylcholinesterase

Acetylcholinesterase (AChE) is an enzyme in the nervous system that breaks down Acetylcholine (AcH), a messaging molecule, into choline and acetate. It's found in high concentrations at junctions between nerve cells and muscles. AChE has various functions beyond just breaking down Ach, and it's present in both nerve and non-nerve tissues. Because AChE is so important, some toxins like insecticides and nerve agents target it. This versatility of AChE makes it a key player in nervous system function and a potential target for drugs to treat diseases[26].

Cyclooxygenase 1

Cyclooxygenase 1 (COX1) and its isoform Cyclooxygenase 2 (COX2) play a substantial role in synthesising various prostaglandins. Due to their linkage with inflammations and pain COX molecules are often targeted by anti-inflammatory drugs. In contrast to COX2, COX1 is found in most tissues across the body. In addition to that COX1 is largely attributed with homeostatic functions such as hemostasis and gastric cytoprotection[27].

Dipeptidyl peptidase IV

Dipeptidyl peptidase IV (DPP4) is partially responsible for hydrolysis of a prolyl bond between two residues from the N-terminus. DPP4 is present in several processes including metabolism and cancer biology. Due to its role within metabolism DPP4 inhibitory drugs have been successfully used in the treatment of diabetes type two. DPP4 also

plays a substantial role in the diagnosis of certain types of cancer. In most cases DPP4 is up regulated near cancerous growth, therefore locally elevated DPP4 levels can be an indicator for cancer[28].

Monoamine oxidase B

Monoamine oxidase B (MAOB) plays a major role in the breakdown of neurotransmitters (monoamines) within the body. The compound is mainly expressed in glial-cells and platelets. Its function categorizes MAOB as an important research compound, as MAOB inhibition has been proven to improve various neurological conditions. This stems from the fact that changes in the monoamine levels are associated with a myriad of neurological problems[29].

Soluble epoxide hydrolase

Soluble epoxide hydrolase (sEH) is part of an inflammatory pathway similar to COX. It has been shown that inhibition of sEH reduces inflammation. In contrast to COX it does not completely disable the synthesis of pro-inflammatory compounds but rather balance their levels[30].

2.1.2 Interactions

Interactions define how proteins interact with each other or other types of ligands. There are a lot of interactions which can be used for determining whether a certain compound might be considered active. The following interactions have been used by the *PLIP Algorithm*[3] to produce the base data for this thesis:

interaction	description[1]
hydrogen bonds	A hydrogen bond is defined as the interaction between a hydrogen atom, connected to a more electronegative atom, and another atom or molecule.
water bridges	A water bridge occurs when the ligand and the protein both bind to a water molecule through hydrogen bonds.
salt bridges	Salt bridges are ion pairs which stick together due to large difference in charge and the resulting electrostatic interaction.
halogen bonds	Halogen bonds are defined as the interactions between the electrophilic region around a halogen atom and a nucleophilic region.
hydrophobic interactions	Aggregates formed as a result of hydrophobicity between hydrocarbons in an aqueous medium are called hydrophobic interactions.
pi-stacking	Interactions between neighboring aromatic rings are called pi-stacking. Due to the pi-electron density the ring is partially positively charged around the periphery and negatively charged above both aromatic faces. As a result electrostatic forces build between aromatic rings, and they are attracted to one another.
pi-cation	Cations and pi-stacks who bind through electrostatic forces at a pi-stacks face are called pi-cation interactions.

Table 2.1: interaction types

2.1.3 Data origin and structure

The provided data is a byproduct of the thesis [1] by Micha Birklbauer. The interaction data was produced using the *PLIP Algorithm* [3] on the aforementioned proteins.

The PLIP Algorithm consists of four major stages:

Structural Preparation – SP
: During the preparation step the input structure is hydrogenated and the ligands(including their binding sites) are extracted.

Functional Characterization – FC
: Using the structure of the complex a myriad of functional groups are detected. This includes binding site atoms, hydrophobic atoms and aromatic rings just to name a few.

Rule Based Matching – RBM
: In the third step the algorithm investigates all interactions between the ligand and the protein, which can be attributed to geometric constraints. For example, hydrogen bonds are detected here.

Filtering of Interactions – FoI
: This is a cleanup step where redundant or overlapping interactions get removed from the dataset.

The result of the PLIP Algorithm is a lineup of every interaction for each binding site and ligand[3]. This data has been used as a basis for the machine learning approaches discussed in this thesis.

2.2 Data partitioning

To validate the results of training various machine learning methods the provided data concerning the five targets was split into a training-set as well as a test-set. To achieve a 70/30 train/test split ratio each sample was randomly assigned to one of the two data partitions[31].

In order to validate the machine learning approaches during training 10-fold cross-validation has been applied. For the process of cross-validation the training dataset is split into n equally large subsets. The type of cross-validation implemented in this thesis uses all but one of these partitions to train the classification model and validates the results with the remaining partition. This process is repeated for all possible validation partitions[32].

2.3 Machine-Learning approaches

The following chapter aims to explain the basic machine learning approaches used for this thesis.

2.3.1 K nearest neighbor

K nearest neighbor (KNN) is a very simple classification algorithm for the numerical features used in this thesis. First, the source dataset is converted into a vector representation. Each sample of the source dataset is represented as and n-dimensional vector, where n is the amount of features within that particular dataset. In addition to that vector the class of each sample is also saved. The classification is achieved by representing a new sample within the aforementioned vector-space. For the new sample the k nearest neighbors are calculated using a distance metric, where k is an odd-number. For this thesis the *euclidean* distance was used. The euclidean distance for two vectors$(\mathrm{v}_1, \mathrm{v}_1)$ in the n dimensional space is defined as follows:

$$euclidian(\mathrm{v}_1, \mathrm{v}_1) = \sqrt{\sum_{i=1}^{n} (\mathrm{v}_{1,i} - \mathrm{v}_{2,i})^2}$$

In the base version of KNN the new sample is put in the same class as the majority of its k-neighbors. In addition to that the k nearest neighbors can also be weighted using their distance to the new sample[33].

K nearest neighbor for this thesis has been implemented using the scikit-learn package.

2.3.2 Random forest

In the following the base concept of the random forest algorithm used for this thesis will be explained.
The random forest algorithm is a collection of identically distributed decision trees[34]. The algorithm can be split into the creation of a bootstrapped dataset, the creation of decision trees and the evaluation of said trees.

Bootstrapping

The goal of this step is to create a new dataset for each decision tree. This happens by randomly selecting samples from the source dataset. It is noteworthy that samples can be selected multiple times when creating such a bootstrapped dataset. Furthermore, not all samples are included in each dataset. Those samples, which are not included in any dataset are called *out-of-bag* samples and are later used for further tree enhancement. With those bootstrapped dataset decision trees can be built in the next step.

Creation of decision trees

For each bootstrapped dataset a new decision tree is created using the following steps. Firstly a number of features is randomly selected. For those selected features it is determined, which feature is best for splitting the data, so that the classes are separated very clearly. This is usually determined using the *Gini*-impurity. The Gini-impurity is a measure of how well a dataset can be divided using a certain feature.

The Gini-impurity can be calculated at each node of a decision-tree and is ranged from 0 to 0.5. Let k be the number of classes and let p_i be the probability of a sample belonging to the class(i) and the Gini-impurity($Gini(D)$) of the dataset(D) at a certain node within the tree can be defined as follows:

$$Gini(D) = 1 - \sum_{i=1}^{k} \mathrm{p}_i^2$$

[35]

After selecting the splitting feature the data is split and the whole process repeats for the newly created nodes. When selecting a new splitting feature all features already existing in the tree can be used again. This whole process is repeated until every path of the tree leads to a clear classification result[34].

Evaluation and optimization

After the construction of the forest the performance can be evaluated using the out-of-bag samples. This is achieved by classifying the samples from the out-of-bag dataset using the random-forest. Each tree within the forest is provided with the data from the sample and votes for a class based on the samples features.

The parameters of a random forest, for example the number of features considered at each node, can be optimized. To achieve that multiple forests with different parameters are generated and the best performing one for the out-of-bag samples is selected[34].

For this paper the random forest approach has been implemented using the scikit-learn package.

2.3.3 Neural networks

The idea to emulate neural system using computers was first discussed in the 1940s by McCulloch and Pitts[36]. Since then the technology around neural networks has improved greatly. The following chapter describes the basic concepts of neural networks and their implementation for this thesis.

The goal of neural networks in general is to process input data through an emulated brain-like structure for a multitude of machine learning tasks. Neural networks are universally appliable algorithms due to their ability to resemble any given function through combining multiple smaller functions. Furthermore, neural networks can be computed using matrix calculations, which are typically carried out on the graphics card of a system, as it is optimized for such operations[37]. In this thesis neural networks are used to determine the activity of protein-ligand interactions using detailed interaction data.

Terminology

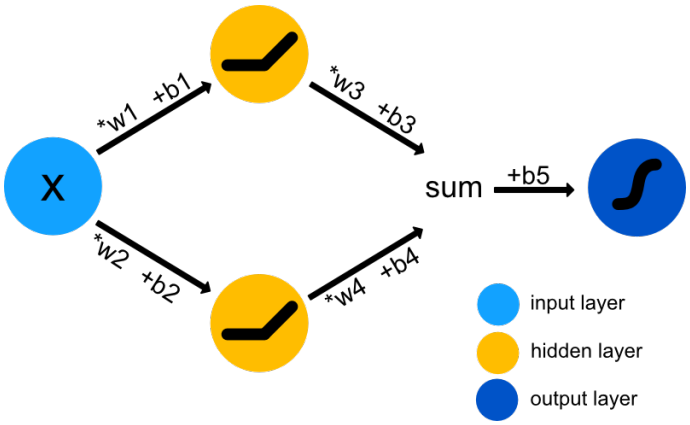
A neural network consists of *neurons* and *synapses*. Neurons are nodes within the neural network that can take one or multiple inputs and transform them into one singular output. The neurons are connected through synapses which can further manipulate values given to them by applying *weights* and *biases*[38].

