technische universität dortmund

Computer-Aided Chemical Biology

Faculty of Chemistry and Chemical Biology

PhD Thesis

Development of an Open-Source Application for Multiprotic Small Molecule pK_A Prediction Based on Machine Learning and Experimental Data

Marcel Baltruschat

1st reviewer: Prof. Dr. Paul Czodrowski

2nd reviewer: Prof. Dr. Stefan M. Kast

Dortmund, 23.06.2024

Danksagung

An dieser Stelle möchte ich mich bei einigen Personen bedanken, die wesentlich zum Gelingen meiner Promotion und dieser Dissertation beigetragen haben:

Zuerst möchte ich mich bei meinem Doktorvater **Prof. Dr. Paul Czodrowski** für die Betreuung, Unterstützung und wissenschaftliche Begleitung bedanken. Er hatte immer ein offenes Ohr für mich und unterstützte mich in allen Belangen. Ebenso möchte ich mich bei **Dr. Michael Edmund Beck** für die übergangsweise Betreuung und die damit verbundenen intensiven fachlichen Diskussionen bedanken. Herrn **Prof. Dr. Stefan M. Kast** danke ich für die Übernahme des Zweitgutachtens.

Danken möchte ich auch meinem sehr guten Freund und Kollegen **Helge Vatheuer**. Mit ihm als Mitstreiter während der Promotion waren auch steinigere Passagen leicht zu bewältigen. Er stand mir jederzeit fachlich und persönlich zur Seite und sorgte dafür, dass auch die Erholung in der Freizeit nicht zu kurz kam. Außerdem möchte ich mich bei **Gianna Isabel Pohl** bedanken. Sie stand jederzeit hinter mir und hat mich während der gesamten Zeit mit Zuspruch und Motivation begleitet und somit maßgeblich zum Gelingen dieser Arbeit beigetragen.

Ein besonderer Dank gilt auch meinen Eltern **Renate und Andreas Baltruschat**. Sie haben mir während des gesamten Studiums immer den Rücken frei gehalten und sind maßgeblich dafür verantwortlich, dass ich in so kurzer Zeit so weit gekommen bin.

Nicht zuletzt möchte ich mich bei allen ehemaligen und aktuellen **Mitgliedern der Arbeitsgruppe** von Prof. Czodrowski für den starken Zusammenhalt und die vielen Gespräche bedanken. Es war eine ganz besondere Zeit für mich.

Contents

Chapter 1

Abstracts

1.1 Zusammenfassung

Die Säure-Base-Dissoziationskonstante (pK_a) und damit der Säure-Base-Charakter eines Moleküls hat einen weitreichenden Einfluss auf seine biopharmazeutischen, pharmakodynamischen und pharmakokinetischen Eigenschaften. Dazu gehören unter anderem Löslichkeit, Absorption, Permeabilität, Proteinbindung und Lipophilie. Diese weitreichenden Auswirkungen sind der Grund dafür, dass pK_a-Werte experimentell bestimmt und im Rahmen des Arzneimittelzulassungsverfahrens angegeben werden müssen. Eine genaue Abschätzung der pK_a-Werte ist daher von entscheidender Bedeutung für ein erfolgreiches Wirkstoffdesign[1–7]. Für die Vorhersage von pK_a-Werten für kleine Moleküle existieren mehrere kommerzielle und nichtkommerzielle Tools und Ansätze, jedoch fehlt ein quelloffenes, frei verfügbares pK_a-Vorhersagetool, das sich in seiner Vorhersagequalität und seinem Funktionsumfang mit den kommerziellen Tools messen kann. Ziel dieser Arbeit ist es, ein solches Tool zu entwickeln, welches auf maschinellem Lernen und ausschließlich experimentell ermittelten pK_a-Daten basiert.

Dazu wurden zunächst experimentell ermittelte pK_a-Daten durch umfangreiche Literatur- und Datenbankrecherchen sowie durch Kooperationen mit verschiedenen Pharmaunternehmen und Softwareanbietern zusammengetragen und standardisiert. Anschließend wurde mit der Entwicklung eines auf maschinellem Lernen basierenden pK_a-Vorhersagetools für monoprotische Moleküle begonnen. Nach der Evaluierung verschiedener Transformationsmethoden vom Molekül zu für das Training verwendbare Eingabedaten sowie verschiedener Machine-Learning-Algorithmen konnte ein Random Forest Modell entwickelt werden, das, auf Basis externer Testdatensätze aus der Literatur und der pharmazeutischen Industrie sowie einer 5-fachen Kreuzva-

lidierung, mit dem kommerziellen Tool ChemAxon Marvin[8] mithalten kann sowie die Vorhersagequalität vergleichbarer, zu diesem Zeitpunkt veröffentlichter Open-Source-Modelle übertrifft. Die 5-fache Kreuzvalidierung ergab einen MAE von 0.682, einen RMSE von 1.032 und einen R^2 von 0.82[9].

Im nächsten Schritt wurde der Fokus auf multiprotische Moleküle gelegt. Hier musste die Datenaufbereitung und Vorverarbeitung erweitert werden, um die konkreten Fragestellungen der Identifizierung und Lokalisierung von Titrationsstellen in Molekülen sowie die Zuordnung der einzelnen experimentellen pK_a-Werte zu ihren jeweiligen titrierbaren Gruppen zu bearbeiten. Zu diesem Zweck wurde das Programm Multiprotic pK_a Processor (MPP) entwickelt. MPP führt alle notwendigen Vorverarbeitungsschritte durch, identifiziert und lokalisiert titrierbare Gruppen und führt alle weiteren vorbereitenden Schritte durch, um einen schnellen Start des maschinellen Lernens auf Basis der bearbeiteten Datensätze zu ermöglichen. Zusätzlich wird eine detaillierte Analyse der titrierbaren Gruppen der Moleküle in den gegebenen Datensätzen durchgeführt. MPP liefert auch eine Liste von SMILES Arbitrary Target Specification (SMARTS)-Mustern, die die am häufigsten vorkommenden titrierbaren Gruppen repräsentieren und auf Basis der Eingabedatensätze generiert wurden. Insgesamt wurden 16 Datensätze aus verschiedenen Quellen mit insgesamt 84 957 pK_a-Werten für 26 568 verschiedene Moleküle verarbeitet und für das maschinelle Lernen vorbereitet.

Schließlich wurde ein multiprotisches pK_a-Vorhersagemodell basierend auf der Graph Convolutional Neural Network (GCN)-Architektur auf der Grundlage der aus MPP resultierenden Datensätze entwickelt. Die Implementierung des GCN sowie die Umwandlung der Moleküle in Graphen und das Hinzufügen von Kanten- und Knoteneigenschaften erfolgte mit PyTorch[10] und PyTorch Geometric[11] (PyG). Zur Optimierung der Architektur und der Hyperparameter wurde eine Bayes'sche Optimierung mit 500 Durchläufen mittels Optuna[12] durchgeführt. Die einzelnen Modelle wurden mit externen Testdatensätzen evaluiert und mit MLflow[13] getrackt und dokumentiert. Das beste Modell erreichte über alle titrierbaren Gruppen aller mono-, di- und triprotischen Moleküle für den Literatur-Testdatensatz von Settimo *et al.*[14] einen MAE von 0.414, einen RMSE von 0.587 und einen R^2 von 0.929. Für den Industrie-Testdatensatz von Novartis[15] erreichte das Modell einen MAE von 0.791, einen RMSE von 1.048 und einen R^2 von 0.808. Insgesamt liegt der MAE-Wert aller Testdatensätze kombiniert bei 0.748. Im Rahmen einer erweiterten Evaluierung wurde ein zweites Modell mit den gleichen Hyperparametern und einem angepassten Trainingsdatensatz trainiert und mit den SAMPL6[16] und SAMPL7[17] Daten-sätzen als externe Testdatensätze evaluiert. Für den monoprotischen Teil beider Datensätze erreichte das Modell einen MAE von 0.442 bzw.

0.722, einen RMSE von 0.592 bzw. 0.907 und einen R^2 von 0.915 bzw. 0.851 für SAMPL6[16] und SAMPL7[17]. Im Falle von SAMPL6[16] können die statistischen Werte aufgrund der Filterung im Zuge der Präprozessierung nicht mit den veröffentlichten statistischen Werten verglichen werden.

1.2 Abstract

The acid-base dissociation constant (pK_a) and thus the acid-base character of a molecule has a far-reaching influence on its biopharmaceutical, pharmacodynamic and pharmacokinetic properties. These include solubility, absorption, permeability, protein binding and lipophilicity. These wide-ranging effects are the reason why pK_a values must be determined experimentally and reported as part of the drug approval process. Accurate estimation of pK_a values is therefore critical for successful drug design [1–7]. Several commercial and non-commercial tools and approaches exist for predicting pK_a values for small molecules, but an open source, freely available pK_a prediction tool that can compete with the commercial tools in prediction quality and feature set is lacking. The aim of this work is to develop such a tool based on machine learning and exclusively experimentally determined pK_a data.

To achieve this, experimentally determined pK_a data were first compiled and standardised through extensive literature and database searches, as well as collaborations with various pharmaceutical companies and software vendors. Subsequently, the development of a machine learning based pK_a prediction tool for monoprotic molecules was initiated. After evaluation of different methods to transform the molecule into input data for training, as well as different machine learning algorithms, a random forest model was developed which, based on external test datasets from the literature and the pharmaceutical industry, as well as a 5-fold cross-validation, was able to compete with the commercial tool ChemAxon Marvin[8] and to outperform the prediction quality of comparable open source models published at that time. The 5-fold cross-validation yielded a MAE of 0.682, a RMSE of 1.032 and a R^2 of 0.82[9].

The next step was to focus on multiprotic molecules. Here the data preparation and preprocessing had to be extended to deal with the specific problems of identifying and localising titration sites in molecules and assigning the individual experimentally determined pK_a values to their respective titratable groups. For this purpose the program MPP was developed. MPP performs all necessary preprocessing steps, identifies and locates titratable groups, and performs all other preparatory steps to allow a quick start of machine learning based on the processed datasets. It also performs a detailed analysis of the titratable groups of the molecules in the given datasets. MPP also provides a list of SMARTS patterns representing the most

abundant titratable groups generated based on the input datasets. A total of 16 datasets from different sources with a total of 84 957 pK_a values for 26 568 different molecules were processed and prepared for machine learning.

Finally, a multiprotic pK_a prediction model based on the GCN architecture was developed using the datasets resulting from MPP. The implementation of GCN as well as the transformation of molecules into graphs and the extension of edge and node properties was done using PyTorch[10] and PyG. Bayesian optimisation with 500 runs was performed using Optuna[12] to optimise the architecture and hyperparameters. Each model was evaluated on external test datasets and tracked and documented using MLflow[13]. The best model achieved a MAE of 0.414, a RMSE of 0.587 and a R^2 of 0.929 across all titration sites of all mono-, di- and triprotic molecules for the Settimo *et al.*[14] literature test dataset. For the Novartis[15] industry test dataset, the model achieved a MAE of 0.791, a RMSE of 1.048 and a R^2 of 0.808. Overall, the MAE of all test datasets combined was 0.748. As part of an extended evaluation, a second model was trained with the same hyperparameters and a customized training dataset, and evaluated using the SAMPL6[16] and SAMPL7[17] datasets as external test datasets. For the monoprotic part of both datasets, the model achieved a MAE of 0.442 and 0.722, a RMSE of 0.592 and 0.907, and a R^2 of 0.915 and 0.851 for SAMPL6[16] and SAMPL7[17], respectively. In the case of SAMPL6[16], the statistical values cannot be compared with the published statistical values due to filtering in the course of preprocessing.

Chapter 2

Introduction

2.1 Acid dissociation constant

2.1.1 Definition

The acid dissociation constant, given as K_a, describes the strength of an acid in a solution. K_a is the equilibrium constant for the dissociation in context of an acid-base reaction

$$\mathrm{HA} \rightleftharpoons \mathrm{A^-} + \mathrm{H^+} \tag{2.1}$$

where HA is an acid that dissociates into its conjugated base $\mathrm{A^-}$ and a hydrogen ion $\mathrm{H^+}$. The acid dissociation constant itself is defined by[18]:

$$K_a = \frac{[\mathrm{A^-}][\mathrm{H^+}]}{[\mathrm{HA}]} \tag{2.2}$$

The square brackets indicate the concentrations of the single parts at equilibrium. If K_a and thus the strength of the equilibrium is less than 1, the acid dominates in neutral form (left side of equation (2.1)), if it is greater than 1, the acid prevails in its dissociated form (right side of equation (2.1)). In most cases, the strength of this equilibrium is of interest, which is why the strength of the acid is usually given as the negative decadic logarithm of K_a for the standard state concentration $c^\circ = 1\frac{mol}{l}$ [19, 20]:

$$\mathrm{p}K_a = -\log_{10}\frac{K_a}{c^\circ} \tag{2.3}$$

Along with this, based on the pK_a value the pH value at which the system is in equilibrium can be calculated with the *Henderson–Hasselbalch* equation. In this context, equilibrium means that both forward and backward reactions happen at the same rate[21]:

$$pH = pK_a + \log_{10} \frac{[A^-]}{[HA]} \tag{2.4}$$

In most experimental determination methods for pK_a values, the measured value is used as a function of pH, which results in a sigmoidal curve. The pK_a value can then be resolved by finding the inflection point at which pH equals pK_a. An exemplary sigmoidal titration curve for carboxylic acid including the distribution of its species are shown in figure 2.1.

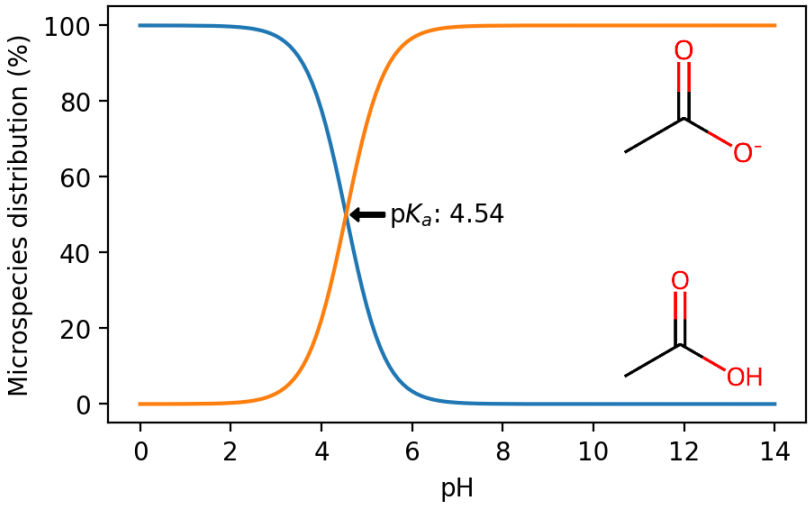


Figure 2.1: Example microspecies distribution of carboxylic acid: The pK_a value can be retrieved by calculating the point where both microspecies are equally distributed. The exemplary pK_a value was retrieved by ChemAxon Marvin[8].

2.1.2 Microscopic and macroscopic pK_a

In the case of multiprotic molecules, i.e. molecules with several ionisable centres, macroscopic and microscopic pK_a values must be distinguished. Macroscopic pK_a values describe the dissociation ability of the entire molecule and are composed of the microscopic pK_a values. A microscopic pK_a value is assigned to a specific titratable group and describes the release or uptake of a proton at this very ionisable centre. The macroscopic pK_a value can be calculated using the following formula[22]:

$$K_a^{macro} = \sum_{j=1}^{M^{deprot}} \frac{1}{\sum_{i=1}^{N^{prot}} \frac{1}{K_{ij}^{micro}}} \tag{2.5}$$

In this equation M^{deprot} represents M deprotonated microstates and N^{prot} stands for N protonated microstates. Most experimental measurement methods only provide macroscopic pK_a values, with the exception of Nuclear Magnetic Resonance (NMR) spectroscopy, for example[23, 24]. However, microscopic pK_a values are necessary to determine the different microstates, whereby a microstate represents the state of the microscopic protonation and the respective tautomer of a molecule.

To illustrate this, figure 2.2 below shows an example of the microstate network for the molecule SM07 containing four protonation sites from the SAMPL6 competition. It can be seen that the molecule can enter a total of eleven microstates, distributed over a total of five overall protonation states from 0 (neutral) to +4. These total protonation states can be distinguished by the columns in the figure. Within a column, the microstates differ in terms of which of the four titratable groups are protonated leading to the overall protonation state. The transition of the individual microstates in the form of (de)protonation is indicated by the black arrows in the figure. The red arrows indicate a transition between two microstates by tautomerization[25].

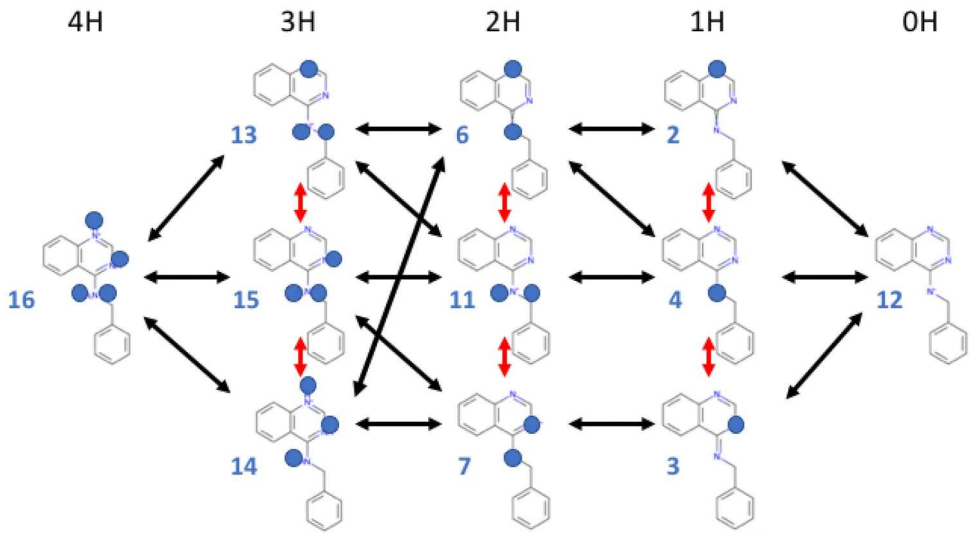


Figure 2.2: Microstate network of SAMPL6 molecule SM07 as depicted in Gunner *et al.* (2020)[25]. The black arrows indicate a microstate transition in terms of (de)protonation, the red arrows indicate a transition between two microstates by tautomerization.