A very simple neural network consists of an input neuron layer, where each neuron corresponds to a feature, a singular output neuron for binary classification. The neurons of the input layer are connected to the output neuron through synapses.

Classification using neural networks

This section is dedicated to demonstrating the classification of a sample within already existing neural network with one hidden layer. In a first step the features of the sample are mapped to the input neurons. After that the values are transferred to the hidden layer through the respective synapse. During this process the weight for that synapse is applied (this is usually done through a multiplication). Before the feature gets to the neuron of hidden layer the bias of that synapse is applied to the feature through an addition. Inside the neuron the value is modified using an activation function. Example for activation functions can be found in the *activation functions* section within this chapter. All the modified values from the hidden layer are transported through synapses to the output neuron which calculates the class using an activation function[38].

Figure 2.1: neural network representation



Back propagation and gradient descent

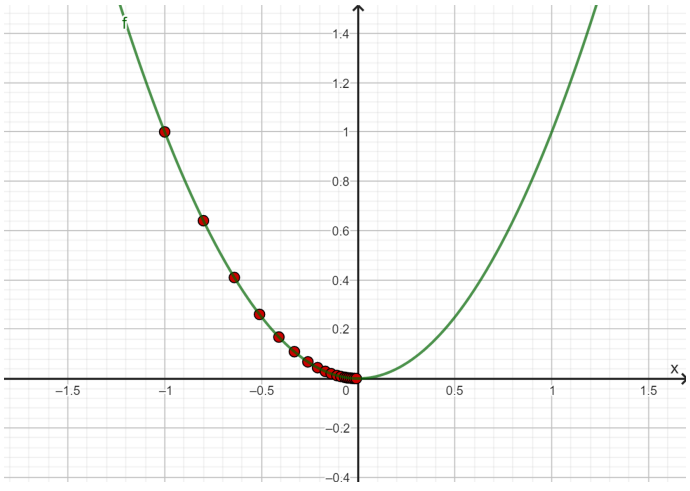
While the activation functions for the neural network can be set as parameters when creating the network the weights and biases are optimized during training. To assess the performance of a set of weights and biases a loss function is used. The loss function determines how well the predicted values compare to the actual values[39]. The *binary cross-entropy (BCE)* is an example for such a loss function and is defined for n samples as follows:

$$BCE = \frac{1}{N}\sum_{i=1}^{N} target_i * log(predicted_i) + (1 - target_i) * log(1 - predicted_i)$$

[40]

With a loss function it is possible to optimize values using gradient descent. This is achieved by calculating the derivative of the loss function with respect to the parameter. Initially the parameter will be initialized with a random value. After that the train set will be evaluated using this particular parameter. Then the derived loss function is applied to the values. Using a fixed *learning rate* the *step size* for gradient descent is calculated by multiplying the learning rate with the result from the derivative loss function. For the next iteration of gradient descent the step size will be subtracted from the value to get a new optimization value. Using this technique not every possible value of the variable needs to be examined. With a smaller proximity to the minimum the step size decreases [41]. The following graph displays a loss function with the points demonstrating the variable values investigated by gradient descent.

Figure 2.2: gradient descent representation



Using this concept the *back propagation* algorithm optimizes all the weights and biases within a neural network all at once. The loss function is derived for each bias and weight. After that the resulting functions are applied to the values and new weights and biases are calculated using the step size. This process is repeated until a certain

threshold for the overall loss is reached or the number of iterations has reached its maximum[42].

Activation functions

Activation functions play a major role in the performance of neural networks. They are specified when creating a new model. For this thesis the *Sigmoid* activation function was used for the output-layer. The sigmoid function is defined as follows:

$$\sigma(x) = \frac{e^x}{1 + e^x}$$

[43] The sigmoid function is very applicable for binary classification, as it provides values between 0 and 1 which can be interpreted as probabilities[44]. The *rectified linear unit (ReLU)* activation function has been proven to improve performance when used in hidden layers[45]. Therefore, this function is used for the hidden layers within the neural network implementation. ReLU is defined as follows:

$$f(x) = max(0, x) \tag{2.1}$$

[45]

Different activation functions work for different datasets, it is therefore crucial to determine which activation function works best for the provided data. Other activation functions include *Tanh* or APL[45].

The practical neural network approach of this thesis has been implemented using the Tensorflow[46] library. The model definitions where optimized manually using the validation accuracy.

2.4 Feature engineering

Feature engineering is the process of manipulating the given features within a dataset with the goal of improving the performance of numerous machine learning techniques applied to the manipulated data. The following chapter explains the various methods that have been used for this thesis.

2.4.1 Feature engineering using random forest

Due to the nature of the random forest algorithm, explained in 2.3.2, it can be used effectively to determine the most important features within a dataset. This section introduces the two feature engineering components within this thesis that are based on the random forest algorithm.

Mean Decrease in Impurity (MDI)

The importance of each feature in a random forest is decided by how well a certain feature can divide the samples in to the desired groups. The mean decrease in impurity is a measure designed to indicate this importance. To calculate it the *Gini*-impurity mentioned in 2.3.2 is needed.

With that in mind the mean decrease in impurity can be calculated with the following steps. At first the initial Gini-impurity needs to be calculated using the formula from 2.3.2 with the classes *active* and *inactive*. In a second step it is necessary to calculate the *weighted Gini* after a split by a feature for each feature. This is achieved by multiplying the relative amount of actives with the Gini-impurity of a given feature for an active classification. This is repeated for the inactive component. Those two numbers combined equate the weighted Gini for that feature. The average of the weighted Gini over all features equates to the mean decrease in impurity. Let g_f be the weighted Gini for a feature(f) and n be the number of features then the mean decrease in impurity(mdi) can be defined as follows:

$$mdi = \frac{1}{n} * \sum_{f=1}^{n} \mathrm{g}_f$$

With this metric the features with larger g_f are deemed the most crucial for classification[47].

In the following, results with the *fe_rf_mdi*-prefix where calculated using this method.

Permutation importance

Permutation importance is a feature engineering technique used to determine the most important features for classification within a tabular dataset. At first a reference score(s) is calculated using a random forest classifier. In the following step a feature column is randomly permutated. After this *corruption* of the source dataset the score is calculated again and compared to the reference score. This step can be repeated K times in order to improve its statistical viability. This process is repeated for all features.

Let j be the feature, K the repetitions per feature and $\mathrm{s}_{k,j}$ the score of each corrupted dataset, and the importance of each feature(i_j) can be calculated as follows:

$$\mathrm{i}_j = s - \frac{1}{K} * \sum_{k=1}^{K} \mathrm{s}_{k,j}$$

If a feature is of greater significance to the model then the score will deviate greater from the reference value[48].

This method is not particularly dependent on the random forest algorithm. The random forest classifier component can be substituted with any other classifier.

In the following, results with the *fe_rf_per*-prefix where calculated using this method.

2.4.2 Physical properties

Feature engineering can also be based on meta-information concerning the provided datasets. For this thesis two methods are proposed to enhance the data by removing possible *noise-features* with the use of additional knowledge concerning the datasets.

Selection of most frequent interactions

To prevent overfitting the forty features(binding-sites) with the most interactions have been selected. By removing the less occurring features the overall performance especially

on the validation- and test-runs should improve. Due to the reduction in the amount of features overfitting can be reduced, and the machine learning models are less distracted by "unimportant" features.

In the following, results with the *fe_freq*-prefix where calculated using this method.

Removal of all hydrophobic interactions

Of all, for this thesis considered interactions, hydrophobic interactions are generally the most frequent but also the weakest[49]. Therefore, it is of interest to reduce the overall amount of features by removing all the hydrophobic interactions from the datasets in order to achieve more granular results.

In the following, results with the *fe_nonhydrop*-prefix where calculated using this method.

2.4.3 Principal component analysis (PCA)

Principal component analysis is a feature engineering method which aims to reduce the noise within a dataset, as well as maximize the amount of variance. PCA works by representing the original dataset as through linear uncorrelated variables or components. This is done in three steps:

1. Restructuring of the data so that the data is represented as a $m \times n$ matrix where m is the number of features and n the number of samples.
2. Subtract off the mean for each feature.
3. Calculation of the principal components using singular value decomposition (SVD).

While the first two steps are quite clear, the third step will be explained in the following.

First the SVD of the dataset needs to be defined.

$$X = U \cdot \Sigma \cdot V^T$$

Where X is the original data matrix, U is the matrix containing the eigenvectors of $X \cdot X^T$, Σ contains the square-roots of the eigenvectors of $X^T \cdot X$ and V contains the eigenvectors of $X^T \cdot X$.

To get the transformed data it is necessary to multiply the U matrix with Σ. The resulting projections are sorted according to variance[50].

In the following, results with the *fe_pca*-prefix where calculated using this method.

2.4.4 Balancing classes

The data used for this thesis is not balanced, as there are more *inactive* samples than *actives*.

Synthetic minority over-sampling(SMOTE) can be used to balance the provided datasets. The aim of this technique is to synthetically generate samples from the minority class to balance the class distribution. The SMOTE algorithm starts by selecting a sample from the minority class and finding its k nearest neighbors within that class. For each of the selected neighbors the difference to the original sample is calculated. The differences are then scaled with a random factor between 0 and 1. Those scaled values

are added to the original sample in order to create new samples. This whole process is repeated for the unbalanced dataset until all classes are equally represented[51].

In the following, results with the *fe_smote*-prefix where calculated using this method.

2.5 Quality metrics

In order to make the results from this thesis comparable to the results from the scoring function introduced in [1] by Micha Birklbauer the same quality metrics have been implemented for this thesis. The following will provide an overview for the used metrics.

2.5.1 Terminology

To calculate the metrics that are mentioned within this chapter the following base terms are necessary:

TP – **T**rue **P**ositives are active samples, which are classified as such

TN – **T**rue **N**egatives are inactive samples, which are classified as such

FP – **F**alse **P**ositives are inactive samples, which are classified as active

FN – **F**alse **N**egatives are active samples, which are classified as inactive

2.5.2 Visual metrics

For better visualization of the four base metrics mentioned in 2.4.1 this thesis displays the resulting data in a confusion matrix. This metric displays distribution of the results over the four base metrics.

In addition to that, the ROC(receiver operating characteristic) curve will also be displayed for the results. The ROC curve is a collection of points in a two-dimensional space, where their location is defined by the FPR 2.4.4 on the x-axis and the TPR($\frac{\#TP}{\#TP+\#FN}$) on the y-axis. Each point on this line depicts the ratio of FPR to TPR at a certain score cutoff. [52]

2.5.3 Accuracy

Accuracy (ACC) describes which portion of the predicted samples was accurately assigned to the correct class and is defined as follows:

$$\text{ACC} = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN}$$

[53]

2.5.4 False positive Rate

False positive Rate (FPR) describes the compounds that were incorrectly classified as active in relation to all inactive compounds and is defined as follows:

$$\text{FPR} = \frac{\#FP}{\#TN + \#FP}$$

[52]

2.5.5 Area under the curve

Area under the curve (AUC) is a metric which stems from the ROC curve. The integral of the ROC curve is calculated using the scikit-learn package and is always between 0 and 1(higher is better)[52].

2.5.6 Yield of Actives

Yield of actives (Ya) describes the true positive compounds in relation to all as active labeled compounds and is defined as follows:

$$\text{Ya} = \frac{\#TP}{\#TP + \#FP}$$

[54]

2.5.7 Enrichment Factor

The enrichment factor (EF) describes the relation of the truly active compounds among all as active predicted complexes and the relative share of active compounds in the dataset. This metric is defined as follows:

$$\text{EF} = \frac{\frac{\#TP}{\#TP+\#FP}}{\frac{\#TP+\#FN}{\#TP+\#TN+\#FP+\#FN}}$$

[52]

2.5.8 Relative Enrichment Factor

The relative enrichment factor (REF) describes the relation of the EF to the maximum achievable EF. The REF is defined as follows:

$$\text{REF} = \frac{100 * \#TP}{\min(\#TP + \#FP, \#TP + \#FN)}$$

[52]

Chapter 3

Results

The following chapter describes the results from the different machine learning approaches and the applied feature engineering techniques described in Methods.

3.1 Feature Engineering Results for AChE

To evaluate different feature engineering approaches the protein *Acetylcholinesterase* was used. The goal of this evaluation is to determine which feature engineering techniques shall be used on the protein-ligand compounds. It is also of interest, which feature engineering techniques work best with each machine learning approach. The ACC measure is used to score the performance of the feature engineering method. The metric is calculated using the validation data.

Neural network

The following table is the result of applying the neural network on the datasets that where manipulated using feature engineering.

Table 3.1: Feature engineering validation accuracy neural network

Name	Validation Accuracy
baseline_nn	0.7801
fe_smote_nn	0.7801
fe_pca_nn	0.7589
fe_rf_mdi_nn	0.7589
fe_rf_per_nn	0.7518
fe_nonhydrop_nn	0.7376
fe_freq_nn	0.7180

The results state that the neural network approach does not improve when applying the proposed feature engineering methods. The SMOTE method comes close to the

performance of the baseline neural network. Therefore, it will be included for the analysis of the five complexes.

K nearest neighbor

The following table is the result of applying the KNN algorithm on the datasets that where manipulated using feature engineering.

Table 3.2: Feature engineering validation accuracy
k nearest neighbor

Name	Validation Accuracy
fe_rf_mdi_knn	0.7934
fe_rf_per_knn	0.7778
fe_freq_knn	0.7664
fe_nonhydrop_knn	0.7550
fe_pca_knn	0.7550
baseline_knn	0.7437
fe_smote_knn	0.7437

The KNN algorithm benefits greatly from the proposed feature engineering methods. To contrast the baseline KNN performance best, the feature engineering methods using random forest will be evaluated for the protein-ligand complexes, as their performance supersedes the other methods.

Random forest

The following table is the result of applying the random forest algorithm on the datasets that where manipulated using feature engineering.

Table 3.3: Feature engineering validation accuracy
random forest

Name	Validation Accuracy
fe_smote_rf	0.8375
baseline_rf	0.8362
fe_rf_mdi_rf	0.8290
fe_rf_per_rf	0.8221
fe_freq_rf	0.8107
fe_nonhydrop_rf	0.8050
fe_pca_rf	0.8005

The random forest algorithm does not benefit greatly from the proposed feature

engineering methods. Due to the more balanced dataset resulting from the SMOTE method a slight performance boost can be observed. Therefore, the SMOTE algorithm will be applied to the remaining protein-ligand complexes.

Overall performance projections

The following table lists all the feature engineering accuracies for all the machine learning approaches.

Table 3.4: Feature Engineering Validation Accuracy overall

Name	Validation Accuracy
fe_smote_rf	0.8375
baseline_rf	0.8362
fe_rf_mdi_rf	0.8290
fe_rf_per_rf	0.8221
fe_freq_rf	0.8107
fe_nonhydrop_rf	0.8050
fe_pca_rf	0.8005
fe_rf_mdi_knn	0.7934
baseline_nn	0.7801
fe_smote_nn	0.7801
fe_rf_per_knn	0.7778
fe_freq_knn	0.7664
fe_pca_nn	0.7589
fe_rf_mdi_nn	0.7589
fe_nonhydrop_knn	0.7550
fe_pca_knn	0.7550
fe_rf_per_nn	0.7518
baseline_knn	0.7437
fe_smote_knn	0.7437
fe_nonhydrop_nn	0.7376
fe_freq_nn	0.7180

As seen in the table the random forest approaches tend to yield the best result when applied to the validation portions of the datasets. Therefore, this approach is very likely to score better on the test sets as well.

3.2 Performance per Protein-Complex

The following chapter is dedicated to evaluate the different machine learning and feature engineering methods on the target compounds. For each protein the top two approaches are evaluated further.

3.2.1 Acetylcholinesterase

The following table presents the results of the different machine learning algorithms on the various test-sets. The ROC curves for the top two performing configurations can be found at 3.3 and 3.4 respectively. The confusion matrices can be found at 3.1 and 3.2. The scoring functions achieved an accuracy score of 81.06% on the test-set. Additionally, the enrichment factor is high which indicates good performance.

Table 3.5: Acetylcholinesterase performance test-set

Name	ACC	FPR	AUC	YA	EF	REF
baseline_rf	0.8106	0.3285	0.7992	0.7716	1.4161	92.6829
fe_smote_rf	0.8007	0.3358	0.7894	0.7653	1.4046	91.4634
fe_smote_nn	0.7708	0.2993	0.7650	0.7684	1.4102	82.9268
baseline_nn	0.7674	0.2920	0.7626	0.7701	1.4134	81.7073
fe_rf_per_knn	0.7575	0.4307	0.7420	0.7177	1.3172	91.4634
baseline_knn	0.6844	0.5766	0.6629	0.6520	1.1966	90.2439
fe_rf_mdi_knn	0.5515	0.4307	0.5530	0.5986	1.0987	59.8639

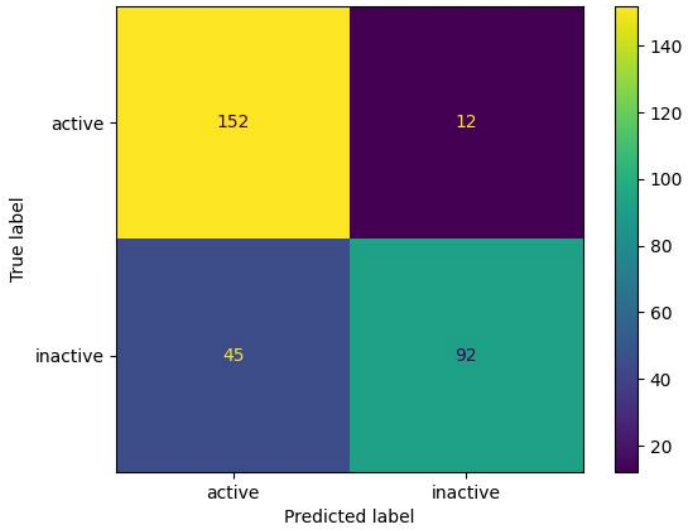


Figure 3.1: Baseline random forest confusion matrix
The confusion matrix for the baseline random forest model shows that it performs relatively well as the diagonal values(152, 92) are proportionally high when compared to the rest of the values. The model is more likely to predict a false positive than it is predicting a false negative.