2.1.3 Influences on pK_a

The acid dissociation constant pK_a is influenced by different parameters and therefore actually not constant. The temperature T for example affects the acid dissociation constant according to the *Van 't Hoff* equation where R is the gas constant and ΔH° is the standard enthalpy change of dissociation:

$$\frac{d}{dT} \ln K_a = \frac{\Delta H^\circ}{RT^2} \tag{2.6}$$

Assuming ΔH° to be independent of temperature plotting of pK_a against $\frac{1}{T}$ results in a linear plot. In this case the pK_a value decreases if the temperature increases. The linear relation does not apply in every case since ΔH° in fact depends on the temperature which results in a non-linear relationship of the pK_a value to temperature. The temperature dependency of the pK_a value of some exemplary molecules are shown in figure 2.3[26, 27].

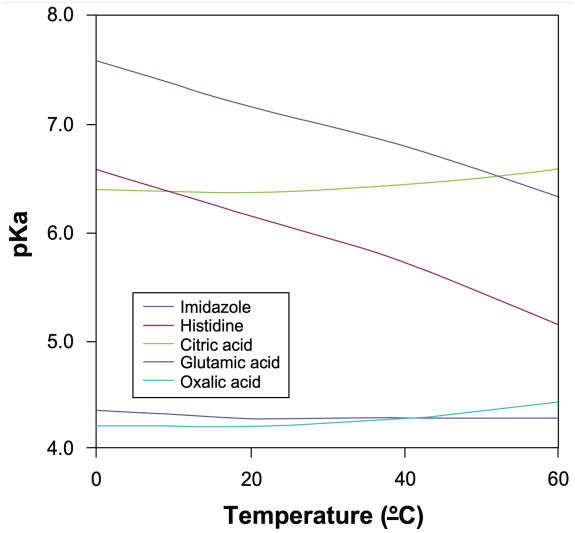


Figure 2.3: Exemplary temperature dependency of pK_a: The figure shows the change in pK_a for different exemplary molecules at different measurement temperatures. It was taken from Reijenga *et al.* (2013)[26] with data from Everaerts *et al.* (1976)[27].

The experimental determination of pK_a values is also influenced by the ionic strength of the solution that is investigated. The ionic strength I is a measure of the concentration of ions in a solution and can be calculated with the following formula[28, 29]:

$$I = \frac{1}{2} \sum_{i=1}^{n} c_i z_i^2 \tag{2.7}$$

Here, c_i represents the concentration of ion i of charge z_i. The influence of ionic strength is often neglected even if it can change the pK_a of a molecule significantly. To take this into account the equation 2.2 has to be extended with the activity of the reacting species[28, 29]:

$$K_a = \frac{a_{H^+} a_{A^-}}{a_{HA}} = \frac{\gamma \frac{[H^+]}{c^\circ} \gamma \frac{[A^-]}{c^\circ}}{\gamma \frac{[HA]}{c^\circ}} \tag{2.8}$$

In this equation γ is the activity coefficient and a is the activity. With the Debye-Hückel equation the activity coefficient can be calculated with z as the charge, T as the temperature and ϵ as dielectric constant[28, 29]:

$$log\gamma = -z^2(1.824 \times 10^6 \times (\epsilon T)^{-\frac{3}{2}}) I^{\frac{1}{2}} \tag{2.9}$$

Unfortunately, this equation is only valid at low ionic strength. For higher ionic strengths it is necessary to include terms for the effective radius of the ions or with empirical constants. Generally speaking for acids in low ionic strengths the pK_a values of the compound under investigation decrease if the ionic strength of the solution increases[26, 28–32].

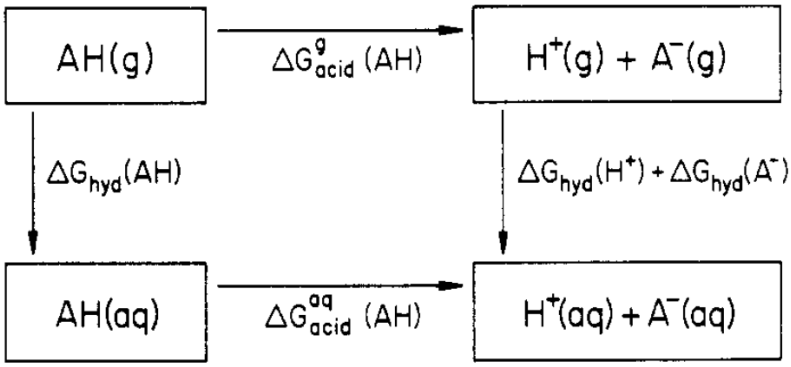


Figure 2.4: Thermodynamic cycle for determination of free energies of hydration for carboxylate ions as depicted in Kang *et al.* (1987)[33].

Finally, the pK_a measurement is influenced by the used solvent. One strategy to calculate the difference of pK_a values of a molecule in different solvents is based on the thermodynamic cycle which describes the procedure of a proton dissociation. The thermodynamic cycle as depicted in Kang *et al.* (1987)[33] is shown in figure 2.4. Based on this the difference between the solvation energies of the dissociated acid and the acid itself influences the pK_a value of a molecule. An example of the

influence of different solvents on the pK_a value of phenol taken from Rossini *et al.* (2018)[34] is shown in figure 2.5[26, 33–35].

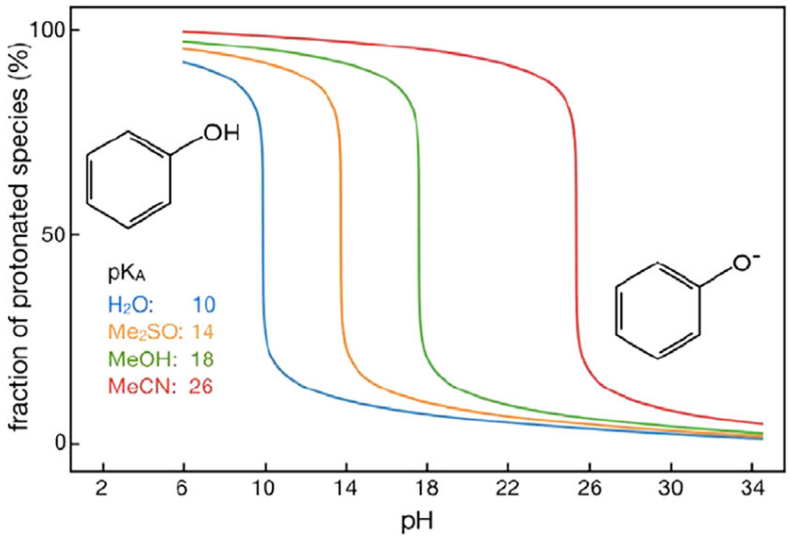


Figure 2.5: Influence of different solvents on pK_a: The pK_a value of phenol changes significantly dependent on the solvent used during the experimental determination. The figure was taken from Rossini *et al.* (2018)[34].

2.1.4 pK_a determination methods

There are several different ways to measure the pK_a value, each having benefits and drawbacks. Not all methods are appropriate for all molecules, thus it is preferable to know in advance which approach will result in a successful measurement in order to save time and money. An overview of the various analytical techniques and the dates of their introduction are shown in figure 2.6 taken from Reijenga *et al.* (2013)[26].

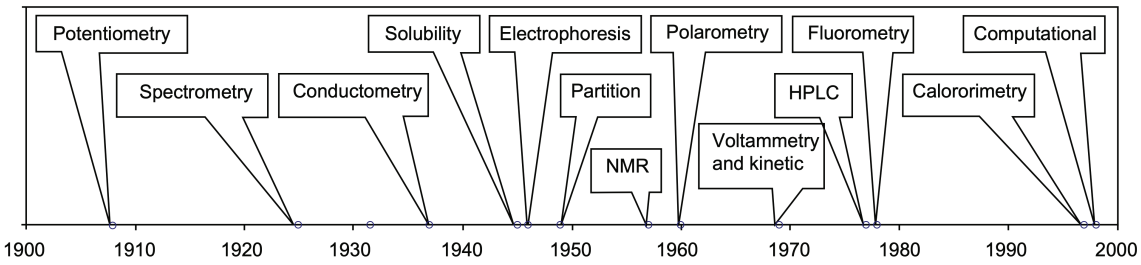


Figure 2.6: Overview of experimental pK_a determination methods taken from Reijenga *et al.* (2013)[26].

Some of the methods used for experimental pK_a value determination are, for example, potentiometry, NMR spectroscopy, UV/VIS spectroscopy and Isothermal Titration Calorimetry (ITC). Potentiometry is highly exact and can be automated well, which makes it commercially interesting, although a high amount of the sample

in the milligram range is required for this method. This makes potentiometry an expensive technique depending on the compound price[26].

NMR spectroscopy has for instance the advantage to be able measure impure mixtures. It works by measuring the alteration of chemical shifts of NMR-active nuclei which depend on the protonation state of adjacent acidic or basic sites. Additionally, with NMR spectroscopy it is possible to determine where the protonation or deprotonation takes place within the molecule[26].

Due to its accuracy, simplicity, and reproducibility, UV/VIS spectrometry, introduced in 1925 by Holmes and Snyder[36], is one of the most widely used techniques for measuring a compound's pK_{a} value. This method's drawback is the requirement that a chromophore or fluorophore must be present close to the ionization site. Without such a chromo- or fluorophore the pK_{a} value can not be determined[26].

ITC enables the simultaneous determination of a reaction's binding enthalpy (ΔH) and binding constant (K_a). It is also possible to compute the thermodynamic parameters entropy change (ΔS) and Gibbs free energy change (ΔG). Generally, ITC is used to measure binding affinities between ligands and proteins. The technique is extended to pK_{a} determination by considering protonation as a binding process[26].

2.2 ChemAxon Marvin

The pK_{a} prediction tool from ChemAxon Marvin[8] (CXM) is used at several points in this thesis and serves as a reference program, both functionally and qualitatively, for the development of a licence-free and open-source multiprotic pK_{a} prediction tool. For this reason, this chapter will take a closer look at the functionality of this program as well as its methodology for calculating pK_{a} values. Unfortunately, the exact methodology behind the pK_{a} prediction by CXM has not yet been published in a peer-reviewed journal. The following information therefore refers to the explanations on the official documentation page[37], as well as two presentations by Szegezdi and Csizmadia at the American Chemical Society meetings in 2004[38] and 2007[39].

The pK_{a} prediction of CXM is based, among other things, on the fact that the partial charge distribution changes as a result of the protonation or deprotonation of the titration sites. This distribution is strongly related to the process of (de)-protonation and can therefore be used to predict or calculate pK_{a} values. The prediction of pK_{a} values using empirically calculated partial charges was first presented by Szegezdi and Csizmadia at the 2004 National Meeting of the American Chemical Society (ACS)[38]. In addition to the partial charges, polarizability and structure-specific

elements such as hydrogen bonding interactions and steric strains are also taken into account. According to the presentations, the procedure can be summarised in the following steps[38, 39]:

1. Determine the most dominant tautomeric form of the molecule
2. Determine the titration sites within the molecule
3. Generate all microspecies
4. Calculate the partial charge distribution of the microspecies[40]
5. Calculate the polarizability of the microspecies[41]
6. Include hydrogen bonding and steric strains
7. Calculation of the ratio of microspecies[42]
8. Calculation of microscopic pK_{a} values
9. Calculation of macroscopic pK_{a} values

According to Szegezdi and Csizmadia (2004 and 2007)[38, 39], the macroscopic pK_{a} values can be calculated from the microscopic pK_{a} values using the following formula:

$$K_{a,i} = \frac{\sum_j c_j^{(i)}}{\sum_k c_k^{(i-1)}}[H^+] \tag{2.10}$$

In equation 2.10, i is an integer from 1 to N, where N is the number of titratable groups in the molecule. $[H^+]$ is the proton concentration of the aqueous solution, $c_j^{(i)}$ is the concentration of the j^{th} microspecies that has released i protons from the fully protonated form of the molecule, and $c_k^{(i-1)}$ is the concentration of the k^{th} microspecies that has released $i-1$ protons from the fully protonated molecule[38, 39].

The underlying model in CXM is based on the titration site-specific partial charges, polarizabilities, steric strains and hydrogen bond interactions obtained by linear or non-linear regression equations[38, 39]. As an example, one such regression equation is explained using the partial charge relationship to the microscopic pK_{a} values. A linear regression is performed using the quadratic functional form, where $p1$, $p2$ and $p3$ represent the regression coefficients[38]:

$$\mathrm{p}K_a = p_1(1+q)^2 + p_2(1+q) + p_3 \tag{2.11}$$

For very weak or very strong bases and acids, the relationship between the microscopic pK_a value and the partial charge corresponds more to the following equation 2.12, where a non-linear regression has been performed with the given equation, than to the previous equation 2.11. In analogy to equation 2.11, equation 2.12 gives a better fit[38]:

$$pK_a = p_1 exp(p_1, q) + p_2 q + p_3 \tag{2.12}$$

For monoprotic molecules, the information from the various linear and non-linear regressions is combined in the following equation 2.13[39]. For multiprotic molecules, Szegezdi and Csizmadia (2007) write: "The pK_a calculation of multiprotic molecules is governed by theoretically derived kinetic equations in our model"[39]. The documentation further states that the macroscopic pK_a values of multiprotic molecules are calculated based on the microspecies distribution and the global mass and charge conservation law[37].

$$pK_a = aQ + bP + cS + d \tag{2.13}$$

In this equation, Q represents the partial charge increment, P the polarizability increment and S the sum of structure specific increments. a, b, c and d are the titration site specific regression coefficients[39].

CXM's performance has been compared with other pK_a prediction tools in a number of publications and shows average prediction quality in the field of empirical commercially available programs[14, 43–45]. This makes CXM a good reference tool for the development of an open source pK_a prediction tool in the context of this thesis.

2.3 Motivation

2.3.1 Importance of the acid dissociation constant

The pK_a value and thus the acid/base character of a molecule has a far-reaching influence on its biopharmaceutical, pharmacodynamic and pharmacokinetic properties. These include solubility, absorption, permeability, protein binding, and lipophilicity, among others. These wide-ranging effects are the reason why pK_a values must be experimentally determined and reported as part of the drug approval process. An overview of the different areas of influence of the dissociation constant is shown in figure 2.7, which was taken from Manallack *et al.* (2013)[3].

The importance of the dissociation constant has already been explained in detail in several publications[1–7].

A practical example of the relevance of the pK_a value within drug discovery was presented by Castro *et al.* (1998)[46]. In their case, the problem of poor oral absorption of a lead structure was to be solved, for which a series of drug candidates or variants of the lead structure were synthesized and measured. It was found that a basic amino group with a pK_a value of approximately 9.7 could cause this absorption problem. Using fluorinated analogs, the pK_a value of the aforementioned group could be reduced to as low as 8.0, thus significantly improving oral absorption. As a second example, pK_a value also plays a key role in blood-brain barrier (BBB) permeability. Fan *et al.* (2010)[47] showed that acidic compounds were the least permeable to the BBB, whereas basic or neutral compounds were on average significantly better at crossing the BBB.

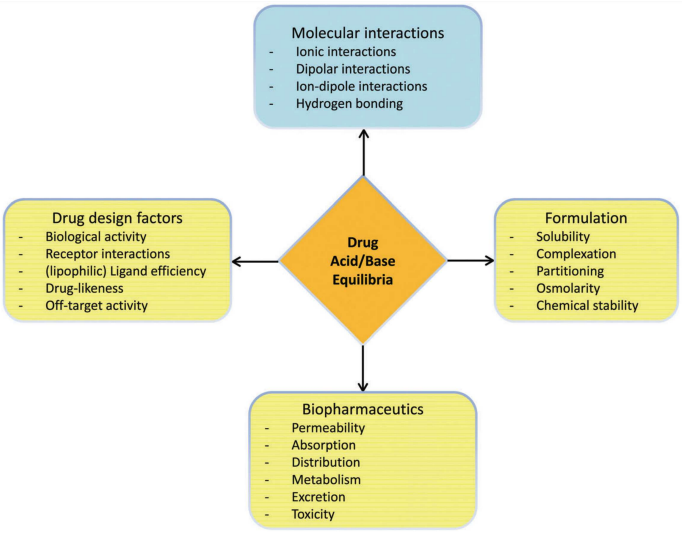


Figure 2.7: Overview of the different areas of influence of the dissociation constant taken from Manallack *et al.*[3].

2.3.2 pK_a prediction methods

Prediction methods for pK_a values can be divided into two main groups: Empirical methods and quantum chemical methods. The group of empirical methods can be divided into the subgroups Linear Free-Energy Relationship (LFER), Quantitative Structure-Property Relationship (QSPR) and database lookups, among others[43]. In the following, QSPR, LFER and quantum chemical methods are elaborated in more detail.

Prediction programs for pK_a values based on QSPR usually use a regression model created by machine learning. In principle, any machine learning algorithm from

linear regression over random forest to neural networks can be used here; in practice, the use of neural networks has prevailed in recent years. However, the use of mathematical models instead of models trained by machine learning is also possible. The advantages of QSPR as described are the high speed in usability and the generally low resource requirements for prediction. At the same time, machine learning provides only approximations based on the underlying dataset and its accuracy[43, 48].

LFER is a type of statistical method that can be used to predict the pK_a values of a molecule. LFER is based on the idea that the free energy change of a chemical reaction is directly proportional to the difference in the pK_a values of the reactants and products. In other words, the larger the difference in pK_a values, the more energetically favorable the reaction will be. To use LFER to predict pK_a values, you first need to collect data on a number of known compounds and their pK_a values. This data is then used to fit a mathematical model that describes the relationship between the pK_a values and the free energy of the reaction. Once the model has been fitted, you can use it to predict the pK_a values of other compounds based on their chemical structure[43, 49–51].

Quantum chemical methods usually have a higher accuracy since they are based on the underlying quantum chemical principles of pK_a itself. This higher accuracy usually comes at the cost of lower speed compared to empirical methods. An example of pK_a prediction software based on quantum chemical methods is Jaguar by Schrödinger. Jaguar models a thermodynamic cycle by means of Density Functional Theory (DFT) calculations and thus calculates the change in Gibbs energy related to the dissociation of a proton from the molecule under consideration. Such methodologies use *ab initio* calculations for the aqueous and gas phases. Depending on the thermodynamic cycle used, several separate geometry optimizations are required per pK_a value prediction. To get the best of both worlds, i.e. quantum chemical and empirical methods, semi-empirical methods are of interest, where classical machine learning is used together with quantum chemical descriptors[43, 52, 53].

Finally, this section will address the broad field of predicting protein pK_a values. This involves calculating or predicting the pK_a values of individual or multiple amino acids within a protein. Unlike the calculation of pK_a values of small molecules, this process requires consideration of the entire protein system, which usually contains a large number of protonatable sites. Due to electrostatic interactions, the titratable groups can occur in 2^n protonation states. Usually, implicit solvent models (e.g. Word and Nicholls (2011)[54], Bashford and Karplus (1990)[55]) are used for the estimation of protein pK_a values. Such implicit solvent models compute the

total titration curve by summing up all charge states of the titratable sites interacting with this group. In literature, several definitions and approaches exist for the assignment of microstate and macrostate pK_a values[56–58]. Due to the difficulty of experimentally determining protein pK_a values, *in silico* approaches have been established. These approaches typically calculate the shift of the deprotonation free energy of an amino acid in solution based on the deprotonation free energy of the amino acid within the protein. There are two categories of methodologies: continuum (such as Poisson-Boltzmann and Generalized Born) and all-atom-based electrostatics molecular dynamics (MD) simulations. Constant-pH MD simulations are an important method in predicting protein pK_a values. This method combines the Generalized Born model with MD simulations, which allow dynamic changes in protonation states. This work focuses solely on the prediction of pK_a values for small molecules and will not discuss the complex field of protein pK_a calculation further[59–61].

2.3.3 Aim of this work

The goal of this work and related research is to develop a freely available pK_a predictor for small molecules. This should be open-source, easy to use and with an unrestrictive license for both research and commercial use. The prediction model should be based exclusively on experimental data and be reproducible, comprehensible and interpretable. The model should also not only be usable for simple molecules from the literature, but also applicable to more complex chemical spaces, for example from the pharmaceutical industry. For this purpose, cooperations with different pharmaceutical and software companies are aimed to improve and extend the data basis of the experimental pK_a values. The tool to be developed should be competitive with commercial products in its prediction quality and not only focus on the most basic and most acidic pK_a values. It should be able to recognize all titratable groups contained in the molecule and predict their pK_a values. Finally, the model should be evaluated with literature data as well as with data from the pharmaceutical partners. The ChemAxon Marvin[8] prediction tool should be used as a reference. The pK_a prediction tool to be developed should match or, if possible, exceed ChemAxon Marvin[8] in terms of functionality and prediction quality. This should provide the scientific community with a licence-free alternative to the licensed and commercial programs on the market.

To achieve this goal, suitable data sources must first be identified. For this purpose, freely available databases and publications shall be collected and evaluated. At the same time, cooperation with different pharmaceutical companies and software vendors should be requested and developed. The collected data must be validated,

standardized and combined, whereby special attention should be paid to data quality. As a starting point for the use of the data for machine learning, monoprotic structures shall be used and an evaluation of different machine learning algorithms shall be performed. Based on this, a concept for multiprotic pK_{a} prediction shall be developed and validated, focusing on empirical or semi-empirical methodologies. Finally, the final model shall be published freely available by means of an easy-to-use tool.

Chapter 3

Methods

3.1 Machine learning

3.1.1 Random Forest

Random Forest is a machine learning algorithm for building predictive models. Both classification and regression models can be trained with this algorithm. A model trained with Random Forest basically only consists of many individual binary decision trees. A single decision tree has a high variance, this effect is reduced by the large set of trees. The algorithm has a high training and prediction speed, since the processing of all decision trees can be parallelized, and is particularly suitable for high-dimensional data. Especially if you only have a small number of data points or molecules, but a large number of influence variables, for example molecular descriptors and fingerprint bits, you can still train meaningful models with Random Forest. This is because each individual decision tree only ever has to process an individual part of the influence variables and samples. The algorithm quantifies the importance of the individual influence variables and recognizes on its own which values have a particularly high influence on the target value. If a prediction is made, the model returns the target value predicted by most of the trees (classification model) or the averaged value of all trees (regression model). In the case of classification models, a probability value for the predicted class is also returned. This allows a statement to be made about the reliability of the prediction[62–64].

An additional advantage of Random Forest is its robustness. The few adjustable parameters themselves have only limited influence on the prediction performance, which is why almost optimal results can often already be achieved with the default settings, based on the performance of the algorithm and the underlying data basis. Notable parameters are the number of trees, the maximum depth of each tree and

the number of influence variables and data points that each individual tree should see. As the number of trees increases, the robustness of the model increases in particular, but the prediction quality reaches its saturation point early on. From here on, increasing the number of trees only leads to higher resource consumption for training, prediction and storage. The maximum depth limits the number of levels of each tree to a chosen value. This parameter is important to prevent overfitting and can therefore contribute to the generalisation of the model. Finally, the number of influence variables and data points per tree increases the diversity of the forest's decision trees. Usually, a randomly selected subset (without recurrences) with the size of the square root of the total number of features is used. The data points to be used per tree are also randomly selected over the entire training dataset, but here the size of the training dataset is usually maintained while the random subset being drawn with reclamation[62–64].

3.1.2 Support Vector Machine

Support Vector Machine (SVM) is a machine learning model that can be used to build predictive classification and regression models, sometimes also divided into Support Vector Regression (SVR) and Support Vector Classification (SVC). This algorithm is one of the supervised learning methods and is very powerful in high-dimensional domains, especially when the number of dimensions, i.e. the influence variables, is greater than the number of data points in the training dataset. In the latter case, it is important to choose the right kernel function and regularization terms to avoid overfitting. The basic aim of the algorithm is to form a hyperplane in the high-dimensional space of the influence variables in order to separate them as good as possible and thus divide them into classes. However, this only works if the objects in the hyperspace are linearly separable assuming the use of a linear kernel function. In real datasets, linear separation is usually not possible, which is why the kernel trick is used here. Here, the vector space and the data points in it are transferred into a higher-dimensional space. With an increasing number of dimensions, every vector set can be linearly separated at some point. In this higher-dimensional space, the separating hyperplane is then determined and this is then calculated back to the original lower-dimensional space with fewer dimensions. In the process, the hyperplane becomes a non-linear hyperplane, which may even be non-contiguous. Using other kernel functions like the polynomial, sigmoidal or radial basis function kernels can improve the predictive performance if the underlying prediction problem is not linear. Even customized kernels can be used which make SVMs also suitable for problems based on graphs or strings[65, 66].

If a regression model is trained, the hyperplane is not supposed to separate, but

to cover all data points in the hyperspace as optimally as possible within a defined tolerated error range. The basic algorithmic procedure, however, remains identical to its use as a classifier[65].

3.1.3 Artificial Neural Networks

Artificial neural networks, often simply called neural networks, describe a machine learning concept or algorithm that is based on the internal structure of the biological brain. In the simplest structure, a graph organized in layers is built up, in which the nodes represent individual neurons and the edges represent the connections between the neurons, i.e. the synapses. Layers can be divided into three categories: input layers which take the input values from a dataset, output layers which return the output of the network and hidden layers which are basically the layers between input and output layers. An exemplary visualization of such an architecture is depicted in figure 3.1 The "signals" or information transmitted via the edges consist of individual floating point numbers. The edges of the neural network have a weight in the form of a positive or negative floating point factor with which the signal is offset. In addition, a bias, i.e. a constant value, is usually included per node and the overall result, also per node, is transformed non-linearly by means of an activation function. In the process of training the neural network, the edge weights and the constant bias values are optimized to reach the desired target value(s) with defined input values via the mathematical calculations. The relationship of input values, weights, bias, activation function and output is depicted in figure 3.2[67, 68].

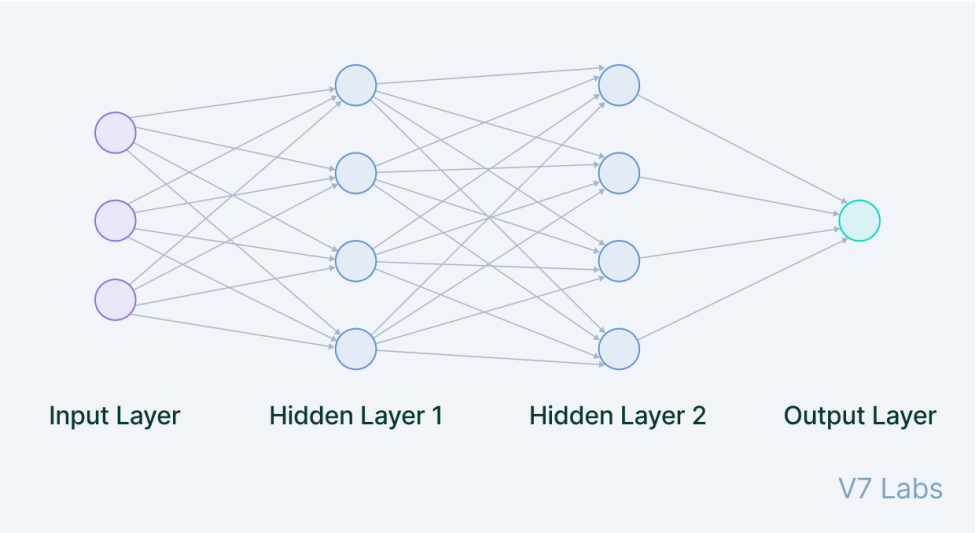


Figure 3.1: Exemplary neural network architecture: The figure shows the basic architecture of a fully-connected neural network[68].

There are many different activation functions which can be used. Examples are Sig-

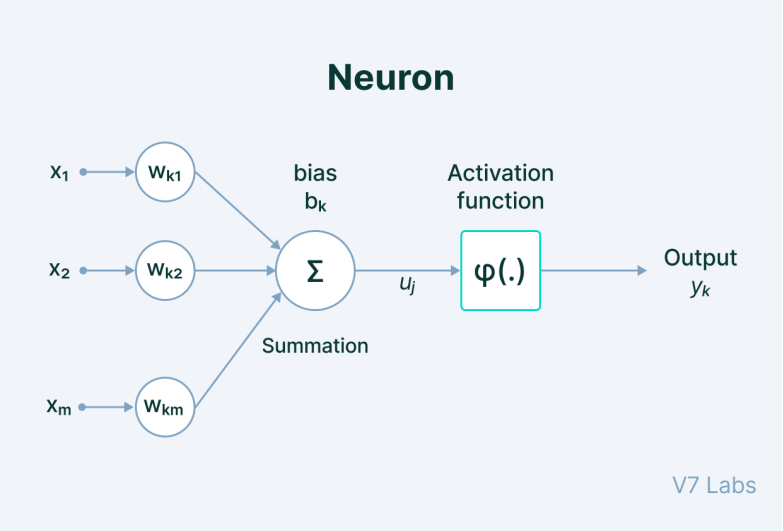


Figure 3.2: Exemplary depiction of a neuron in a neural network: Here, the relationship between input variables ($x_{1..n}$), weights ($w_{k1...m}$), bias (b_k), activation function ($\varphi(.)$) and output (y_k) is shown[68].

moid, hyperbolic tangent (tanh), Softmax or rectified linear unit (ReLU). Although linear activation functions can also be used in principle, a (small) neural network can only solve difficult problems with non-linear activation functions. The choice of the activation function(s) has also a significant influence on the training speed. Sigmoid (eq. 3.1), for example, is much more complex to calculate than ReLU (eq. 3.2)[67].

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

$$f(x) = \max(0, x) \tag{3.2}$$

The training of the network, i.e. the adjustment of the weights, is performed by a so-called optimizer. Optimizers in general are algorithms that adjust attributes (weights, learning rate, momentum, etc.) of a neural network to minimize the loss, i.e. the prediction error. Examples of optimizers are Stochastic Gradient Descent (SGD), Adagrad and Adaptive Moment Estimation (Adam). The loss can be calculated with different functions depending on the application area. For regression tasks, for example, the mean absolute error (MAE) (L1), mean square error (MSE) (L2) or root-mean-square error (RMSE) are suitable; for classification tasks, Cross-Entropy is usually used in conjunction with the Softmax activation function in the output layer[67, 69].

Another important part of neural network training is regularization. It should prevent the model from being overfitted, i.e. only memorizing the training data and not generalizing. Besides the regularization methods, which can be applied directly to

the training data, e.g. data augmentation, there are the methods dropout and early stopping as well as L1 and L2 regularization. With dropout, intermediate layers are inserted, which randomly set a specified proportion of the signals to 0. With early stopping, the training is terminated prematurely as soon as the prediction quality on an external validation dataset no longer improves but deteriorates. L1 and L2 regularization both add a penalty term to the loss functions, which can be adjusted by a factor λ. L1 regularization adds an absolute value of the magnitude, L2 regularization adds a squared magnitude of the coefficient[67, 69].

3.1.4 Graph Neural Networks

Graph Neural Networks (GNNs) represent a special form of neural networks. As the name suggests, they are designed to deal with graphs as input data. This property allows simplified learning without or with less information loss, since graphs do not have to be converted into a vector or similar. A graph is a data structure that consists of nodes that are connected via edges that have a direction. In a graph, nodes and edges can have attributes as additional information. We encounter graphs or graph-based data in many areas of everyday life. These include, for example, social networks, citation networks, or the prediction of purchasing behavior ("people who bought this product also bought that product"). In a citation network the nodes would represent publications while the edges represents the citation of one publication in another one. GNNs can be used in two different prediction modes: node-level predictions and graph-level predictions. With node-level predictions the network predicts a class or a feature for one or more nodes within the graph network. Graph-level predictions are more comparable to classical machine learning, where a graph represents a sample for which one or more target values are to be predicted. Here, the graph usually serves as a more useful and lossless input format[70, 71].

In particular, the supergroup of GNNs can be roughly distinguished into two distinct subgroups: The Message Passing Neural Networks (MPNNs) and the Graph Convolutional Neural Networks (GCNs). MPNNs can be described as aggregation and update functions that are repeated over several iterations while every iteration can be seen as one GNN layer. This method is especially used for node-level predictions where only a subset of the nodes lacks the target classes or values. In principle, GCNs function like classical convolutional neural networks, which have their strength especially in image recognition, processing or classification. Technically, an image is also a bidirectional graph, where the individual pixels are the nodes and the arrangement of the pixels is represented by the edges between neighboring pixels. As with images, in the case of GCNs, features of neighboring nodes are added together via the convolutional layer and multiplied by filter weights. This

identifies and extracts important or distinctive regions of a graph. Based on this basic approach, many different implementations of such layers exist, which may be more or less suitable depending on the application area[70–73].

3.2 Programming language and frameworks

3.2.1 Python

The Python scripting language was developed in the early 1990s by Guido van Rossum at the *Centrum voor Wiskunde en Informatica* in Amsterdam. Python is particularly characterized by its speed compared to other scripting languages, such as Perl, and offers the possibility to extend the already very comprehensive standard library in a simple way. Through the Python Application Programming Interface (API) it is also possible to write extensions in the C programming language and use them in Python. Thus, even machine-oriented and time-critical extensions can be developed and used for Python. Also, Python offers full support for object orientation and modularization. Python is licensed under the Python Software Foundation (PSF) license. This license was designed by the PSF and allows the Python interpreter to be integrated into and distributed with commercial applications at no license cost. This feature makes Python particularly interesting for industry. In addition, the scripting language is platform independent. It is possible to run Python programs without modifications on different operating systems, as long as a Python interpreter with all the required libraries is installed[74].

Python is ranked number 1 in the TIOBE Index in November 2022. It has thus replaced C as the most popular programming language for already several months now based on this particular index. The TIOBE Index is a programming language ranking based on the number of searches made by users in the 25 most popular search engines[75]. Python is also one of the most widely used programming languages in machine learning, data science and cheminformatics. Especially in computational chemistry, new tools in and libraries for Python are published regularly. Areas of application are, for example, the visualisation of chemical spaces, quantum chemistry, molecular modelling or virtual screening. Python is also used in web and internet development, for building platform independent graphical user interfaces and for supporting other programming languages in terms of build control, management and testing[76].

3.2.2 RDKit

RDKit is an open-source toolkit for tasks in the field of chemoinformatics. The algorithms and data structures are written in C++, but can be used in Python through a wrapper. Thus, RDKit achieves a very high execution speed. With RDKit, chemical structures can be processed in both two-dimensional and three-dimensional space. For instance, it is possible to calculate the three-dimensional structure of a molecule, to read in and manipulate molecular structures from different formats and to carry out complex alignments and substructure searches. Furthermore, more than 200 molecular descriptors can be calculated and several different fingerprint algorithms can be used[77].

With over 140 contributors (as of November 2022), RDKit is an actively developed and community-driven project that has become the standard library for cheminformatics questions in Python and C++. It has been published under the Berkeley Software Distribution (BSD) 3 Clause licence, which allows the software to be used and distributed commercially and in modified form. Comparable alternatives are, for example, Chemistry Development Kit (CDK) or OpenEye Toolkits, the former, however, with less functionality and only for Java, the latter only with a charged licence. Due to its wide range of functions, RDKit is used in many areas and current publications, such as for working with reaction data, in the area of preprocessing chemical data for machine learning or for virtual screening[77, 78].