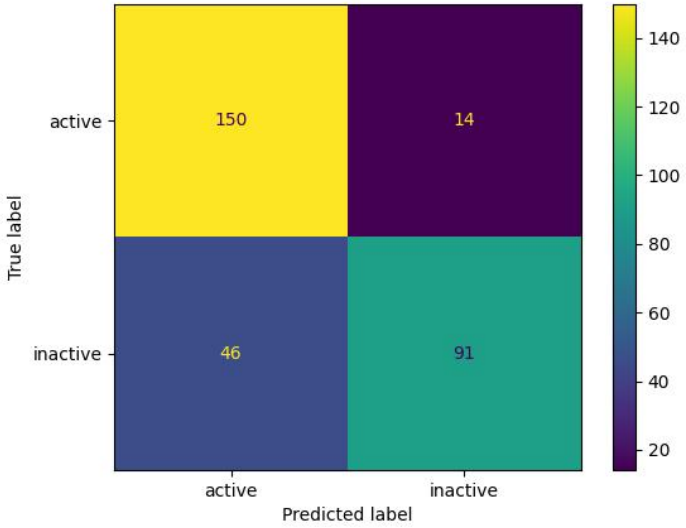


Figure 3.2: SMOTE random forest confusion matrix
The matrix shows that the model is a fractionally less likely to predict a false negative when compared to 3.1. However, the accuracy is marginally worse.

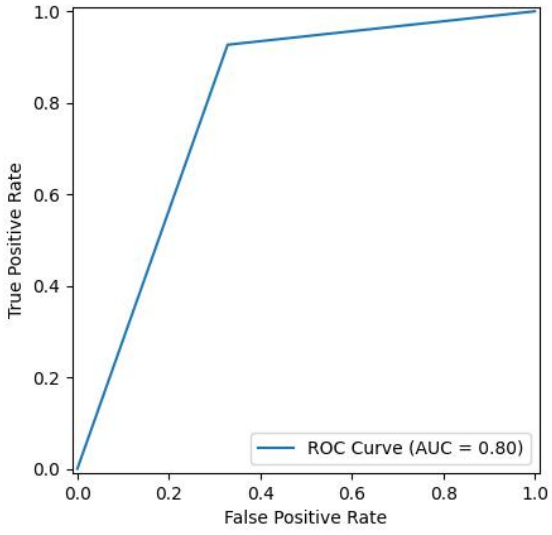


Figure 3.3: Baseline random forest ROC curve
The ROC curve leans towards the top left corner, which indicates a good performance. It is closer to a perfect 1.0 than it is to the random threshold of 0.5.

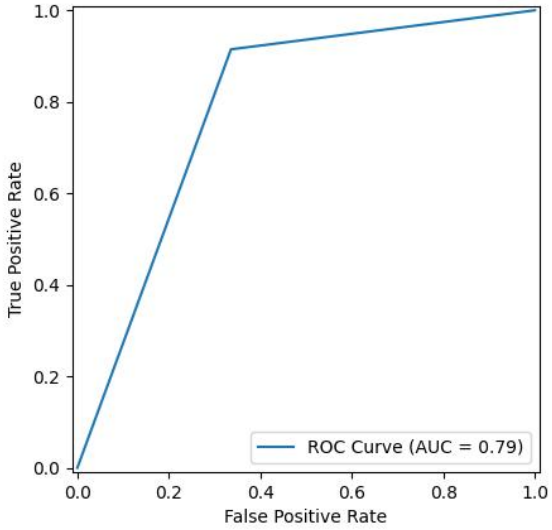


Figure 3.4: SMOTE random forest ROC curve
The appearance of the ROC curve signifies a good discriminative performance, although the AUC score of 0.79 indicates that the performance is not as good as 3.3.

3.2.2 Cyclooxygenase 1

This table summarizes the performance of various machine learning algorithms on different test sets. The top two performing configurations are visualized in ROC curves (see Figures 3.7 and 3.8) and confusion matrices (see Figures 3.5 and 3.6). Additionally, the scoring functions achieved an accuracy of 77.24% on the test set. The models where able to achieve a low FPR of 1.8%.

Table 3.6: Cyclooxygenase 1 performance test-set

Name	ACC	FPR	AUC	YA	EF	REF
baseline_rf	0.7724	0.0183	0.6344	0.8710	2.8909	87.0968
fe_smote_rf	0.7628	0.0138	0.6155	0.8846	2.9362	88.4615
fe_rf_per_knn	0.7019	0.0826	0.5598	0.5135	1.7044	51.3514
baseline_knn	0.6859	0.1147	0.5544	0.4565	1.5153	45.6522
baseline_nn	0.6827	0.0872	0.5309	0.4242	1.4081	42.4242
fe_smote_nn	0.6827	0.1147	0.5490	0.4444	1.4752	44.4444
fe_rf_mdi_knn	0.6250	0.1835	0.4987	0.2982	0.9899	29.8246

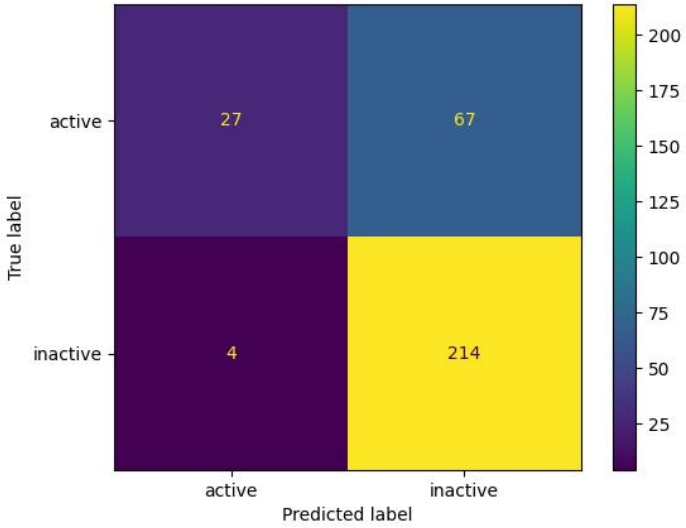


Figure 3.5: Baseline random forest confusion matrix
This test-set contains more inactives than the test-set of the AChE protein. This can be seen as the number of true negatives is far greater. Additionally, the number of false-positives is really low, which is an important metric for drug design.

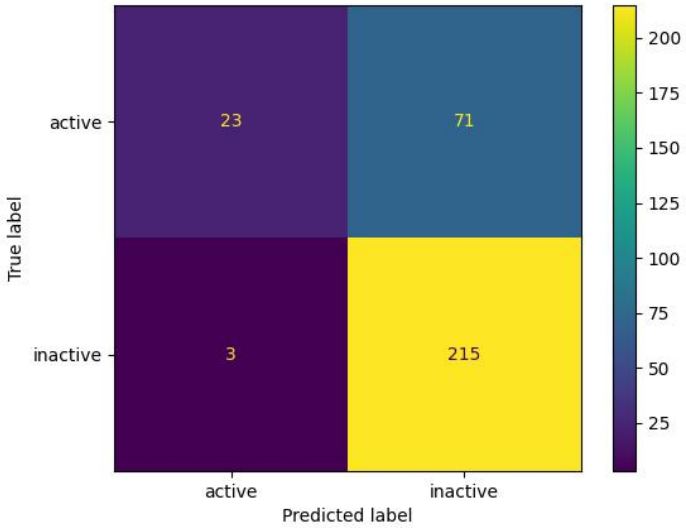


Figure 3.6: SMOTE random forest confusion matrix
The matrix shows that the model performed reasonably well with the diagonal values(23,215) being proportionally larger than the values of the other diagonal(3,71).

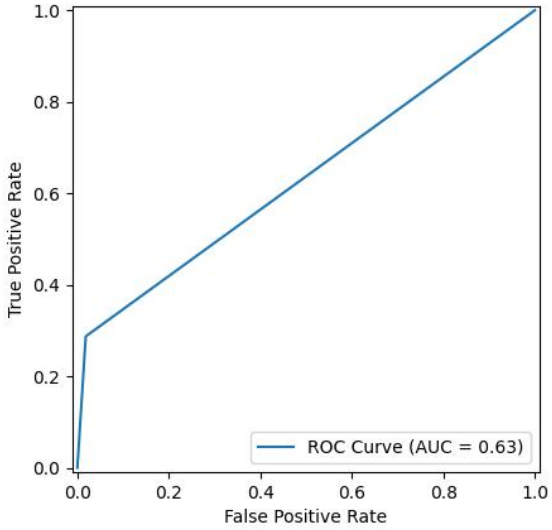


Figure 3.7: Baseline random forest ROC curve
The ROC curve indicates fair performance, as the AUC value is closer to 0.5 which would be a random estimator than it is to 1.0 which would indicate a perfect classifier.

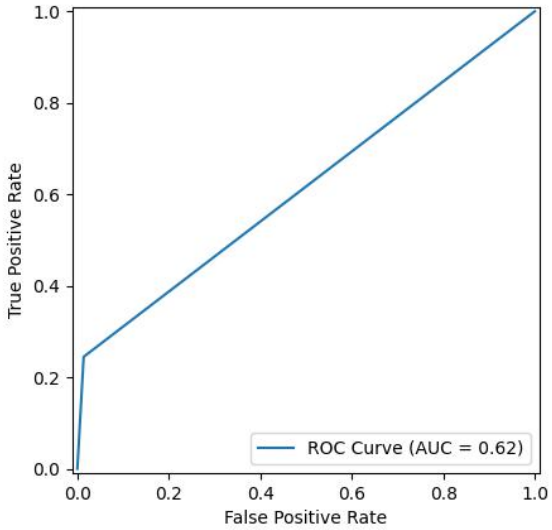


Figure 3.8: SMOTE random forest ROC curve
The ROC curve indicates performance close to a random classifier. The AUC value is slightly worse than the value presented in 3.3.

3.2.3 Dipeptidyl peptidase IV

Below the table representing the different results of the machine learning algorithms on the various test-sets can be found. The confusion matrices for the top two performing configurations can be found at 3.11 and 3.12 respectively. The ROC curves can be found at 3.9 and 3.10. The best scoring function was able to reach an accuracy of 77.21% on the test-set. The models were able achieve 0.773 AUC and 0.798 yield of active scores.