3.2.3 Scikit-learn

The Python library Scikit-learn provides an extensive collection of machine learning algorithms. The development of this library was initialized by David Cournapeau and the first version was released in June 2007. Scikit-learn is still under active development by a large community on GitHub. Scikit-learn can be used to train both classification and regression models using a wide variety of methods, including Random Forest, Support Vector Machine, k-Nearest-Neighbors, and many more. The library also offers the possibility of training neural networks, whereby the setting options here, in contrast to libraries primarily oriented to neural networks such as PyTorch or Keras/TensorFlow, are strongly limited and only multilayer perceptrons are supported. In addition to Python, the framework is also based on NumPy, SciPy and Matplotlib. Through easy and intuitive use of Scikit-learn, complex models can already be created with only a few lines of code. The models are realized as objects within Python and can therefore also be exported and saved as a file. In addition to algorithms for creating predictive models using machine learning, Scikit-learn also offers methods for clustering, preprocessing and reducing the dimensions of data. In

each case, all common methods are supported. In addition, Scikit-learn can perform hyperparameter searches and cross-validations for predictive models to find the best settings, i.e. the best model, for the available data[79].

3.2.4 Pandas

The Python library Pandas provides data structures and operations for managing data and its analysis. Pandas first appeared in 2008 and was initially developed by Wes McKinney. With Pandas the DataFrame, which is known mainly from the programming language R, finds its way into the programming language Python. With this data structure data can be administered in a table structure, whereby the different columns can be of different data types. A number of analysis tools are available, such as the direct visualization of the data using Matplotlib. Additionally, it supports reading and writing various data formats like comma-separated values (CSV), Microsoft Excel, Structured Query Language (SQL) databases, Hierarchical Data Format 5 (HDF5) and many more. It is capable of database-like aggregation, transformation, merging and joining, as well as dealing with time series. Pandas is optimized for performance to be able to handle very big datasets efficiently[80, 81].

3.2.5 PyTorch and PyTorch Geometric

PyTorch is an open source machine learning framework for Python originally developed by Meta AI (formerly known as Facebook, Inc.). It is organized within the PyTorch Foundation which is again part of the Linux Foundation. PyTorch describes itself as an end-to-end machine learning framework that "enables fast, flexible experimentation and efficient production through a user-friendly frontend, distributed training and ecosystem of tools and libraries"[82]. PyTorch was developed as an alternative to TensorFlow/Keras and gives the developers the possibility to control and tune their machine learning pipeline including preprocessing, training and testing in a very detailed and low-level way. As of 2019 and based on the number of unique references at nine conferences about computer vision, natural language processing and general machine learning, PyTorch gained a lot of traction in the research community. From 2017 to 2019, the ratio of references rose steeply in PyTorch's favor, clearly replacing TensorFlow as the most used tool in the vast majority of the investigated conferences[10, 83].

PyTorch Geometric[11] (PyG) is a Python library for developing and training GNNs with PyTorch. The first release on GitHub was published in May 2018 by its author Matthias Fey. PyG is built on top of PyTorch and provides methods and classes for creating, reading and preprocessing graph data as well as deep learning on graphs

serving more than 50 different implemented neural network layers of various publications. Additionally, it brings many example or benchmark graph datasets together with data management classes, multi-GPU support and a detailed documentation to easily get started[11].

Chapter 4

Monoprotic pK_a Prediction

Experimental pK_a data were collected and analyzed through an extensive literature search, screening of several publicly available databases for chemistry data, and agreements with several collaborators. To get a first overview of the use of the data for machine learning and the corresponding prediction of pK_a values for small molecules, the compiled data were filtered to include only monoprotic molecules. The unique structure-property relationship given here allowed classical machine learning methods to be tested and compared first, without directly dealing with the complexity of multiple pK_a values per molecule in the case of multiprotic molecules, as well as the different number of target values per molecule depending on the number of titration sites.

The following chapters were previously published in this form in the publication *Baltruschat and Czodrowski* (2020)[9] in the journal *F1000Research* and successfully went through a peer review process. The content was adopted unchanged.

4.1 Introduction

The acid-base dissociation constant (pK_a) of a drug has a far-reaching influence on pharmacokinetics by altering the solubility, membrane permeability and protein binding affinity of the drug. Several publications summarize these findings in a very comprehensive manner[1–7]. An accurate estimation of pK_a values is therefore of utmost importance for successful drug design. Several (commercial and non-commercial) tools and approaches for small molecule pK_a prediction are available: MoKa[84] uses molecular interaction fields, whereas ACD/Labs Percepta Classic[85], ChemAxon Marvin[8] and Epik[86] make use of the Hammet-Taft equation. By means of Jaguar[22], a quantum-mechanical approach to pK_a prediction

becomes possible. Recently, the usage of neural nets for pK_a prediction became prominent[87–89]. In particular, the publication by Williams *et al.*[89] makes use of a publicly available dataset provided by the application DataWarrior[90] and provides a freely available pK_a prediction tool called OPERA.

As this article is part of a Python collection issue, we provide a pK_a prediction method entirely written in Python and make it available open source (including all data)[91]. Our tool computes the macroscopic pK_a value for a monoprotic compound. Our model solely differentiates between a base and acid based on the predicted pK_a value; i.e., we do not offer separate models for acids and bases. In addition to pK_a data from DataWarrior[90], we also employ pK_a data from ChEMBL[92]. As external validation sets, we use compound data provided by Novartis[15] and a manually curated dataset compiled from literature[14, 43, 93–95], which are not part of the training data.

4.2 Methods

4.2.1 Dataset preparation

A ChEMBL[92] web search was performed to find all assays containing pK_a measurement data. The following restrictions were made: it must be a physicochemical assay, the measurements must be taken from scientific literature, the assay must be in "small-molecule physicochemical format" and the organism taxonomy must be set to "N/A". This results in a list of 1 140 ChEMBL assays downloaded as comma-separated values (CSV) file. Using a Python script, the CSV file was read in and processed further, extracting all additional information required from an internally hosted copy of the ChEMBL database via Structured Query Language (SQL). Only pK_a measurements, i.e. ChEMBL activities, were taken into account that were specified as exact ("standard_relation" equals "=") and for which one of the following names was specified as "standard_type": "pka", "pka value", "pka1", "pka2", "pka3" or "pka4" (case-insensitive). Measured values for which the molecular structure was not available were also sorted out. The resulting 8 111 pK_a measured values were saved as structure-data file (SDF).

A flat file from DataWarrior[90] named "pKaInWater.dwar" was used in addition. In this case, the file was converted to an SDF only and contains 7 911 entries with valid molecular structures.

These datasets were concatenated for the purpose of this study and preprocessed as follows:

- Removal of all salts from molecules
- Removal of molecules containing nitro groups, Boron, Selenium or Silicium
- Filtering by Lipinski's rule of five (one violation allowed)
- Keeping only pK_a data points between 2 and 12
- Tautomer standardization of all molecules
- Protonation of all molecules at pH 7.4
- Keeping only monoprotic molecules regarding the specified pK_a range
- Combination of data points from duplicated structures while removing outliers

All steps up to filtering out all pK_a values outside the range of 2 to 12 were performed with Python and RDKit[77]. The QUACPAC[96] Tautomers tool from OpenEye was used for tautomer standardization and setting the protonation state to pH 7.4. The ChemAxon Marvin[8] tool was used to filter out the multiprotic compounds. It predicted the pK_a values of all molecules in the range 2 to 12 and then retained only those molecules where ChemAxon Marvin[8] did not predict more than one pK_a in that range.

The removal of the outliers is performed in two steps. First, before combining multiple measurements for the same molecules, all entries where the pK_a predicted by ChemAxon Marvin[8] differs from the experimental value by more than four log units are removed. All molecules were then combined on the basis of the canonical isomeric SMILES. In the second step, when combining several measured values of a molecule, all those values that deviate from the mean value by more than two standard deviations are removed. The remaining values are arithmetically averaged.

After all, 5 994 unique monoprotic molecules with experimental pK_a values remained. The distribution of pK_a values is given in figure 4.1. The same preprocessing steps were also performed on an external test dataset provided to us by Novartis[15] (280 molecules) and a manual curation (123 molecules) from literature[14, 43, 93–95]. The Novartis[15] dataset consists of 280 unique molecules with a molecular weight between 129 and 670 daltons (mean value 348.68, standard deviation 94.17). The calculated LogP values vary between -1.54 and 6.30 (mean value 3.01, standard deviation 1.41). The 280 molecules spread over 228 unique Murcko Scaffolds. The ten most common murcko scaffolds cover 15% of the molecules of the total dataset (42/280). A histogram of the pairwise comparison between the training set and the two external test sets (fingerprint: 4 096 bit MorganFeatures, radius=3) is given in figure 4.2.

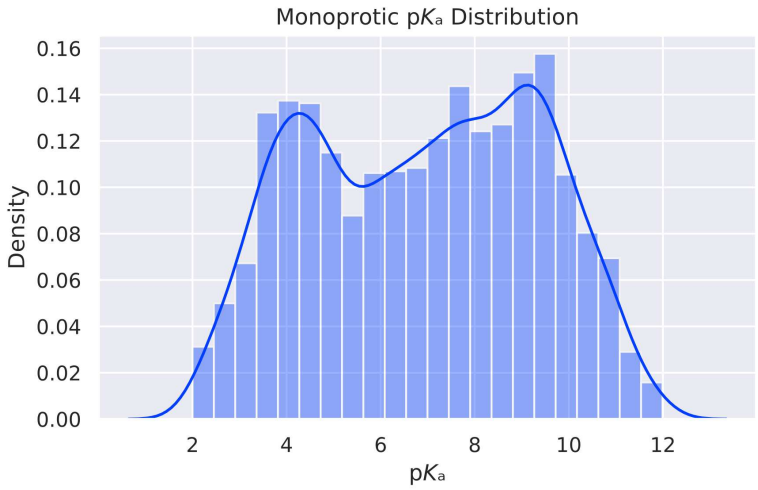


Figure 4.1: Distribution of the individual pK_a values.

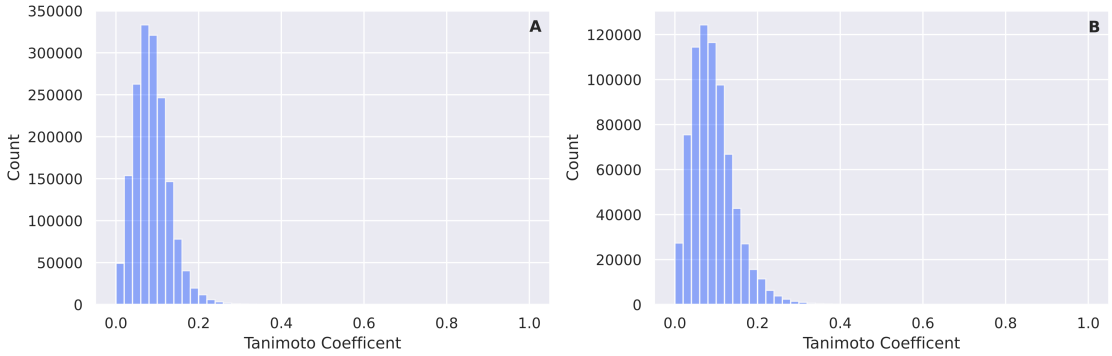


Figure 4.2: (**A**) Pairwise comparison between the training set and the Novartis[15] test set (Fingerprint: 4 096 bit MorganFeatures radius=3). (**B**) Pairwise comparison between the training set and the test set compiled by manual curation (Fingerprint: 4 096 bit MorganFeatures radius=3).

4.2.2 Learning

First, to simplify cross-validation, a class "CVRegressor" was defined, which can serve as a wrapper for any regressor implementing the Scikit-Learn[79] interface. This class simplifies cross-validation itself, training and prediction with the cross-validated model. Next, 196 of the 200 available RDKit descriptors ("MaxPartialCharge", "MinPartialCharge", "MaxAbsPartialCharge" and "MinAbsPartialCharge" were not used because they are computed as "NaN" for many molecules), and a 4 096-bit long MorganFeature fingerprint and radius 3 were calculated for the training dataset. Random Forest (RF), Support Vector Regression (SVR) (two configurations), Multi-layer Perceptron (MLP) (three configurations) and XGradientBoost (XGB) were used as basic regressors. Unless otherwise specified, the

Scikit-Learn default parameters (version 0.22.1) were used. For the RF model, only the number of trees was increased to 1 000. For SVR the size of the cache was increased to 4 096 megabytes in the first configuration, but this only increases the training speed and has no influence on the model quality. In the second configuration the parameter "gamma" was additionally set to the value "auto". For MLP in the first configuration the number of hidden layers was increased to two and the number of neurons per layer to 500. In the second configuration, early stopping was additionally activated, where 10% of the training data was separated as validation data. If the error of the validation data did not improve by more than 0.001 over ten training epochs, the training is stopped early to avoid overtraining. In the third configuration three hidden layers with a size of 250 neurons each were used with early stopping still activated. For XGB the default parameters of the used library XGBoost[97] (version 0.90) were applied. The training of RF, MLP and XGB was parallelized on 12 central processing unit (CPU) cores and the generation of the folds for cross-validation as well as the training itself were random seeded to a value of 24 to ensure reproducibility. This resulted in a total of seven different machine learning configurations.

Six different descriptor/fingerprint combinations were also tested. First only the RDKit descriptors, followed by only the fingerprints and finally both combined. Additionally, all three combinations were tested again in a standardized form (z-transformed). As a result, 42 combinations of regressor and training data configurations were compared.

A 5-fold cross-validation was performed for all configurations, which were evaluated using the mean absolute error (MAE), root-mean-square error (RMSE) and the empirical coefficient of determination (R^2). After training was completed for all configurations, two external test datasets, which do not contain training data, were used to re-validate each trained cross-validated model. Here, MAE, RMSE, and R^2 were also calculated as statistical quality measures. To ensure that no training data was contained in the test datasets, the canonical isomeric Simplified Molecular-Input Line-Entry System (SMILES) were checked for matches in both training and test datasets and corresponding hits were removed from the test datasets.

4.2.3 Implementation

The following Python dependencies have to be met: Python $\geq$ 3.7, NumPy $\geq$ 1.18, Scikit-Learn $\geq$ 0.22, RDKit $\geq$ 2019.09.3, Pandas $\geq$ 0.25, XGBoost $\geq$ 0.90, JupyterLab $\geq$ 1.2, Matplotlib $\geq$ 3.1, Seaborn $\geq$ 0.9.

For the data preparation pipeline, ChemAxon Marvin[8] and OpenEye QUACPAC/-

Tautomers[96] are required. To use the provided prediction model with the included Python script, ChemAxon Marvin[8] is not required.

First of all a working Miniconda/Anaconda installation is needed. Miniconda can be downloaded at `https://conda.io/en/latest/miniconda.html`.

Now an environment named "ml_pka" with all needed dependencies can be created and activated with:

```
conda env create -f environment.yml
conda activate ml_pka
```

Alternatively, a new environment can be created manually without the environment.yml file:

```
conda create -n ml_pka python=3.7
conda activate ml_pka
```

In case of Linux or macOS:

```
conda install -c defaults -c rdkit -c conda-forge scikit-learn \
    rdkit xgboost jupyterlab matplotlib seaborn
```

In case of Windows:

```
conda install -c defaults -c rdkit scikit-learn rdkit jupyterlab \
    matplotlib seaborn
pip install xgboost
```

4.2.4 Operation

Prediction pipeline

To use the data preparation pipeline the repository folder hast to be entered and the created conda environment must be activated. Additionally the ChemAxon Marvin[8] commandline tool `cxcalc` and the QUACPAC[96] commandline tool `tautomers` have to be added to the `PATH` variable.

Also the environment variables `OE_LICENSE` (containing the path to the OpenEye license file) and `JAVA_HOME` (referring to the Java installation folder, which is needed for `cxcalc`) have to be set.

After preparation a small usage information can be displayed with `bash run_pipeline.sh -h`. Exemplary call:

```
bash run_pipeline.sh \
    --train datasets/chembl25.sdf \
    --test datasets/novartis_cleaned_mono_unique_notraindata.sdf
```

Prediction tool

First of all the repository folder has to be entered and the created conda environment must be activated. To use the prediction tool the machine learning model has to be retrained. To do so the training script should be called, it will train the 5-fold cross-validated Random Forest machine learning model using 12 cpu cores. If the number of cores has to be adjusted the `train_model.py` can be edited by changing the value of the variable `EST_JOBS`.

```
python train_model.py
```

To use the prediction tool with the trained model QUACPAC/Tautomers have to be available as mentioned in the section above. Now the python script can be called with an SDF and an output path:

```
python predict_sdf.py my_test_file.sdf my_output_file.sdf
```

It should be noted that this model was built for monoprotic structures regarding a pH range of 2 to 12. If the model is used with multiprotic structures, the predicted values will probably not be correct.

4.3 Results

4.3.1 Different experimental methods

One crucial point in the field of pK_a measurements (and its usage for pK_a predictions) was linked to the different experimental methods[26, 98]. Based on the Novartis[15] set, the correlation between capillary electrophoresis and potentiometric measurements (for 15 data points) was convincing enough (MAE= 0.202, RMSE= 0.264, R^2= 0.981) for us to combine pK_a measurements from these different experimental methods (see figure 4.3).

We also compared the pK_a values of 187 monoprotic molecules contained in both the ChEMBL and DataWarrior datasets. Due to the missing annotation, it remained unclear if different experimental methods were used or multiple measurements with the same experimental method have been performed (or a mixture of both). Either way, this comparison was an additional proof-of-concept that the ChEMBL and DataWarrior pK_a data sources can be combined after careful curation. The afore-

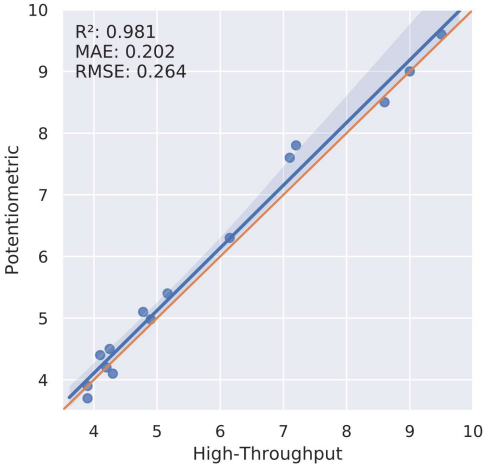


Figure 4.3: Correlation of Novartis[15] compounds measured in potentiometric and high-throughput (capillary electrophoresis) set-up.

mentioned intersection is given in figure 4.4. The correlation coefficient between the annotated pK_a values for these two datasets R^2 was 0.949, the MAE was 0.275, and the RMSE was 0.576.