Table 3.7: Dipeptidyl peptidase IV performance test-set

Name	ACC	FPR	AUC	YA	EF	REF
baseline_rf	0.7721	0.2041	0.7730	0.7984	1.5393	79.8387
fe_smote_rf	0.7662	0.2122	0.7670	0.7912	1.5254	79.1165
fe_rf_per_knn	0.7112	0.3714	0.7082	0.6957	1.3412	78.7879
baseline_nn	0.6896	0.3347	0.6887	0.6963	1.3425	71.2121
baseline_knn	0.6896	0.3959	0.6865	0.6767	1.3046	76.8939
fe_smote_nn	0.6896	0.3306	0.6889	0.6978	1.3453	70.8333
fe_rf_mdi_knn	0.4892	0.5143	0.4891	0.5078	0.9791	50.7812

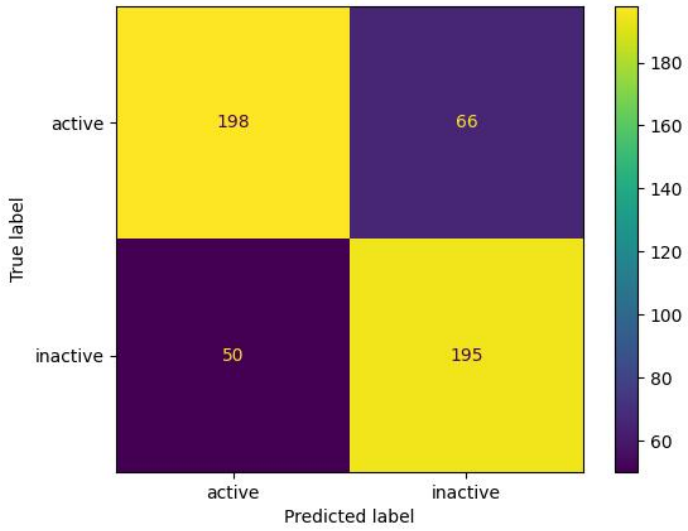


Figure 3.9: Baseline random forest confusion matrix
The confusion matrix indicates a very well-balanced dataset as there are nearly the same amount of actives as there are positives. The model is more likely to detect false negatives than it is to predict false positives.

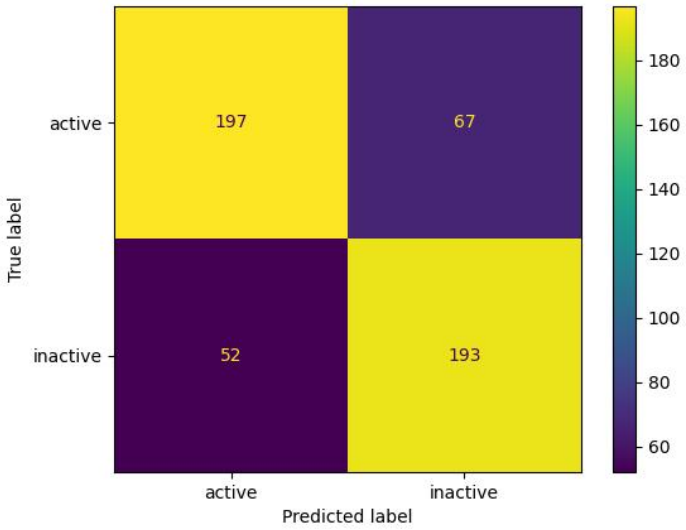


Figure 3.10: SMOTE random forest confusion matrix
The confusion matrix describes a reasonably good model performance as the values in the diagonal(198, 195) far outweigh the other values.

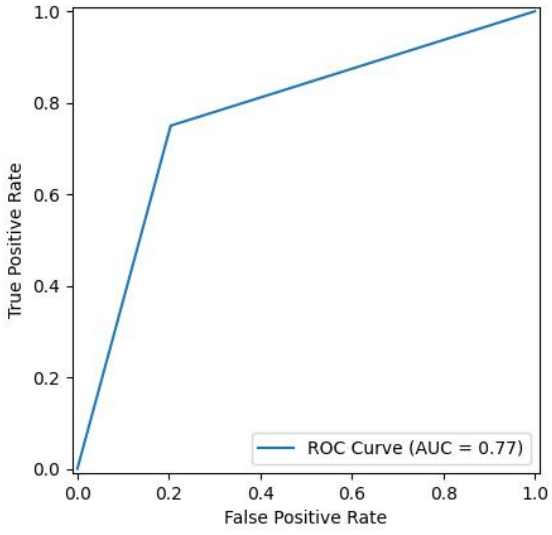


Figure 3.11: Baseline random forest ROC curve
The model is performing well at classifying between positive and negative classes. The AUC score is 0.77 which further indicates good performance.

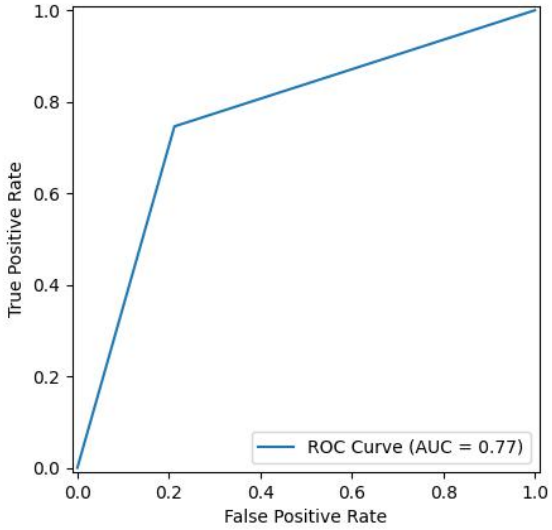


Figure 3.12: SMOTE random forest ROC curve
The ROC curve suggests that the performance of the random forest doesn't change with the introduction of SMOTE.

3.2.4 Monoamine oxidase B

This table summarizes the performance of various machine learning algorithms across different test sets. The top two configurations, visualized in ROC curves (see Figures 3.15 and 3.16), are further analyzed in confusion matrices (see Figures 3.13 and 3.14). The best scoring function achieved an accuracy of 75.98% on the test set. Furthermore, this result is accompanied by a low FPR of 13.89% and a fair REF of 69.697%.

Table 3.8: Monoamine oxidase B performance test-set

Name	ACC	FPR	AUC	YA	EF	REF
baseline_rf	0.7589	0.1389	0.7181	0.6970	1.9515	69.6970
fe_rf_per_knn	0.7054	0.1944	0.6653	0.6000	1.6800	60.0000
fe_smote_rf	0.7054	0.1944	0.6653	0.6000	1.6800	60.0000
baseline_nn	0.6964	0.1806	0.6472	0.5938	1.6625	59.3750
baseline_knn	0.6786	0.1667	0.6167	0.5714	1.6000	57.1429
fe_smote_nn	0.6696	0.2222	0.6264	0.5429	1.5200	54.2857
fe_rf_mdi_knn	0.5804	0.3333	0.5458	0.4146	1.1610	42.5000

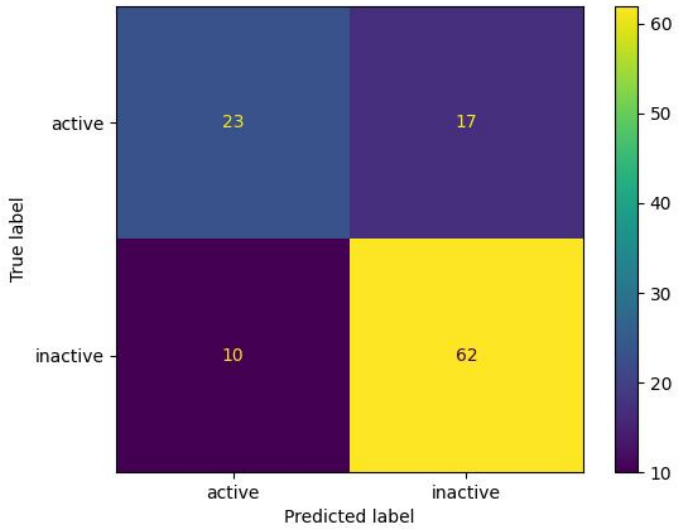


Figure 3.13: Baseline random forest confusion matrix
The confusion matrix shows, that the majority of samples in the test-set is inactive. The number of false positives is smaller, than the number of false negatives.

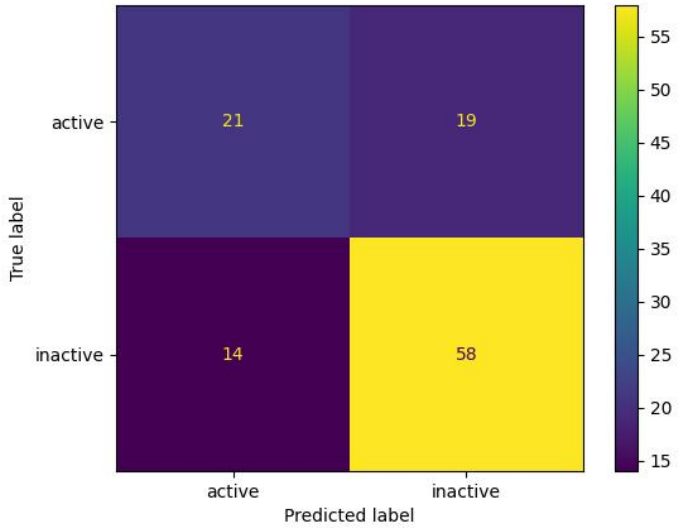


Figure 3.14: Feature engineering permutation importance confusion matrix
The model is reasonably accurate, as there are 21 true positives and 58 true negatives. Overall it performs slightly worse than the baseline random forest.

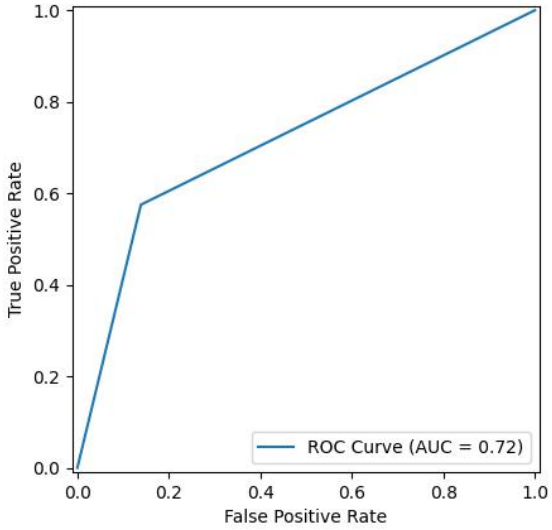


Figure 3.15: Baseline random forest ROC curve
Achieving an AUC of 0.72, the model demonstrates strong ability to distinguish between positive and negative instances.

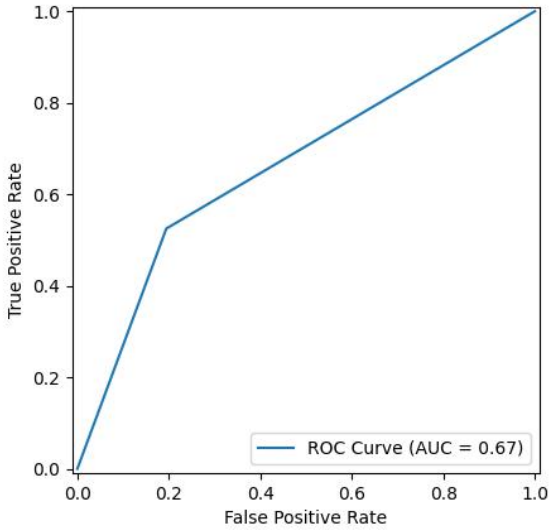


Figure 3.16: Feature engineering permutation importance ROC curve
The ROC curve suggests good discrimination between classes, but the AUC score of 0.67 indicates that it falls short of the reference value of 3.15.

3.2.5 Soluble epoxide hydrolase

This table compares the performance of various machine learning algorithms across different test sets. The two most effective configurations (visualized in confusion matrices: figures 3.13 and 3.14) are further analyzed in ROC curves (figures 3.15 and 3.16). Overall, the scoring functions achieved an accuracy of 80.00% on the test set. Additionally, the random forest models did not classify a single inactive compound as active.

Table 3.9: Soluble epoxide hydrolase performance test-set

Name	ACC	FPR	AUC	YA	EF	REF
fe_rf_per_knn	0.8000	0.0667	0.6667	0.6667	2.6667	66.6667
baseline_nn	0.7833	0.0667	0.6333	0.6250	2.5000	62.5000
baseline_rf	0.7667	0.0000	0.5333	1.0000	4.0000	100.0000
fe_smote_rf	0.7667	0.0000	0.5333	1.0000	4.0000	100.0000
baseline_knn	0.7333	0.0222	0.4889	0.0000	0.0000	0.0000
fe_rf_mdi_knn	0.7000	0.1333	0.5333	0.3333	1.3333	33.3333
fe_smote_nn	0.7000	0.0889	0.4889	0.2000	0.8000	20.0000

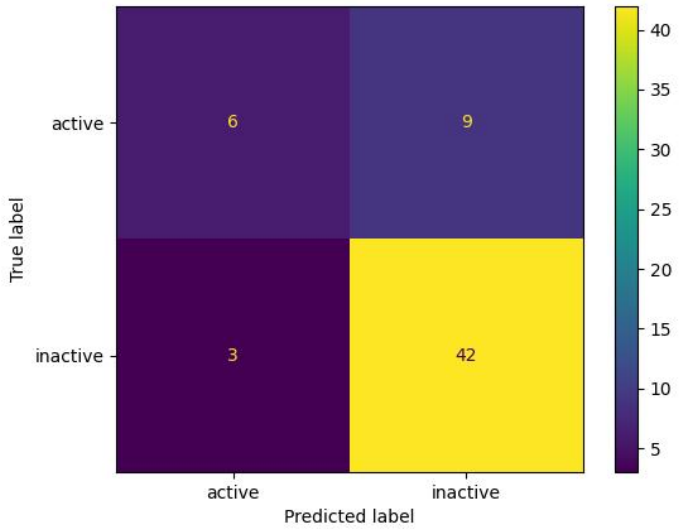


Figure 3.17: Feature engineering permutation importance confusion matrix for KNN The confusion matrix indicates a very small test-set. In addition to that the dataset is also very imbalanced.

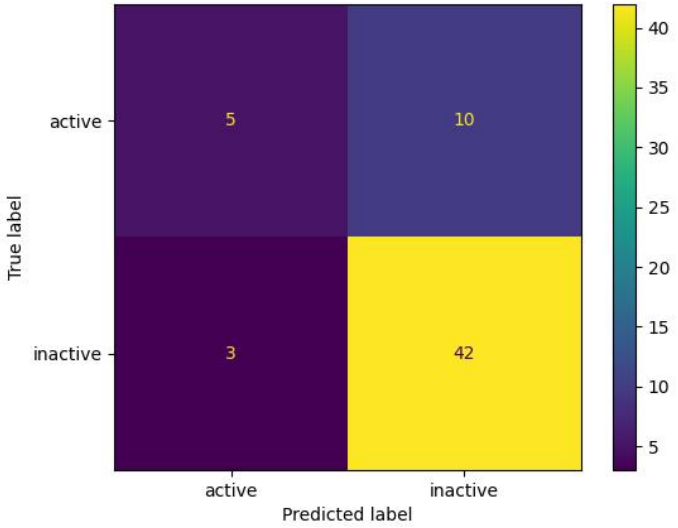


Figure 3.18: Baseline neural network confusion matrix
The proportionally high values in the diagonal(5, 42) signify good overall performance. Although the KNN performance of 3.17 is marginally better.

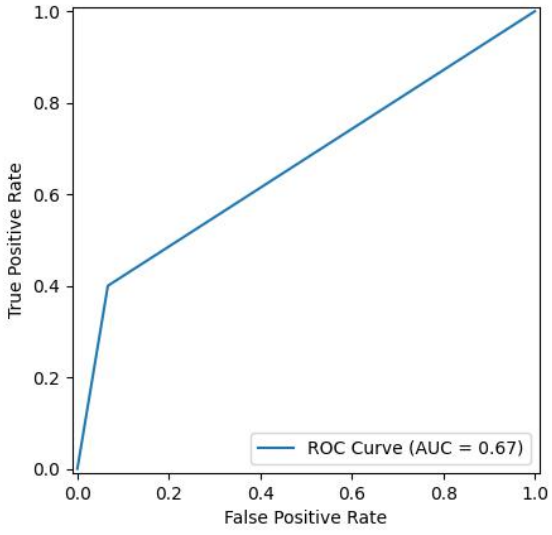


Figure 3.19: Feature engineering permutation importance ROC curve
An AUC of 0.67 demonstrates the model's ability to differentiate between positive and negative examples. Overall the performance is close to a random classifier.

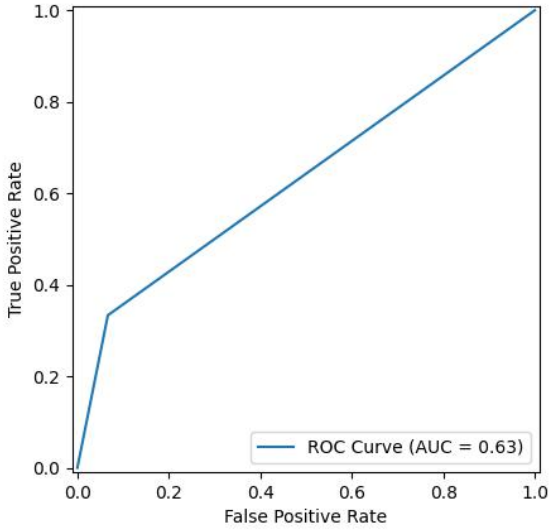


Figure 3.20: Baseline neural network ROC curve
The ROC curve demonstrates only fair performance as it is much closer to a random classifier than it is to a perfect classifier which would be a 1.0 AUC.

3.3 Performance Overview – Comparing ML-approaches

This chapter aims to compare the accumulated results generated in 3.2. To achieve the desired results the best performing version of each machine learning algorithm has been selected for each protein. In the following step the average for each of the machine learning algorithms is calculated. The results of this calculation can be seen in the table below.

Table 3.10: machine learning algorithms comparison

Name	ACC	FPR	AUC	YA	EF	REF
rf	0.7762	0.1380	0.6916	0.8276	2.3596	85.8631
knn	0.7352	0.2292	0.6684	0.6387	1.7419	69.6539
nn	0.7246	0.1937	0.6530	0.6215	1.6647	63.6876

In is obvious that the random forest algorithm outperforms the other algorithms. RF achieves noticeably better performance across all metrics.

Chapter 4

Discussion

4.1 Conclusion

The goals for this thesis where twofold. Firstly, the applicability of numerous machine learning approaches for activity prediction, and secondly the application of feature engineering methods to the base datasets.