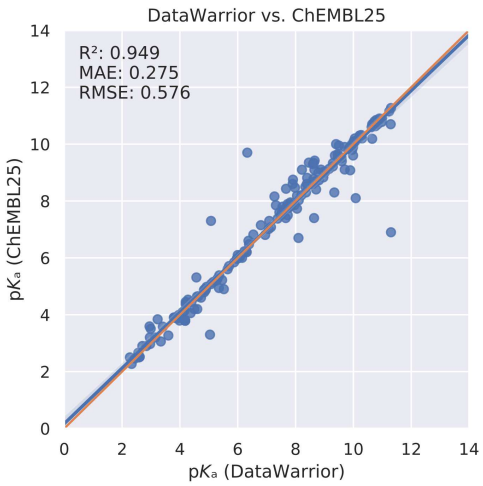


Figure 4.4: Intersection between ChEMBL[92] and DataWarrior[90] datasets.

The compounds for which the pK_a values between the different sources deviate by more than two units are as follows:

- Hydralazine
  - pK_a (DataWarrior) 5.075
  - pK_a (ChEMBL25) 7.3

- Edaravone
  - pK_{a} (DataWarrior) 11.3
  - pK_{a} (ChEMBL25) 6.9
- Trifluoromethanesulfonamide
  - pK_{a} (DataWarrior) 6.33
  - pK_{a} (ChEMBL25) 9.7

Since the annotation about the experimental settings is not given in the DataWarrior file, we can only hypothesize that these differences are due to the different experimental settings.

4.3.2 Machine learning

The statistics for a five-fold cross-validation are given in table 4.1. In terms of the mean absolute error, a random forest with scaled MorganFeatures (radius=3) and descriptors gave the best performing model (MAE= 0.682, RMSE= 1.032, R^2= 0.82). For the two external test sets (see table 4.2), a random forest with MorganFeatures (radius=3) gave the best model.

- Novartis[15]: MAE= 1.147, RMSE= 1.513, R^2= 0.569
- LiteratureCompilation[14, 43, 93–95]: MAE= 0.532, RMSE= 0.785, R^2= 0.889

The predictive performance for ChemAxon Marvin[8] and the OPERA tool[89] were as follows:

- Novartis[15]
  - ChemAxon Marvin[8]: MAE= 0.856, RMSE= 1.166, R^2= 0.744
  - OPERA[89]: MAE= 2.274, RMSE= 3.059, R^2= -0.754
- LiteratureCompilation[14, 43, 93–95]
  - ChemAxon Marvin[8]: MAE= 0.566, RMSE= 0.865, R^2= 0.866
  - OPERA[89]: MAE= 1.737, RMSE= 2.182, R^2= 0.124

This showed that our model had a slightly better performance than ChemAxon Marvin[8] for the LiteratureCompilation[14, 43, 93–95], but ChemAxon Marvin[8] performed better for the Novartis[15] dataset. For both datasets, our models[91] had a better predictive performance than the OPERA[89] tool. Since some molecules

had to be omitted for prediction with OPERA[89] due to none or multiple predicted pK_a values, no consistent significance test could be performed for all comparisons.

Model (seed=24)	Train Configuration	Cross-Validation					
		MAE (mean)	MAE (std)	RMSE (mean)	RMSE (std)	R^2 (mean)	R^2 (std)
Random Forest n_estimators=1000	Desc (196 RDKit)	0.718	0.022	1.077	0.021	0.804	0.010
	FCFP6 (4096 bit)	0.708	0.021	1.094	0.029	0.797	0.008
	Desc + FCFP6	0.683	0.017	1.032	0.013	0.820	0.005
	Desc (196 RDKit, scaled)	0.717	0.022	1.076	0.022	0.804	0.011
	FCFP6 (4096 bit, scaled)	0.708	0.021	1.094	0.029	0.797	0.008
	Desc + FCFP6 (scaled)	0.682	0.017	1.032	0.013	0.820	0.005
Support Vector Machine	Desc (196 RDKit)	2.100	0.037	2.436	0.035	-0.004	0.004
	FCFP6 (4096 bit)	0.851	0.025	1.240	0.035	0.740	0.012
	Desc + FCFP6	2.100	0.037	2.436	0.035	-0.004	0.004
	Desc (196 RDKit, scaled)	0.876	0.033	1.282	0.047	0.722	0.015
	FCFP6 (4096 bit, scaled)	1.090	0.034	1.466	0.041	0.637	0.014
	Desc + FCFP6 (scaled)	1.020	0.037	1.400	0.047	0.668	0.016
Support Vector Machine gamma='auto'	Desc (196 RDKit)	2.016	0.042	2.362	0.039	0.056	0.009
	FCFP6 (4096 bit)	1.612	0.031	1.926	0.033	0.373	0.007
	Desc + FCFP6	1.642	0.061	2.052	0.060	0.288	0.027
	Desc (196 RDKit, scaled)	0.882	0.035	1.288	0.048	0.719	0.016
	FCFP6 (4096 bit, scaled)	1.090	0.034	1.465	0.041	0.637	0.014
	Desc + FCFP6 (scaled)	1.019	0.037	1.400	0.047	0.669	0.016
Multilayer Perceptron hidden_layer_sizes=(500, 500)	Desc (196 RDKit)	#NA	#NA	#NA	#NA	#NA	#NA
	FCFP6 (4096 bit)	0.866	0.025	1.270	0.047	0.727	0.019
	Desc + FCFP6	#NA	#NA	#NA	#NA	#NA	#NA
	Desc (196 RDKit, scaled)	0.726	0.018	1.102	0.050	0.794	0.022
	FCFP6 (4096 bit, scaled)	1.037	0.045	1.457	0.057	0.640	0.024
	Desc + FCFP6 (scaled)	0.968	0.032	1.383	0.040	0.677	0.014
Multilayer Perceptron hidden_layer_sizes=(500, 500) early_stopping=True	Desc (196 RDKit)	#NA	#NA	#NA	#NA	#NA	#NA
	FCFP6 (4096 bit)	0.894	0.024	1.297	0.040	0.715	0.016
	Desc + FCFP6	#NA	#NA	#NA	#NA	#NA	#NA
	Desc (196 RDKit, scaled)	0.768	0.034	1.161	0.090	0.770	0.038
	FCFP6 (4096 bit, scaled)	1.031	0.037	1.447	0.057	0.645	0.026
	Desc + FCFP6 (scaled)	0.984	0.029	1.404	0.035	0.666	0.017
Multilayer Perceptron hidden_layer_sizes=(250, 250, 250) early_stopping=True	Desc (196 RDKit)	#NA	#NA	#NA	#NA	#NA	#NA
	FCFP6 (4096 bit)	0.869	0.023	1.265	0.039	0.729	0.016
	Desc + FCFP6	#NA	#NA	#NA	#NA	#NA	#NA
	Desc (196 RDKit, scaled)	0.775	0.008	1.158	0.033	0.773	0.013
	FCFP6 (4096 bit, scaled)	1.026	0.038	1.455	0.053	0.642	0.022
	Desc + FCFP6 (scaled)	0.973	0.035	1.388	0.053	0.674	0.023
XGBoost	Desc (196 RDKit)	1.020	0.014	1.353	0.021	0.691	0.007
	FCFP6 (4096 bit)	1.094	0.027	1.423	0.036	0.657	0.011
	Desc + FCFP6	1.018	0.010	1.346	0.022	0.694	0.005
	Desc (196 RDKit, scaled)	1.020	0.014	1.353	0.021	0.691	0.007
	FCFP6 (4096 bit, scaled)	1.094	0.027	1.423	0.036	0.657	0.011
	Desc + FCFP6 (scaled)	1.018	0.010	1.346	0.022	0.694	0.005

Table 4.1: Model performances based on cross-validation: The two best and worst performing models are highlighted in green and red. For those neural networks where the values were specified as "not available" ("#NA"), the weights could not be optimized properly due to the large value range of the RDKit descriptors, so training failed here.

Table 4.2: Model performances based on external testsets: The two best and worst performing models are highlighted in green and red. For those neural networks where the values were specified as "not available" ("#NA"), the weights could not be optimized properly due to the large value range of the RDKit descriptors, so training failed here. (*) For OPERA 6 molecules from LiteratureCompilation and 31 molecules from Novartis were left out because OPERA predicted either two or zero pK_a values.

Model (seed=24)	Train Configuration	Novartis			LiteratureCompilation		
		MAE	RMSE	R^2	MAE	RMSE	R^2
Random Forest n_estimators=1 000	Desc (196 RDKit)	1.259	1.607	0.513	0.689	0.979	0.828
	FCFP6 (4 096 bit)	1.147	1.513	0.569	0.532	0.785	0.889
	Desc + FCFP6	1.200	1.532	0.558	0.628	0.884	0.860
	Desc (196 RDKit, scaled)	1.259	1.607	0.513	0.688	0.979	0.828
	FCFP6 (4 096 bit, scaled)	1.147	1.513	0.569	0.532	0.785	0.889
	Desc + FCFP6 (scaled)	1.198	1.531	0.558	0.628	0.884	0.860
Support Vector Machine	Desc (196 RDKit)	2.177	2.451	-0.132	2.180	2.441	-0.070
	FCFP6 (4 096 bit)	1.423	1.732	0.435	0.688	0.981	0.827
	Desc + FCFP6	2.177	2.451	-0.132	2.180	2.441	-0.070
	Desc (196 RDKit, scaled)	1.382	1.735	0.433	0.772	1.058	0.799
	FCFP6 (4 096 bit, scaled)	1.771	2.035	0.219	1.115	1.422	0.637
	Desc + FCFP6 (scaled)	1.746	2.015	0.235	1.044	1.345	0.675
Support Vector Machine gamma='auto'	Desc (196 RDKit)	2.162	2.428	-0.111	1.921	2.242	0.097
	FCFP6 (4 096 bit)	1.686	1.932	0.297	1.429	1.670	0.499
	Desc + FCFP6	2.161	2.442	-0.124	1.611	2.004	0.279
	Desc (196 RDKit, scaled)	1.378	1.732	0.435	0.766	1.049	0.802
	FCFP6 (4 096 bit, scaled)	1.770	2.034	0.220	1.114	1.421	0.637
	Desc + FCFP6 (scaled)	1.744	2.013	0.236	1.043	1.343	0.676
Multilayer Perceptron hidden_layer_sizes=(500, 500)	Desc (196 RDKit)	#NA	#NA	#NA	#NA	#NA	#NA
	FCFP6 (4 096 bit)	1.414	1.773	0.407	0.852	1.169	0.755
	Desc + FCFP6	#NA	#NA	#NA	#NA	#NA	#NA
	Desc (196 RDKit, scaled)	1.318	1.634	0.497	0.688	0.942	0.841
	FCFP6 (4 096 bit, scaled)	1.627	2.033	0.221	1.102	1.569	0.558
	Desc + FCFP6 (scaled)	1.542	1.941	0.290	1.001	1.427	0.634
Multilayer Perceptron hidden_layer_sizes=(500, 500) early_stopping=True	Desc (196 RDKit)	#NA	#NA	#NA	#NA	#NA	#NA
	FCFP6 (4 096 bit)	1.404	1.772	0.408	0.846	1.154	0.761
	Desc + FCFP6	#NA	#NA	#NA	#NA	#NA	#NA
	Desc (196 RDKit, scaled)	1.298	1.626	0.502	0.701	0.936	0.843
	FCFP6 (4 096 bit, scaled)	1.611	2.028	0.225	1.141	1.575	0.554
	Desc + FCFP6 (scaled)	1.605	1.998	0.248	0.987	1.365	0.665
Multilayer Perceptron hidden_layer_sizes=(250, 250, 250) early_stopping=True	Desc (196 RDKit)	#NA	#NA	#NA	#NA	#NA	#NA
	FCFP6 (4 096 bit)	1.363	1.717	0.445	0.860	1.164	0.757
	Desc + FCFP6	#NA	#NA	#NA	#NA	#NA	#NA
	Desc (196 RDKit, scaled)	1.354	1.705	0.452	0.777	1.057	0.799
	FCFP6 (4 096 bit, scaled)	1.584	1.989	0.254	1.053	1.468	0.613
	Desc + FCFP6 (scaled)	1.581	1.963	0.274	0.953	1.352	0.672
XGBoost	Desc (196 RDKit)	1.367	0.453	1.704	0.819	0.806	1.040
	FCFP6 (4 096 bit)	1.280	0.503	1.624	0.782	0.823	0.992
	Desc + FCFP6	1.293	0.495	1.637	0.774	0.822	0.995
	Desc (196 RDKit, scaled)	1.367	0.453	1.704	0.819	0.806	1.040
	FCFP6 (4 096 bit, scaled)	1.280	0.503	1.624	0.782	0.823	0.992
	Desc + FCFP6 (scaled)	1.293	0.495	1.637	0.774	0.822	0.995
ChemAxon Marvin (V20.1.0)		0.856	1.166	0.744	0.566	0.865	0.866
OPERA (V2.5)*		2.274	3.059	-0.754	1.737	2.182	0.124

4.4 Discussion and conclusions

The developed model offers the possibility to predict pK_a values for monoprotic molecules with good accuracy. However, since the model has been trained exclusively with monoprotic molecules, only monoprotic molecules can be predicted properly. In this respect the model is limited. Nevertheless, the results show that the performance for monoprotic molecules can compete with the performance of existing prediction tools. The good performance of ChemAxon Marvin[8] on the Novartis[15] set is interesting to note: the RMSE was almost 0.4 units better than our top performing model. This could be because the training set of ChemAxon Marvin[8] is much larger than our own training set. This provides a better foundation for the training of the ChemAxon Marvin[8] model. In contrast, ChemAxon Marvin[8] performed slightly worse than our top model on the LiteratureCompilation. The OPERA tool performed significantly worse than our model on both external test sets. We assume that the addition of 2 470 ChEMBL pK_a datapoints to our training set which were not part of the OPERA training set led to this drop in predictive performance. In addition, the preprocessing of the data was performed differently by OPERA in comparison to our preprocessing procedure.

As next step for the enhancement and improvement of our pK_a prediction model[91], we are currently expanding it to multiprotic molecules. We are also investigating the impact of different neural net architectures and types (such as graph neural nets) and the development of individual models for acids and bases. From a chemistry perspective, an analysis of pK_a effects of different functional groups (e.g. by means of matched molecular pairs analysis) is an on-going effort for a future publication.

4.5 Data availability

Zenodo: czodrowskilab/Machine-learning-meets-pKa article. `https://doi.org/10.5281/zenodo.366224517`[91].

The following datasets were used in this study:

- `AvLiLuMoVe.sdf` - Manually combined literature pK_a data[14, 43, 93–95].
- `chembl25.sdf` - Experimental pK_a data extracted from ChEMBL25[92].
- `datawarrior.sdf` - pK_a data shipped with DataWarrior[90].
- `combined_training_datasets_unique.sdf` - Preprocessed and combined data from datasets chembl25.sdf and datawarrior.sdf, used as training dataset.

- `AvLiLuMoVe_cleaned_mono_unique_notraindata.sdf` - used as external testset.
- `novartis_cleaned_mono_unique_notraindata.sdf` - inhouse dataset provided by Novartis[15], used as external testset.

The datasets are also available at
`https://github.com/czodrowskilab/Machine-learning-meets-pKa`.

License: Massachusetts Institute of Technology (MIT) license.

4.6 Software availability

The source code is available at
`https://github.com/czodrowskilab/Machine-learning-meets-pKa`.

Archived source code at time of publication:
`https://doi.org/10.5281/zenodo.366224517`[91].

License: MIT license.

4.7 Acknowledgments

Ed Griffen (MedChemica) is acknowledged for his investigations on our initial dataset which revealed some wrongly annotated data points. We thank Bob Clark, Eric Jamois and Michael Lawless (Simulations Plus) for their support and advise in terms of data curation. Alpha Lee and Matt Robinson (Cambridge University) are appreciated for fruitful discussions.

Chapter 5

Multiprotic pK_a Processing

After the extensive evaluation of different prediction methods and preprocessing for monoprotic molecules and the related analysis of the underlying datasets, the next step should be to work with multiprotic molecules. In the case of monoprotic molecules, experimentally there is a clear structure-value relationship. In the case of several available experimentally determined pK_a values, these can be combined by calculating the arithmetic mean as an example. In multiprotic molecules, several titratable groups are present in each molecule. For this reason, simple averaging is not possible without additional information about the assignment of the respective pK_a value to its titratable group. Unfortunately, the information about the assignment of the experimentally determined pK_a values to the respective titration site is not available in almost all datasets. This makes an assignment prior to the calculation of the pK_a values necessary. In our approach, we rely on the assignment by the external tool ChemAxon Marvin[8] (CXM). Therefore, an external post-hoc validation of this assignment is crucial. Additionally, such an assignment of the values is additionally relevant to improve the quality of the finally trained prediction models by including this information in the training process. At this stage, an extensive post-hoc analysis with molecules where the titration site and pK_a values are clear from the experiment is essential. This chapter and the methods it describes therefore serve as a preprocessing of the datasets used for the machine learning described in the upcoming main chapter 6.

With this the tool Multiprotic pK_a Processor (MPP) was developed. MPP performs all necessary preprocessing steps, identifies and locates titratable groups, and makes all remaining preparatory steps to enable a jump-start for machine learning based on the processed datasets. Additionally, a detailed analysis regarding the titratable groups of the molecules in the given datasets is performed. MPP also provides

a SMILES Arbitrary Target Specification (SMARTS) pattern list representing the most frequently occurring titratable groups which is generated based on the input datasets.

5.1 Datasets

A dataset compilation consisting of 16 individual datasets was used. These include data from public databases, such as ChEMBL[99] and DataWarrior[90], data from the Statistical Assessment of the Modeling of Proteins and Ligands (SAMPL) challenges[16, 17, 100], data from experimental measurements in our own laboratory, data that could be extracted from various publications as well as data kindly provided by Idorsia Pharmaceuticals Ltd.[101], Roche Pharma AG[102], Novartis Pharma AG[15] and OpenEye Scientific Software[103]. A list of the datasets and their composition can be found in table 5.1. Unfortunately, not all datasets are publicly available, as they are subject to certain licensing conditions or confidentiality agreements.