The first goal was addressed by implementing *k nearest neighbor*, *random forest* and *neural network* algorithms. The results from those baseline implementations are very promising. The second goal builds upon the first goal. Through the implementation of various feature engineering methods the results of the baseline machine learning algorithms improved for a select number of configurations. An especially noteworthy feature engineering method in this context is SMOTE. Through the balancing of the provided datasets using SMOTE the accuracy of all models improved. When comparing the different machine learning approaches the *random forest* algorithm was able to achieve the best performance overall.

The overall performance of the models compares well to the scoring functions introduced in [1]. The following table represents the results for the AChE protein in [1]:

Figure 4.1: Scoring results for ACHE Birklbauer

STRATEGY	ACC	FPR	AUC	YA	EF	REF
+	0.690	0.472	0.733	0.686	1.232	81.955
++	0.665	0.208	0.747	0.773	1.389	77.320
+-	0.749	0.302	0.831	0.766	1.377	78.947
++--	0.745	0.226	0.812	0.8	1.438	80

The following results were achieved for the AChE protein using the methods proposed in this thesis:

Table 4.1: Acetylcholinesterase performance test-set

Name	ACC	FPR	AUC	YA	EF	REF
baseline_rf	0.8106	0.3285	0.7992	0.7716	1.4161	92.6829
fe_smote_rf	0.8007	0.3358	0.7894	0.7653	1.4046	91.4634
fe_smote_nn	0.7708	0.2993	0.7650	0.7684	1.4102	82.9268
baseline_nn	0.7674	0.2920	0.7626	0.7701	1.4134	81.7073
fe_rf_per_knn	0.7575	0.4307	0.7420	0.7177	1.3172	91.4634
baseline_knn	0.6844	0.5766	0.6629	0.6520	1.1966	90.2439
fe_rf_mdi_knn	0.5515	0.4307	0.5530	0.5986	1.0987	59.8639

The comparison between the two tables underlines the performance improvements of the machine learning approaches over the simple scoring functions introduced in [1]. Generally the improvements in ACC and REF are most notable. The improvements in the REF metric are of particular interest, as there is less money spent on investigating false positive compounds. Similar performance benefits can be observed for the other protein complexes.

4.2 Improvements and outlook

This thesis serves as a proof of concept for the applicability of machine learning models in combination with feature engineering for protein docking, as the achieved results are more accurate than those achieved using traditional approaches.

However, the machine learning approaches for protein docking where implemented using standard approaches and models. Through the use of more problem-tailored algorithmic methods the quality of the results will probably increase. Additionally, only a small sample of protein-ligand complexes was researched for this thesis. The machine learning methods proposed within this thesis need to be evaluated on a larger subset of protein-ligand compounds to determine whether machine learning is a viable approach for protein docking.

References

Literature

[1] Micha Johannes Birklbauer. "Automatic identification of important interactionsand interaction-frequency-based scoring inprotein-ligand complexes". MA thesis. FH Hagenberg, Aug. 31, 2021 (cit. on pp. v, vi, 2, 7, 15, 33, 34).

[2] Michael M. Mysinger et al. "Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking". *Journal of Medicinal Chemistry* 55.14 (July 2012), pp. 6582–6594. DOI: 10.1021/jm300687e. (Visited on 04/30/2024) (cit. on pp. v, vi, 2).

[3] Sebastian Salentin et al. "PLIP: fully automated protein–ligand interaction profiler". *Nucleic Acids Research* 43.Web Server issue (July 2015), W443–W447. DOI: 10.1093/nar/gkv315. (Visited on 04/23/2024) (cit. on pp. v, vi, 2, 6, 7).

[4] S. Myers and A. Baker. "Drug discovery–an operating model for a new era". eng. *Nature Biotechnology* 19.8 (Aug. 2001), pp. 727–730. DOI: 10.1038/90765 (cit. on p. 1).

[5] Joseph A. DiMasi, Ronald W. Hansen, and Henry G. Grabowski. "The price of innovation: new estimates of drug development costs". eng. *Journal of Health Economics* 22.2 (Mar. 2003), pp. 151–185. DOI: 10.1016/S0167-6296(02)00126-1 (cit. on p. 1).

[6] Lorenz M. Mayr and Peter Fuerst. "The Future of High-Throughput Screening". *SLAS Discovery* 13.6 (July 2008), pp. 443–448. DOI: 10.1177/1087057108319644. (Visited on 02/29/2024) (cit. on p. 1).

[7] Aleix Gimeno et al. "The Light and Dark Sides of Virtual Screening: What Is There to Know?" *International Journal of Molecular Sciences* 20.6 (Mar. 2019), p. 1375. DOI: 10.3390/ijms20061375. (Visited on 02/29/2024) (cit. on pp. 1, 2).

[8] Nataraj S. Pagadala, Khajamohiddin Syed, and Jack Tuszynski. "Software for molecular docking: a review". *Biophysical Reviews* 9.2 (Jan. 2017), pp. 91–102. DOI: 10.1007/s12551-016-0247-1. (Visited on 02/29/2024) (cit. on pp. 1, 2).

[9] A. Lavecchia and C. Di Giovanni. "Virtual screening strategies in drug discovery: a critical review". eng. *Current Medicinal Chemistry* 20.23 (2013), pp. 2839–2860. DOI: 10.2174/09298673113209990001 (cit. on p. 1).

[10] Jin Li, Ailing Fu, and Le Zhang. "An Overview of Scoring Functions Used for Protein–Ligand Interactions in Molecular Docking". en. *Interdisciplinary Sciences: Computational Life Sciences* 11.2 (June 2019), pp. 320–328. DOI: 10.1007/s12539-019-00327-w. (Visited on 02/29/2024) (cit. on p. 2).

[11] Balachandran Manavalan and Jooyoung Lee. "SVMQA: support-vector-machine-based protein single-model quality assessment". eng. *Bioinformatics (Oxford, England)* 33.16 (Aug. 2017), pp. 2496–2503. DOI: 10.1093/bioinformatics/btx222 (cit. on p. 3).

[12] Fatma-Elzahraa Eid, Mahmoud ElHefnawi, and Lenwood S. Heath. "DeNovo: virus-host sequence-based protein–protein interaction prediction". *Bioinformatics* 32.8 (Apr. 2016), pp. 1144–1150. DOI: 10.1093/bioinformatics/btv737. (Visited on 04/30/2024) (cit. on p. 3).

[13] Bin Li et al. "Development of a Drug-Response Modeling Framework to Identify Cell Line Derived Translational Biomarkers That Can Predict Treatment Outcome to Erlotinib or Sorafenib". *PLoS ONE* 10.6 (June 2015), e0130700. DOI: 10.1371/journal.pone.0130700. (Visited on 05/01/2024) (cit. on p. 3).

[14] Simon Johansson et al. "AI-assisted synthesis prediction". *Drug Discovery Today: Technologies*. Artificial Intelligence 32-33 (Dec. 2019), pp. 65–72. DOI: 10.1016/j.ddtec.2020.06.002. (Visited on 05/01/2024) (cit. on p. 3).

[15] Hongming Chen et al. "The rise of deep learning in drug discovery". eng. *Drug Discovery Today* 23.6 (June 2018), pp. 1241–1250. DOI: 10.1016/j.drudis.2018.01.039 (cit. on p. 3).

[16] Xiangxiang Zeng et al. "Target identification among known drugs by deep learning from heterogeneous networks". *Chemical Science* 11.7 (), pp. 1775–1797. DOI: 10.1039/c9sc04336e. (Visited on 04/30/2024) (cit. on p. 3).

[17] Srilekha Mamidala. *NeuroCADR: Drug Repurposing to Reveal Novel Anti-Epileptic Drug Candidates Through an Integrated Computational Approach*. Tech. rep. arXiv:2309.13047 [cs, q-bio] type: article. arXiv, Sept. 2023. DOI: 10.48550/arXiv.2309.13047. (Visited on 05/14/2024) (cit. on p. 3).

[18] Oren Z. Kraus, Jimmy Lei Ba, and Brendan J. Frey. "Classifying and segmenting microscopy images with deep multiple instance learning". *Bioinformatics* 32.12 (June 2016), pp. i52–i59. DOI: 10.1093/bioinformatics/btw252. (Visited on 05/01/2024) (cit. on p. 3).

[19] Kexin Huang et al. *MolDesigner: Interactive Design of Efficacious Drugs with Deep Learning*. Tech. rep. arXiv:2010.03951 [cs, q-bio] type: article. arXiv, Oct. 2020. DOI: 10.48550/arXiv.2010.03951. (Visited on 05/14/2024) (cit. on p. 3).

[20] Na Zhang and Hongtao Zhao. "Enriching screening libraries with bioactive fragment space". eng. *Bioorganic & Medicinal Chemistry Letters* 26.15 (Aug. 2016), pp. 3594–3597. DOI: 10.1016/j.bmcl.2016.06.013 (cit. on p. 3).

[21] Oleg Trott and Arthur J. Olson. "AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading". en. *Journal of Computational Chemistry* 31.2 (2010), pp. 455–461. DOI: 10.1002/jcc.21334. (Visited on 05/20/2024) (cit. on p. 3).

[22] Oliver Korb, Thomas Stützle, and Thomas E. Exner. "PLANTS: Application of Ant Colony Optimization to Structure-Based Drug Design". en. In: *Ant Colony Optimization and Swarm Intelligence*. Ed. by Marco Dorigo et al. Berlin, Heidelberg: Springer, 2006, pp. 247–258. DOI: 10.1007/11839088_22 (cit. on p. 3).

[23] Garrett M. Morris et al. "AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility". *Journal of computational chemistry* 30.16 (Dec. 2009), pp. 2785–2791. DOI: 10.1002/jcc.21256. (Visited on 05/20/2024) (cit. on p. 3).