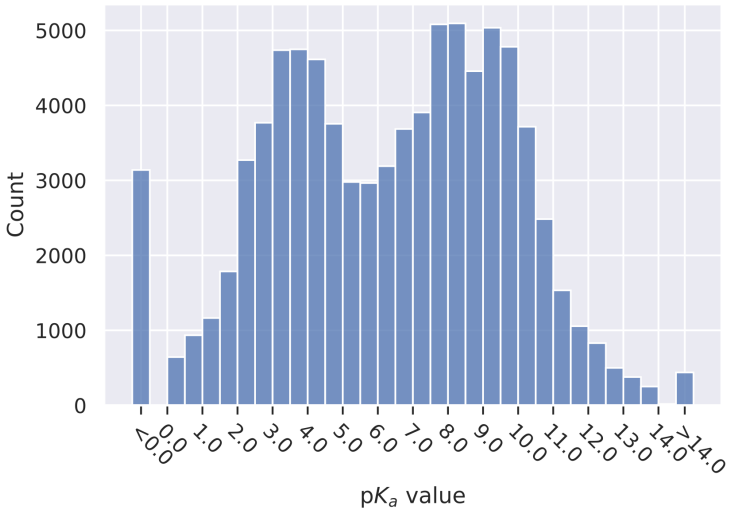


Figure 5.1: Distribution of pK_a values in the combined dataset

To illuminate the composition and properties of the entire dataset compilation, the distribution of the pK_a values and the measurement temperatures were analysed, as well as the chemical space was visualised. Figure 5.1 shows the distribution of the pK_a values. Here, it can be seen that the majority of the pK_a values are in the range between 2 and 12, but there is also a considerable amount of pK_a values that are smaller than zero. The pK_a measurements were carried out at different temperatures. pK_a values can be significantly influenced by the temperature[104]. Therefore, the prime focus is on measurements at room temperature, i.e. between

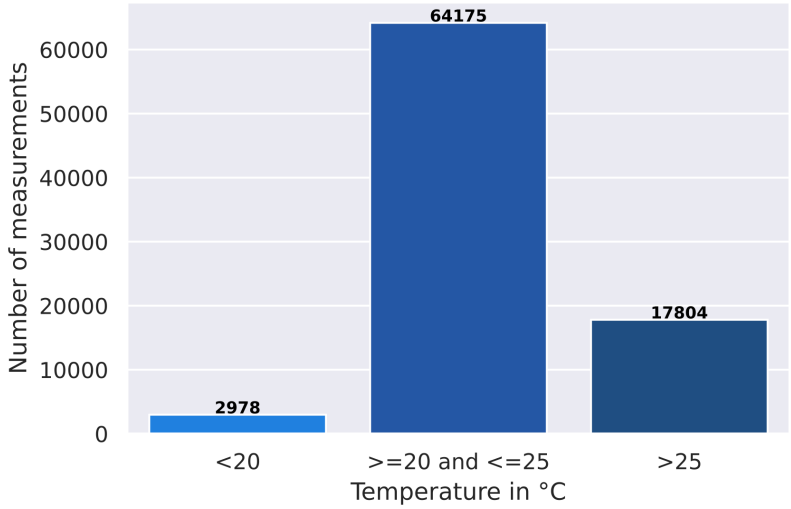


Figure 5.2: Distribution of temperatures at which the pK_a values in the combined dataset were determined

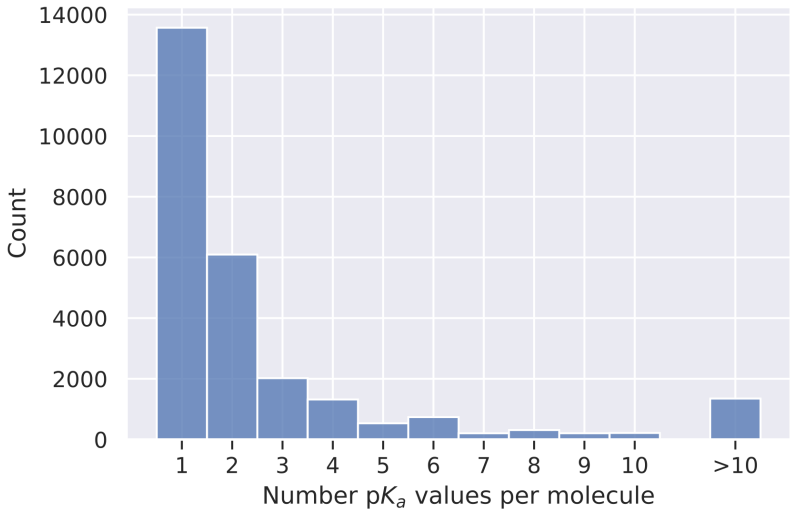


Figure 5.3: Distribution of the number of pK_a values per unique molecule in the combined dataset

20 and 25°C. Figure 5.2 shows the distribution of the temperatures at which the pK_a values were determined. Here it is clear that most of the measurements were done at room temperature, only a few measurements were done below 20°C. Measurements above the 25°C mark are more frequently included in the total dataset, in particular measurements that were performed at human body temperature. During the compilation of the different datasets, it was noticed that in several cases information about the temperature and solvent where not provided. In these cases 25°C and water were specified for temperature and solvent, respectively.

In figure 5.3 the distribution of the number of pK_a values per unique molecule is

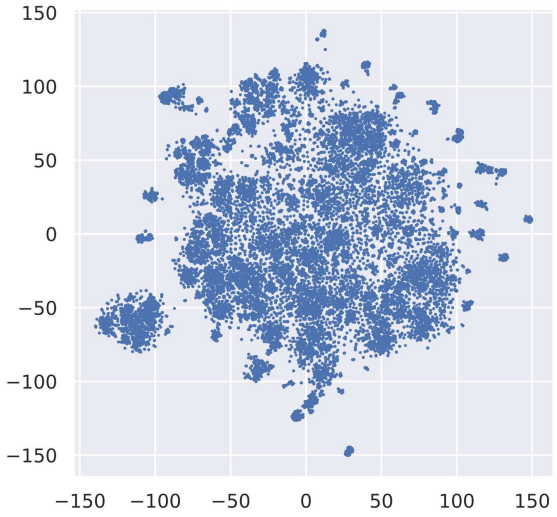


Figure 5.4: Visualized chemical space of the combined dataset: For the visualization, 4 096 bit Morgan fingerprints with radius three were used. A principle component analysis (PCA) was performed to reduce the dimension to 50, then a t-distributed stochastic neighbour embedding (t-SNE) was done to reduce the dimension further to two. The resulting two dimensions were plotted.

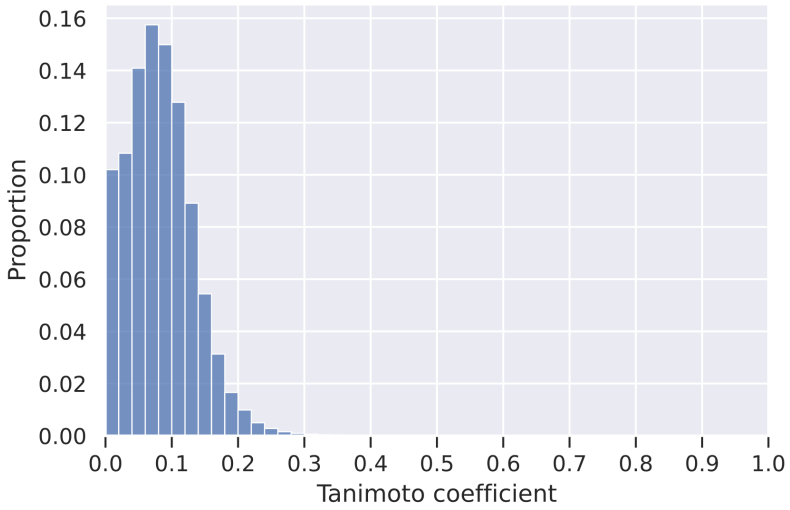


Figure 5.5: Distribution of Tanimoto coefficients of all combinations of molecules from the combined dataset

shown. Many of the measurements for one molecule are identical or similar; it should be mentioned that they may originate from different data sources or were measured at different temperatures. It can be seen that for the majority of the molecules in our dataset only one pK_a value was determined. This leads to the assumption that it is a monoprotic molecule, but of course this does not necessarily needs to be the case.

Name	pK_a values	Unique molecules	Source
Alkhzem *et al.*	24	5	[56]
ChEMBL 29	9 809	7 508	[99]
Czodrowski Lab	252	145	
DataWarrior	7 909	6 296	[90]
Hunt *et al.*	2 481	2 272	[105]
Idorsia	127	63	[101]
Literature Compilation	1 754	1 329	[1, 43–45, 95, 106–108]
Novartis	1 025	646	[15]
OpenEye	55 119	11 119	[103]
Roche 1	1 762	1 734	[102]
Roche 2	2 876	2 136	[102]
SAMPL6	31	24	[16]
SAMPL7	20	20	[17]
SAMPL8	25	21	[100]
Settimo *et al.*	610	503	[14]
Yu *et al.*	1 133	1 112	[109]
Summary	84 957	26 568	

Table 5.1: List of all datasets used in this contribution: The dataset entries were standardised beforehand by removing salts, neutralising charges and applying tautomer canonisation. Entries for which one of the standardisation steps failed as well as invalid molecular structures were removed.

Finally, the chemical space as well as the diversity of the dataset were examined. The chemical space is depicted in figure 5.4. For this purpose, a 4 096 bit long Morgan fingerprint with radius 3 was calculated for each molecule. Next, a PCA was performed to reduce the dimensions to 50. A t-SNE was then performed to further reduce the dimensions to 2. The resulting dimensions were plotted. In figure 5.5 the distribution of the Tanimoto coefficients of all combinations of molecules from our dataset is shown. As basis for the Tanimoto coefficients a full length Morgan fingerprint with radius 3 was used. It can be seen, that the Tanimoto coefficients are nearly all between 0.0 and 0.2 which underlines the diversity of our dataset.

5.2 Pipeline

The entire pipeline can be divided into four different stages and will be described in detail in the following paragraphs:

- Dataset preparation
- SMARTS tree generation
- Validation of localisation strategies
- Value assignment

Regarding the used terminology: The entire pipeline was developed with the aim that it can be right away used in conjunction with machine learning. In detail, this means that when the pipeline is initiated, the user can specify which datasets will be used in the next step as training and test datasets for machine learning. This pipeline must therefore be considered as part of the data preprocessing in the context of this very use. However, it should be made clear that no machine learning was applied in the course of the MPP. The use of the MPP processed datasets for machine learning is discussed in the chapter 6. In addition to the training and test dataset, two other dataset terminologies are used in the further course: the *grouping dataset*, on which the assignment of the pK_a values in the process of the pipeline is based, and the *validation dataset*, which is used for validating the strategy to locate the titratable groups. Since the more data available, the more precise the assignment of pK_a values, specific training and test datasets can be used for subsequent machine learning, for example, but a larger pK_a dataset can be used as grouping dataset for the assignment, which does not affect the training or the associated testing in the context of machine learning. For example, if only a small number of experimental pK_a values are available for which a machine learning model should be trained later, a large molecule dataset for which no experimental pK_a values are available, such as all the small molecules in the ChEMBL[99] database, could be used as a grouping dataset for reliable titration site localisation using SMARTS trees (see chapter 5.2.2). In this way, a sufficiently extensive SMARTS tree can be built without having to have experimental pK_a values for all molecules..

The described pipeline corresponds to the `mpp_pipeline.sh` shell script that was used to calculate the results of this manuscript but has only a few configuration options. To achieve all possible configuration options, the underlying Python scripts `gen_clean_unique_dataset.py` (dataset preparation), `gen_smarts_tree.py` (SMARTS tree generation), `validate_results.py` (validation of localisation strategies) and `values_to_groups.py` (value assignment) can be used independently.

Unless otherwise stated, all cheminformatics-related tasks were performed entirely using the RDKit[110] library.

5.2.1 Dataset preparation

All structure-data files (SDFs) used contain at least the attribute `pKa` (case-sensitive), in which exactly one floating point number is specified per molecule: It contains a pK_a value of the corresponding molecule. In the case of multiprotic molecules, the molecule may occur several times in the dataset with different pK_a values. Additionally, the attribute `temp` is present, which contains the temperature in °C at which the pK_a measurement were performed.

First, if multiple training datasets are specified, they are combined and saved into one SDF. The same procedure is performed with the validation datasets. If no separate grouping dataset is specified, all training and test datasets are also combined in the same way. The resulting dataset is used as the grouping dataset in the further course. All test datasets, as well as the combined training dataset, the combined validation dataset and the grouping dataset, are then cleaned, filtered and standardised.

Within the cleaning part, all molecules are first checked for their validity. Entries with missing pK_a values or pK_a values that cannot be converted into floating point numbers are sorted out. In addition, the atomic numbers of all molecules are canonicalized using the canonical rank order, since consistent atomic numbering is indispensable for the identification of titratable groups and the assignment of pK_a values. Subsequently, salts are removed from the individual molecules, whereby entries that still contain multiple fragments afterwards are sorted out.

In the following filtering step, molecules were filtered out if they contain more than 35 heavy atoms. In addition, molecules containing other elements than H, C, N, O, F, P, S, Cl, Br or I, and molecules containing isotopes were removed. If the molecule contains a special attribute, in this case the attribute `orig_atom`, it will not be taken into account in the two filtering steps mentioned before, and thus will not be sorted out. This mechanism serves to retain certain molecules for further investigation in the process of the pipeline. The mentioned attribute contains, if available, the locations of the titratable groups, which have already been determined by the experimental measurement. This makes it possible to check afterwards whether the locations determined by the pipeline correspond to those of the experimental measurements. Furthermore, all entries are sorted out whose pK_a values were not determined within the temperature range of 20 to 25°C (inclusive).

Then all molecule entries were standardised. To do so, the charge state and tautomeric forms were standardised using *OpenEye QUACPAC/Tautomers*[96]. A final sanitization check was performed before all identical molecules were combined based

on their canonical isomeric Simplified Molecular-Input Line-Entry System (SMILES) string. The properties of the combined entries, i.e. the pK_a values, temperatures, etc., were kept as a JavaScript Object Notation (JSON) list. This unique set of molecules was then handed over to the CXM commandline tool `cxcalc` to add predicted pK_a values at a temperature of 25°C as well as the corresponding titratable site locations to the molecules. Here, only predicted pK_a values in the range of 2 to 12 were kept. The final set of molecules was then saved as a new SDF. Before proceeding to the next step, a check of redundancy between training and test set is performed. Additionally, a check is performed to see if the validation dataset contains molecules from the grouping dataset. In both cases these molecules are then removed from the test datasets and validation dataset, respectively.

5.2.2 SMARTS tree generation

In order to arrive at structural motifs representing titration sites SMARTS patterns are most suitable. More specifically, a rooted representation would be the best access to such a representation. A SMARTS tree is such a representation and is a rooted tree known from graph theory, where the root node represents a SMARTS pattern, which is a substructure of the SMARTS patterns of all hierarchically subordinate nodes in the tree. An example of such a SMARTS tree is shown in figure 5.6.

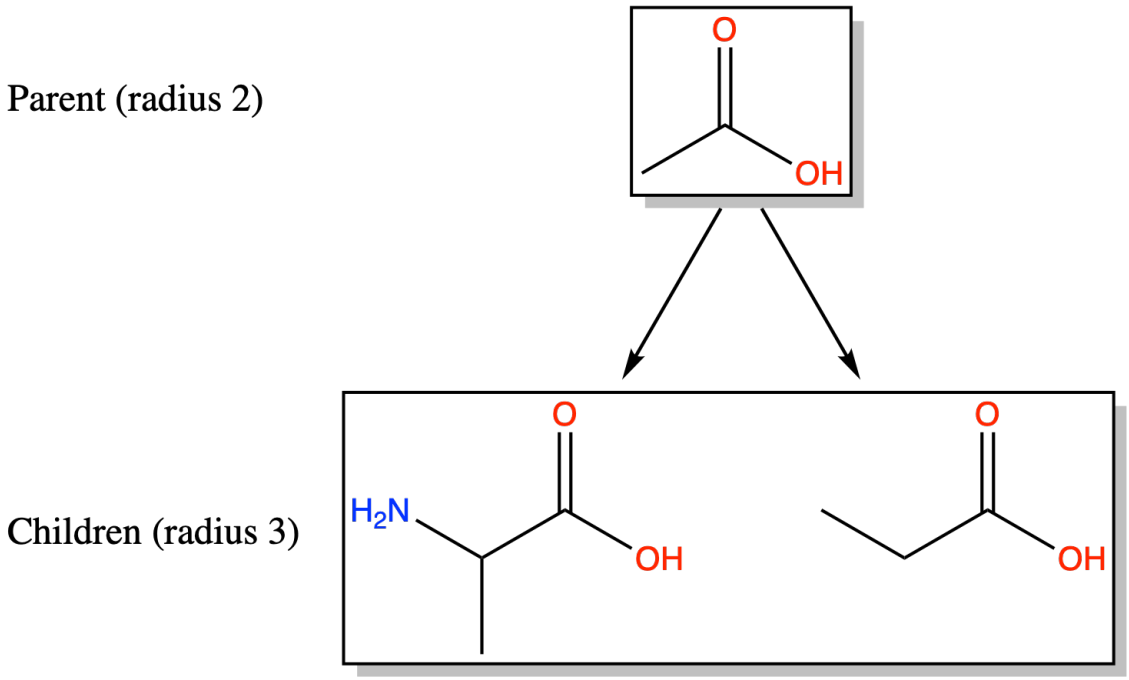


Figure 5.6: Visualized example of a SMARTS tree: Here, a radius 2 fragment containing a carboxylic group acts as the parent while the two children fragments contain the parent fragment as a substructure.

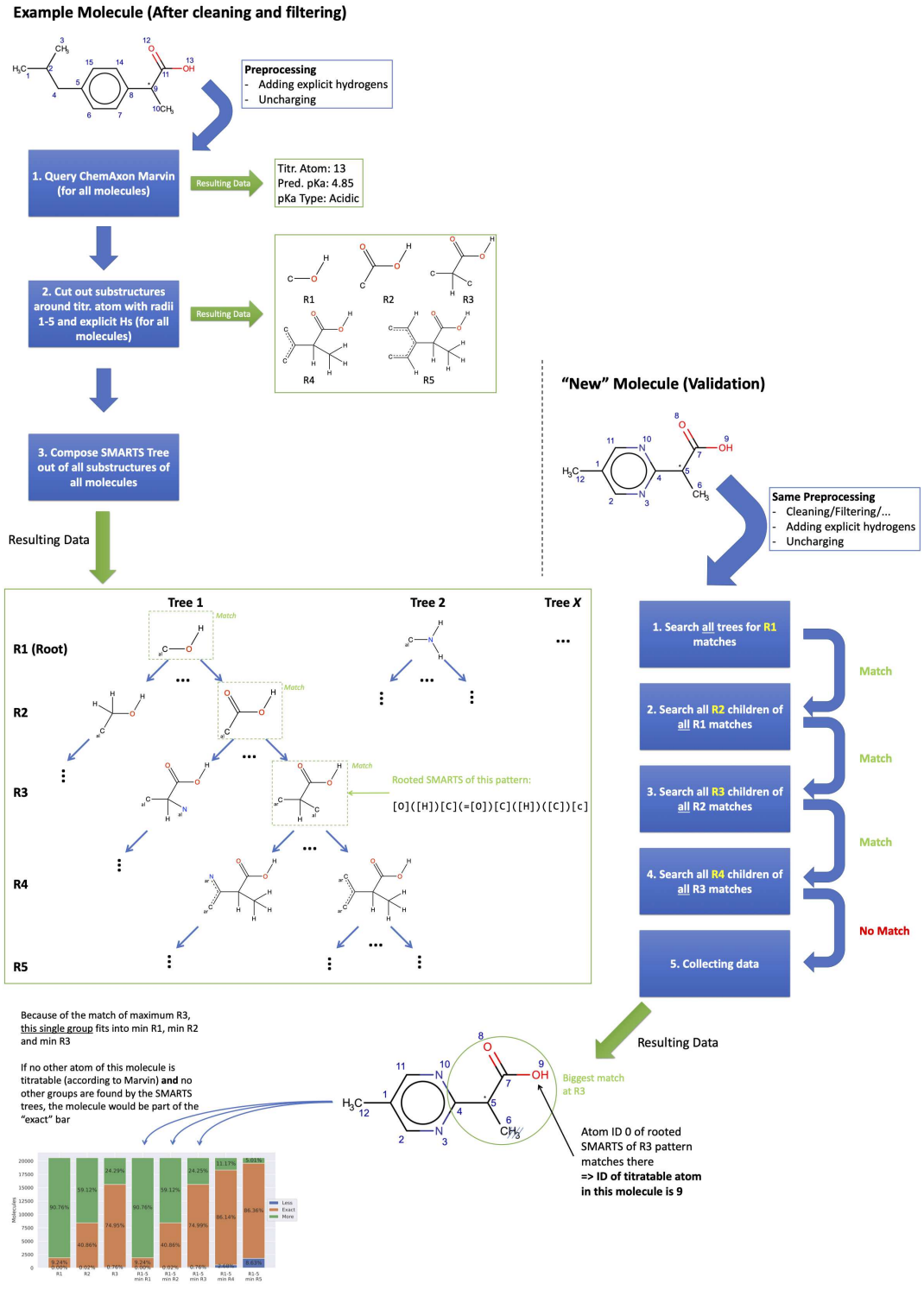


Figure 5.7: Workflow of the generation of the rooted SMARTS trees

In order to compare and verify the results of CXM an additional tool called Dimorphite-DL[111] (DDL) was used. DDL generates all possible ionisation states of all molecules in a given SMILES file for a specified pH range. To use the DDL tool, the SDF generated in dataset preparation step was first converted into a SMILES file, which was then passed to the DDL script. In addition, the pH range was restricted to 2 to 12 and the maximum number of variants to be generated was increased from 128 to 100 000, which is practically equivalent to removing the restriction. The result is also a SMILES file with all possible ionisation states. In order to generate the locations of the titratable groups from this, a formal charge matrix was created in which each column corresponds to an atom and each row to an ionisation state. The corresponding formal charge is entered in the individual cells of the matrix. All columns that do not consist exclusively of a single value mark a titratable group rooted at that corresponding atom of that column. An example of this procedure is shown in figure 5.8.

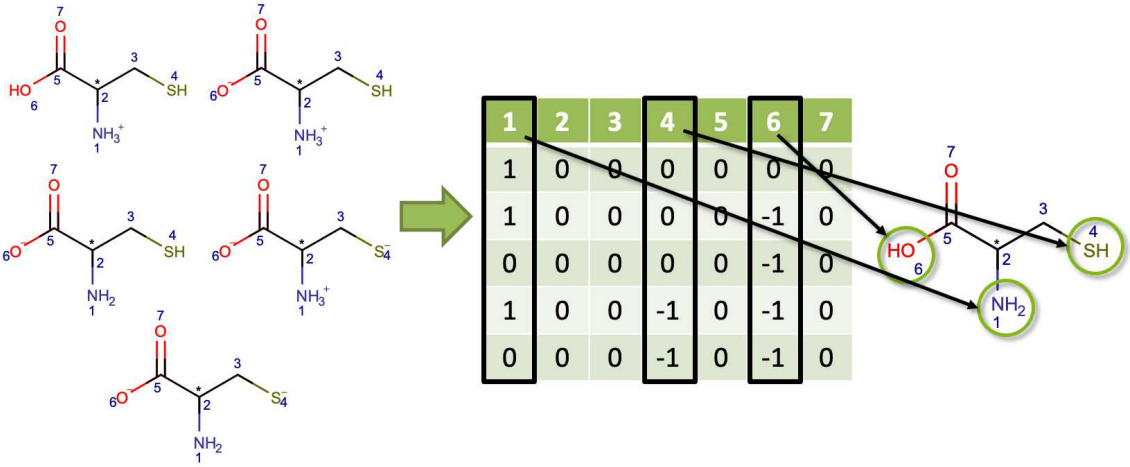


Figure 5.8: Example of the formal charge matrix procedure: Here, the molecule has three titratable groups and 5 different microstates. These microstates are converted into a formal charge matrix where each column corresponds to an atom, and each row belongs to a microstate. Every atom, that has not only zeros in its column must be a titratable group.

From the locations of the titratable groups of CXM as well as the locations resulted from the DDL run, the titratable groups can then be extracted. For this purpose, different radii from radius 0 to radius 5 (inclusive) were tested. For each radius, the titratable groups were cut out in the corresponding size and counted to get an overview of the most frequently occurring titratable groups. For generating the SMARTS trees only the CXM results were used. The reason why the results of both tools were not combined and used here is that otherwise the result would become significantly more unspecific and too many titratable groups would be found in the

dataset. This is probably due to the fact that the classification of the titratable groups in the desired pH range of 2 to 12 cannot be carried out as accurately by DDL and thus many of the additional groups resulting here are just outside of this pH range. CXM is at an advantage here due to the additional accurate prediction of the pK_a value and can therefore perform a more precise classification.

Subsequently, based on CXM, a set of the 20 most frequently occurring titratable groups with radius 1 was created. Every single titratable groups with radius 1 forms the root of a SMARTS tree. For each of these roots, a SMARTS tree was then created, which contains all subordinate SMARTS patterns of all radii found in the dataset. A detailed visualized description of the workflow for the generation of the SMARTS trees is given in figure 5.7.

5.2.3 Validation of localisation strategies

In the next step, different strategies for reliably localising the titratable groups were validated. For this purpose, two different datasets were used. First, the strategies are validated with the grouping dataset, in a second step an external validation dataset (Settimo *et al.*[14]) was used. This validation verifies the SMARTS trees derived in the previous step.

Twelve different strategies were tested:

1. SMARTS pattern with radius 1 - termed "R1"
2. SMARTS pattern with radius 2 - termed "R2"
3. SMARTS pattern with radius 3 - termed "R3"
4. Substructure hits with SMARTS pattern of radius 1, followed by subordinate SMARTS pattern of radius 2 - termed "R1 with R2 valid."
5. Substructure hits with SMARTS pattern of radius 1, followed by subordinate SMARTS pattern of radius 3 - termed "R1 with R3 valid."
6. Substructure hits with SMARTS pattern of radius 1, followed by subordinate SMARTS pattern of radius 4 - termed "R1 with R4 valid."
7. Substructure hits with SMARTS pattern of radius 1, followed by subordinate SMARTS pattern of radius 5 - termed "R1 with R5 valid."
8. Full tree search: the SMARTS tree was consecutively searched with SMARTS patterns of radii 1 to 5 until no more matches were found; the highest radius of a match is radius 1 - termed "R1-R5 min R1"

9. Full tree search: the SMARTS tree was consecutively searched with SMARTS patterns of radii 1 to 5 until no more matches were found; the highest radius of a match is radius 2 - termed "R1-R5 min R2"
10. Full tree search: the SMARTS tree was consecutively searched with SMARTS patterns of radii 1 to 5 until no more matches were found; the highest radius of a match is radius 3 - termed "R1-R5 min R3"
11. Full tree search: the SMARTS tree was consecutively searched with SMARTS patterns of radii 1 to 5 until no more matches were found; the highest radius of a match is radius 4 - termed "R1-R5 min R4"
12. Full tree search: the SMARTS tree was consecutively searched with SMARTS patterns of radii 1 to 5 until no more matches were found; the highest radius of a match is radius 5 - termed "R1-R5 min R5"

First a simple assignment using the SMARTS patterns with radius 1, radius 2 as well as radius 3 (1.-3.) was performed, followed by a check using the generated SMARTS trees. The latter works in two steps. First, substructure hits with the radius 1 patterns are searched for. Each hit found is checked in the second step with the hierarchically subordinate SMARTS patterns of one single larger radius. If a hit can also be found here, the hit is classified as valid. In the case of the second step, the radii 2 to 5 (including, 4.-7.) were checked respectively. Finally, also a full tree search was performed. For this purpose, the tree was processed level by level for each radius 1 match until no more matches could be found. The highest radius at which a match was found was saved. This value was then used as a confidence value: The higher the radius, the more certain this is a correctly located titratable group. Several confidence limits were then evaluated (8.-12.). In addition, the time required for each strategy was measured in order to be able to make a statement about the efficiency of the method.

5.2.4 Value assignment

The method that worked best in the previous step is used in this final step to assign the experimental pK_a values to the individual groups. Here, predicted pK_a values by CXM and averaged reference values were used for the assignment of pK_a values to the respective titratable groups. The outcome will be described in detail in the Results section (cf. 5.3).

Only molecules were kept for which the number of titratable groups found matched the number of groups found by CXM. In addition, all experimental values per molecule that did not fall between 2 and 12 $\pm$ 0.3 (experimental tolerance) were

sorted out. Subsequently, the remaining experimentally determined pK_a values per molecule were checked with respect to multiple measurements. Here, an experimental tolerance of ± 0.3 was also taken into account. If a molecule then had fewer experimental values (prior to and after allocation), it was also sorted out.

The experimental tolerance of 0.3 was chosen based on the underlying dataset. Here it is important that the tolerance is large enough to correctly summarize all individual experimentally determined pK_a values of a single titratable group of a molecule and to assign this value to this one group. If the tolerance is too small, two experimental values that actually describe the same titratable group could be considered as values for two different titration sites. If the tolerance is too large, the opposite happens and values that actually belong to two different titration sites are combined into one. For a better understanding, this problem is explained in the following using the example of piperazine, shown in figure 5.9. It should be noted that, due to symmetry, it is not possible to assign the individual pK_a values to the respective titration site as shown in figure 5.9. This "mapping" is done by CXM for visualisation purposes only. The example described below is a pure dataset inspection intended to illustrate how the data base is consisted within the available experimental datasets for this exemplary molecule.

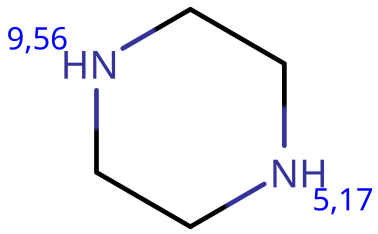


Figure 5.9: Piperazine with predicted pK_a values drawn and calculated by ChemAxon Marvin[8]

ChemAxon Marvin[8] predicts a pK_a value of 5.17 (A) and 9.56 (B) for the two titratable groups. In the dataset used, there are 21 experimental measurements in 5 different source (ChEMBL29, DataWarrior, Literature Compilation, OpenEye and Yu *et al.*, compare table 5.1) for this molecule. Ten of the values can apparently be assigned to the titratable group A, eleven values to group B. The values are as follows:

(A) 5.333, 5.333, 5.424, 5.55, 5.56, 5.6, 5.6, 5.63, 5.68, 5.81

(B) 9.72, 9.731, 9.731, 9.8, 9.8, 9.81, 9.82, 9.83, 9.864, 9.89, 9.93

For group A, the standard deviation of the experimental values is 0.151 with a total

spread of 0.477. For group B, the standard deviation is 0.067 with a total spread of 0.210. With an experimental tolerance of 0.3, the two groups are separated without problems in this example, however, a tolerance of 0.13 would also be sufficient here, since the maximum difference between two successive values of a group is only 0.13. In the fictitious situation that only the values 5.333 and 5.81 were contained in the underlying dataset, the selected tolerance would not be sufficient and the values would be taken for two different titratable groups. An additional example of different experimental readings regarding piperazine can be seen in the table in figure 5.10, which is taken from the publicly available iBonD database[112].

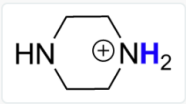
Structure	Solvent	pK_a	Method
	H_2O	9.72	Calor.
	H_2O	9.71	PTM
	H_2O	10.03	SM
	H_2O	5.6	Calor.
	H_2O	5.41	PTM
	H_2O	5.66	SM

Figure 5.10: Experimental pK_a values for piperazine contained in the iBonD database[112]

After this preprocessing, the actual assignment of the values based on the predictions by CXM as reference values took place. Three different methods were investigated: (1) a simple assignment by iteration over the experimental values and titratable groups together with a check of the respective experimental error, (2) an assignment via an error matrix with experimental and predicted values on the respective axes, and (3) the generation of all possible assignments by permutation from experimental value to the corresponding titratable group, choosing the permutation with the lowest total error. The corresponding algorithms of each method can be viewed in the appendix (see chapter A.1). Method (1) strongly depends on the ordering of the experimental values as well as the predicted values at the time of assignment. This leads to inconsistent results which is why this method was discarded. The quality of methods (2) and (3) were evaluated by calculating the mean absolute error (MAE), root-mean-square error (RMSE) and coefficient of determination (R^2) metrics for the resulting assignments. The method which gave the best results in terms of these metrics was chosen for the final assignment. Finally, only molecules with less than

six titratable groups and a maximum absolute error of 2.0 each were kept. The resulting dataset was saved as an SDF.

5.3 Results

All datasets from table 5.1 (except Settimo *et al.*[14]) were specified as combined training and grouping dataset for the pipeline run presented here. The Settimo *et al.*[14] dataset was used as validation dataset. This chapter and its subchapters are intended to present the results of MPP based on the datasets from table 5.1, without already addressing the use of MPP as preprocessing for machine learning, as described in chapter 6. Therefore, for this pipeline run, no test dataset according to the terminology described in chapter 5.2 was specified for analysis.

5.3.1 Dataset preparation

During training/grouping dataset preparation, 4 669 entries were sorted out due to containing more than 35 heavy atoms, prohibited elements or isotopes. 19 431 entries were sorted out due to temperature rules, and 15 other entries could not be tautomer-standardised. The remaining entries were then combined, leaving only unique molecules which resulted in 22 671 molecules. CXM was then applied for the prediction of the pK_a values and the localisation of titratable groups. The associated filtering by pK_a range from 2 to 12 finally resulted in 20 604 molecules.

The filtering steps removed 120 molecules in the validation dataset (Settimo *et al.*[14]). Additionally, 361 molecules were removed since they were already included in the grouping dataset. This left 131 molecules for validation.

5.3.2 SMARTS trees

For the creation of the SMARTS trees, a detailed statistical evaluation of the distribution and frequency of the individual titratable groups was carried out by using the tools CXM and DDL.

ChemAxon Marvin

The results for CXM (radius 0, i.e. only the elements) are shown in figure 5.11. Here, it can be seen that CXM mostly finds titratable nitrogen, oxygen and sulfur atoms while very few titratable carbon and phosphor atoms could be detected. Figure 5.12 shows the top 10 of the titratable groups identified by CXM given a radius of 1. The results for radii 2 to 5 can be found in the supplementary content (A.3).