[24] Sergio Ruiz-Carmona et al. "rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids". *PLoS Computational Biology* 10.4 (Apr. 2014), e1003571. DOI: 10.1371/journal.pcbi.1003571. (Visited on 05/20/2024) (cit. on p. 3).

[25] Connor Morris et al. "MILCDock: Machine Learning Enhanced Consensus Docking for Virtual Screening in Drug Discovery". *Journal of chemical information and modeling* 62 (Nov. 2022). DOI: 10.1021/acs.jcim.2c00705 (cit. on p. 3).

[26] Anurag Tripathi and U. C. Srivastava. "Acetylcholinesterase :A Versatile Enzyme of Nervous System". *Annals of Neurosciences* 15.4 (Feb. 2010), pp. 106–111. DOI: 10.5214/95. (Visited on 04/22/2024) (cit. on p. 5).

[27] Carol A. Rouzer and Lawrence J. Marnett. "Cyclooxygenases: structural and functional insights". *Journal of Lipid Research* 50.Suppl (Apr. 2009), S29–S34. DOI: 10.1194/jlr.R800042-JLR200. (Visited on 04/22/2024) (cit. on p. 5).

[28] Denise M. T. Yu et al. "The dipeptidyl peptidase IV family in cancer and cell biology". en. *The FEBS Journal* 277.5 (2010), pp. 1126–1144. DOI: 10.1111/j.1742-4658.2009.07526.x. (Visited on 04/22/2024) (cit. on p. 6).

[29] Rona R. Ramsay. "Molecular aspects of monoamine oxidase B". *Progress in Neuro-Psychopharmacology and Biological Psychiatry* 69 (Aug. 2016), pp. 81–89. DOI: 10.1016/j.pnpbp.2016.02.005. (Visited on 04/22/2024) (cit. on p. 6).

[30] Kara R. Schmelzer et al. "Soluble epoxide hydrolase is a therapeutic target for acute inflammation". eng. *Proceedings of the National Academy of Sciences of the United States of America* 102.28 (July 2005), pp. 9772–9777. DOI: 10.1073/pnas.0503279102 (cit. on p. 6).

[31] Yun Xu and Royston Goodacre. "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning". en. *Journal of Analysis and Testing* 2.3 (July 2018), pp. 249–262. DOI: 10.1007/s41664-018-0068-2. (Visited on 04/21/2024) (cit. on p. 8).

[32] Annette M. Molinaro, Richard Simon, and Ruth M. Pfeiffer. "Prediction error estimation: a comparison of resampling methods". *Bioinformatics* 21.15 (May 2005), pp. 3301–3307. DOI: 10.1093/bioinformatics/bti499. eprint: https://academic.oup.com/bioinformatics/article-pdf/21/15/3301/50340684/bioinformatics_21_15_3301.pdf (cit. on p. 8).

[33] Padraig Cunningham and Sarah Jane Delany. “k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples)”. *ACM Computing Surveys* 54.6 (July 2022). arXiv:2004.04523 [cs, stat], pp. 1–25. DOI: 10.1145/3459665. (Visited on 05/12/2024) (cit. on p. 8).

[34] Leo Breiman. “Random Forests”. en. *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. DOI: 10.1023/A:1010933404324. (Visited on 05/10/2024) (cit. on pp. 8, 9).

[35] Fatih Karabiber. *Gini Impurity*. URL: https://www.learndatasci.com/glossary/gini-impurity/ (visited on 05/11/2024) (cit. on p. 9).

[36] Warren S. McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. en. *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. DOI: 10.1007/BF02478259. (Visited on 05/18/2024) (cit. on p. 9).

[37] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. “Artificial neural networks: a tutorial”. *Computer* 29.3 (1996), pp. 31–44. DOI: 10.1109/2.485891 (cit. on p. 10).

[38] Juergen Schmidhuber. “Deep Learning in Neural Networks: An Overview”. *Neural Networks* 61 (Jan. 2015). arXiv:1404.7828 [cs], pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003. (Visited on 05/18/2024) (cit. on p. 10).

[39] Katarzyna Janocha and Wojciech Marian Czarnecki. *On Loss Functions for Deep Neural Networks in Classification*. Tech. rep. arXiv:1702.05659 [cs] type: article. arXiv, Feb. 2017. DOI: 10.48550/arXiv.1702.05659. (Visited on 05/18/2024) (cit. on p. 11).

[40] Thomas J. Archdeacon. *Correlation and regression analysis: a historian’s guide*. eng. OCLC: 27266095. Madison, Wis.: University of Wisconsin Press, 1994 (cit. on p. 11).

[41] Jiawei Zhang. *Gradient Descent based Optimization Algorithms for Deep Learning Models Training*. Tech. rep. arXiv:1903.03614 [cs, stat] type: article. arXiv, Mar. 2019. DOI: 10.48550/arXiv.1903.03614. (Visited on 05/18/2024) (cit. on p. 11).

[42] Sebastian Raschka. *How to compute gradients with backpropagation for arbitrary loss and activation functions?* en. May 2024. URL: https://sebastianraschka.com/faq/docs/backprop-arbitrary.html (visited on 05/18/2024) (cit. on p. 12).

[43] Spain) International Workshop on Artificial Neural Networks (1995 : Torremolinos. *From natural to artificial neural computation : International Workshop on Artificial Neural Networks, Malaga-Torremolinos, Spain, June 7-9, 1995 : proceedings*. eng. Berlin ; New York : Springer-Verlag, 1995. URL: http://archive.org/details/fromnaturaltoart1995inte (visited on 05/18/2024) (cit. on p. 12).

[44] Xiaoge Zhang, Prabhakar Srinivasan, and Sankaran Mahadevan. “Sequential Deep Learning from NTSB Reports for Aviation Safety Prognosis”. *Safety Science* 142 (June 2021). DOI: 10.1016/j.ssci.2021.105390 (cit. on p. 12).

[45] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*. Tech. rep. arXiv:2109.14545 [cs] type: article. arXiv, June 2022. DOI: 10.48550/arXiv.2109.14545. (Visited on 05/18/2024) (cit. on p. 12).

[46] Martín Abadi et al. *TensorFlow: A system for large-scale machine learning*. Tech. rep. arXiv:1605.08695 [cs] type: article. arXiv, May 2016. DOI: 10.48550/arXiv.1605.08695. (Visited on 05/14/2024) (cit. on p. 12).

[47] Aneesha B. Soman. *Gini Impurity vs Gini Importance vs Mean Decrease Impurity*. en. Oct. 2023. URL: https://medium.com/@aneesha161994/gini-impurity-vs-gini-importance-vs-mean-decrease-impurity-51408bdd0cf1 (visited on 05/11/2024) (cit. on p. 13).

[48] *4.2. Permutation feature importance*. en. URL: https://scikit-learn/stable/modules/permutation_importance.html (visited on 05/10/2024) (cit. on p. 13).

[49] Renato Ferreira de Freitas and Matthieu Schapira. "A systematic analysis of atomic protein–ligand interactions in the PDB". en. *MedChemComm* 8.10 (Oct. 2017), pp. 1970–1981. DOI: 10.1039/C7MD00381A. (Visited on 05/11/2024) (cit. on p. 14).

[50] Jonathon Shlens. *A Tutorial on Principal Component Analysis*. Tech. rep. arXiv:1404.1100 [cs, stat] type: article. arXiv, Apr. 2014. DOI: 10.48550/arXiv.1404.1100. (Visited on 05/11/2024) (cit. on p. 14).

[51] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". *Journal of Artificial Intelligence Research* 16 (June 2002). arXiv:1106.1813 [cs], pp. 321–357. DOI: 10.1613/jair.953. (Visited on 05/11/2024) (cit. on p. 15).

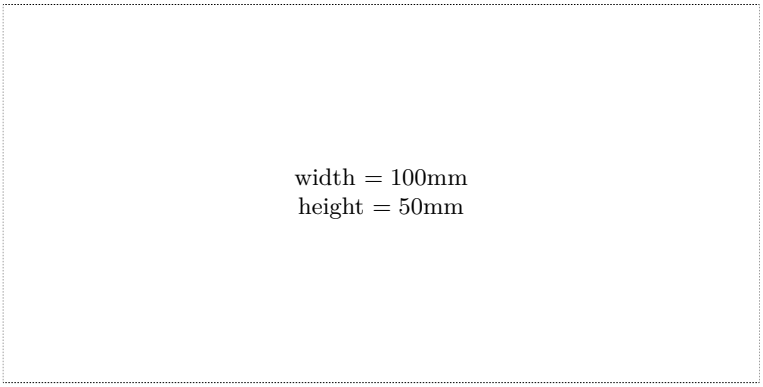
[52] Julio Cesar Dias Lopes et al. "The power metric: a new statistically robust enrichment-type metric for virtual screening applications with early recovery capability". *Journal of Cheminformatics* 9.1 (Feb. 2017), p. 7. DOI: 10.1186/s13321-016-0189-4. (Visited on 04/20/2024) (cit. on pp. 15, 16).

[53] Mohammad Hossin and Sulaiman M.N. "A Review on Evaluation Metrics for Data Classification Evaluations". *International Journal of Data Mining & Knowledge Management Process* 5 (Mar. 2015), pp. 01–11. DOI: 10.5121/ijdkp.2015.5201 (cit. on p. 15).

[54] Deborah Giordano et al. "Drug Design by Pharmacophore and Virtual Screening Approach". *Pharmaceuticals* 15.5 (May 2022), p. 646. DOI: 10.3390/ph15050646. (Visited on 04/20/2024) (cit. on p. 16).

Check Final Print Size

— Check final print size! —



— Remove this page after printing! —