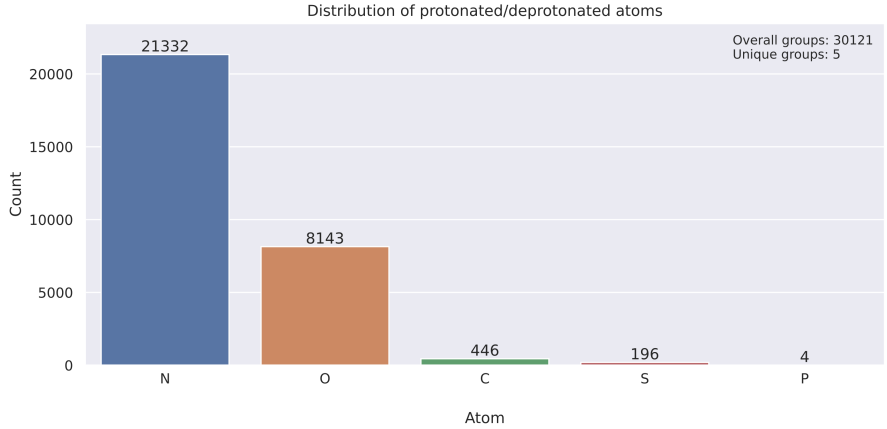


Figure 5.11: Distribution of titratable elements calculated with ChemAxon Marvin[8]

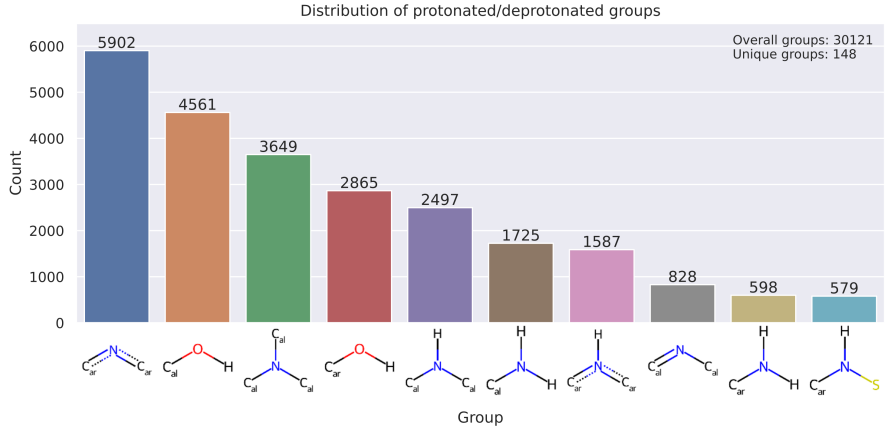


Figure 5.12: Distribution of the top 10 titratable groups with radius 1 calculated with ChemAxon Marvin[8]

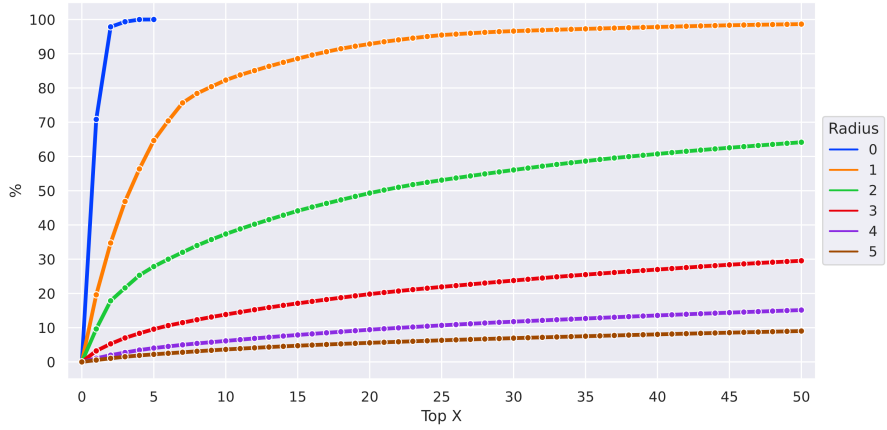


Figure 5.13: Saturation curve of titratable groups calculated with ChemAxon Marvin[8]

At this point the question arose how many titratable groups (sorted by frequency) are necessary to cover the majority of the titratable groups of the datasets. This investigation was carried out for all investigated radii from 0 to 5 as well as for both respective tools (CXM and DDL). The CXM results are shown in figure 5.13. It can be seen that the top 20 titratable groups with radius 1 are sufficient to cover nearly 95% of all titratable groups in the entire dataset. Therefore, the top 20 radius 1 SMARTS patterns were extracted for further analysis. These SMARTS patterns are shown in figure 5.14 sorted in descending order by the number of occurrences.

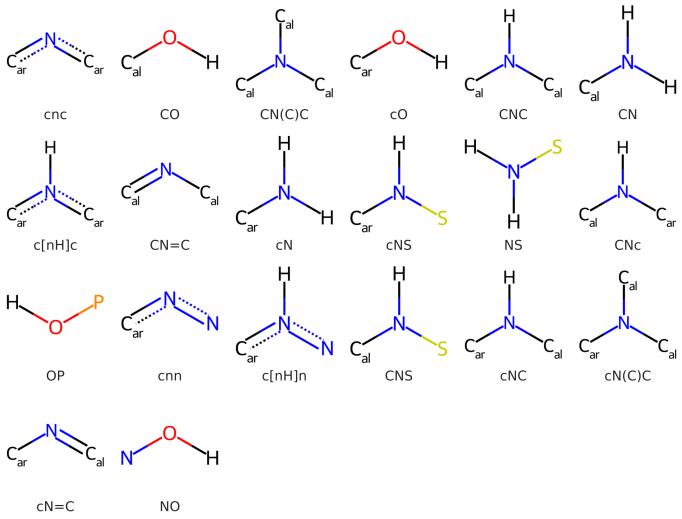


Figure 5.14: Combined and unified top 20 titratable groups of ChemAxon Marvin[8], sorted in descending order by number of occurrences

Dimorphite-DL

The results generated in the process of creating the SMARTS trees for DDL for radius 0 (i.e. only the elements) are shown in figure 5.15. Compared to the results of CXM, DDL does not find titratable carbons and phosphors in addition to the elements nitrogen, oxygen and sulphur. After investigating the underlying data used in DDL, it can be seen that there are no cases with carbon and or phosphorus included. Thus, DDL technically cannot find these groups. It is also noticeable that DDL finds a significantly higher total number of titratable groups within the dataset used than CXM. Here it can be assumed that the limitation to the pH range from 2 to 12 cannot be made accurately with DDL due to the missing or not intended ability to predict pK_a values and thus groups are identified which could only be titrated outside the pH range mentioned. Figure 5.16 shows the top 10 found titratable groups with radius 1 for DDL. Apart from the different absolute numbers of the found groups of DDL and CXM, it is noticeable that the top 10 of both tools are

almost identical, only the order differs in some places. This finding also applies to radii 2 to 5, the corresponding figures can be found in the supplementary content (A.4).

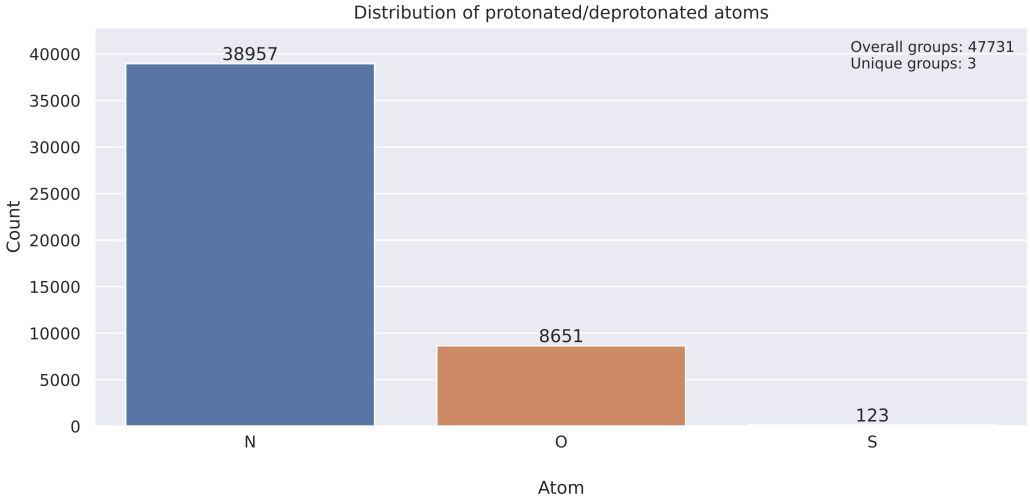


Figure 5.15: Distribution of titratable elements calculated with DDL

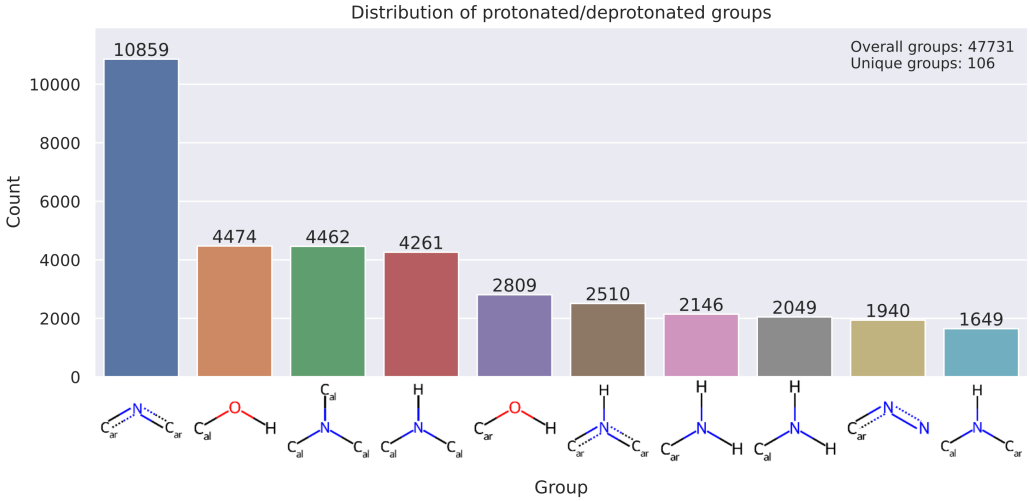


Figure 5.16: Distribution of titratable groups with radius 1 calculated with DDL

The results of the investigation of the amount of titratable groups that are needed to cover almost the entire dataset are shown in figure 5.17. Compared to the CXM results the figures look identical at first glance, but it can be seen that fewer groups are needed for the titratable groups from CXM to obtain a higher coverage than DDL. This difference becomes more significant as the radius increases. With DDL the top 20 titratable groups with radius 1 are also sufficient to cover nearly 95% of all titratable groups in the entire dataset. A visualisation of these SMARTS patterns is shown in figure 5.18 sorted in descending order by the number of occurrences.

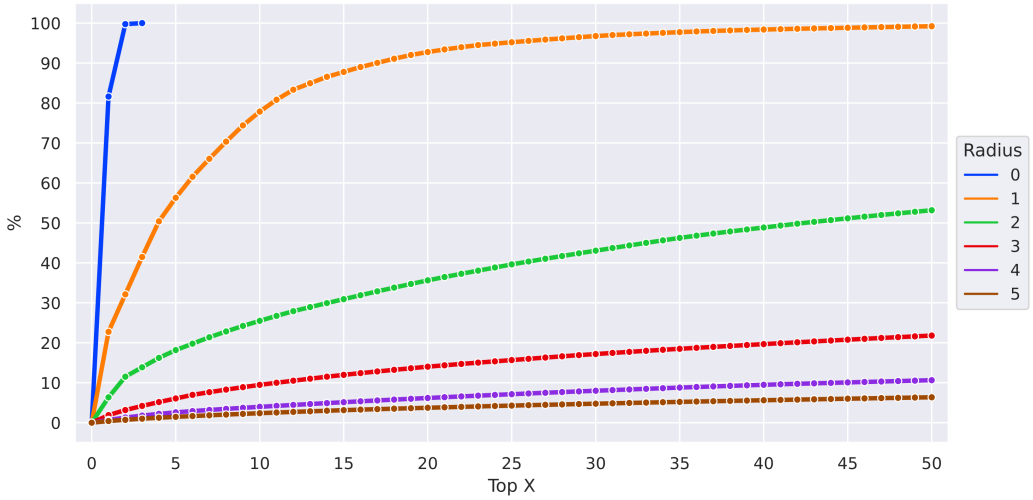


Figure 5.17: Saturation curve of titratable groups calculated with DDL

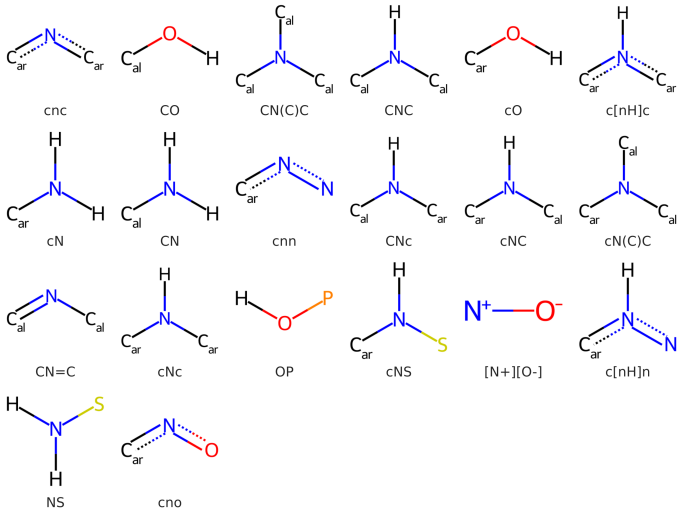


Figure 5.18: Combined and unified top 20 titratable groups of DDL, sorted in descending order by number of occurrences

5.3.3 Validation

In the next step, the different validation strategies were evaluated. The results of the eight most important strategies using the grouping dataset are shown in figure 5.19. An extended version of this figure including all twelve tested strategies as well as the corresponding run-times can be investigated in the supplementary information (A.2). The performance is given by the accuracy of the different validation strategies to correctly assign a titratable group. For example, the use of the radius 1 SMARTS pattern resulted in 9.24% correct assignments. For 90.76% of the cases, too many groups were found compared to the titratable groups predicted by CXM. Radius

3 yields a much higher accuracy but at the cost of drastically increased runtime. This was one motivating aspect for us to develop the SMARTS trees in order to reduce the computational burden. As it is shown in the figure, the results using the SMARTS tree approach with minimal radii of 1 to 3 are very close to the results of using the plain radius 1 to 3 pattern but it is much faster. In figure 5.20 the results of the evaluation using the validation dataset are shown. Here, it can be seen that the methods work less reliably for datasets that are not included in the grouping dataset. The methods practically "overtrain", which means the SMARTS pattern are too specific to detect titratable groups in unseen molecules, but still find only the correct groups given a radius of 3 for 60.31% of the molecules. For the localization of titratable groups in external datasets it is therefore better to choose a lower minimum radius. In contrast, it is better for the preparation of the entire dataset in relation to a subsequent machine learning to apply a higher minimum radius of 4 or even 5.

In the further process the tree approach with minimal radius 4 was chosen to make the actual assignment of the experimentally determined pK_a values to the corresponding titratable groups.

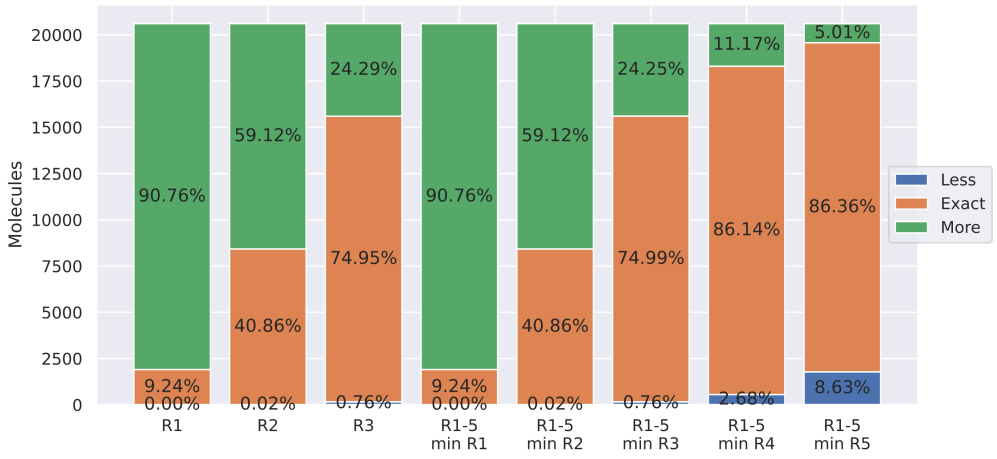


Figure 5.19: Visualization of the evaluation of the different location strategies for the grouping dataset.

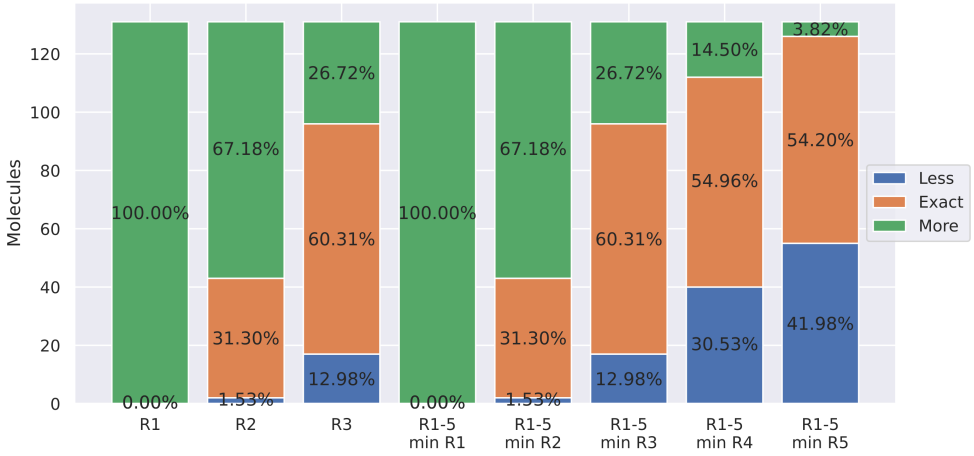


Figure 5.20: Visualization of the evaluation of the different location strategies for the validation dataset

5.3.4 Value assignment

In the final step, the experimental values were assigned to their corresponding titratable groups. In order to come up with a fully annotated dataset, in which each identified titratable group is mapped to the corresponding experimental value, the molecules with missing experimental values were sorted out. In addition, the molecules for which the number of titratable groups found did not correspond to the number of groups determined by CXM were removed, since a clear statement about the validity of the assignment could not be made here. Given this criterion, 13 588 molecules with a total of 16 439 titratable groups remained, which corresponds to about 66% of the molecules in this step. Of the 13 588 molecules, 2 518 molecules had multiple titration sites. As described in chapter 5.2.4, methods (2) and (3) were evaluated with respect to their quality. The assignment using method (2) resulted in a MAE of 0.866, a RMSE of 1.413 and a R^2 of 0.708. This compares to an MAE of 0.859, an RMSE of 1.375 and an R^2 of 0.722 using method (3). Therefore, method (3) was used for the following assignment.

Figure 5.21 shows the result of this assignment for each titratable group, with the respective experimental values plotted on the X axis and the predicted values of CXM plotted on the Y axis. Here, it can be seen that there are apparently still many outliers where the assignment did not work correctly: either the experimental value is not correct or the CXM prediction is inaccurate. To gain insight into the most prominent outliers, two monoprotic molecules were analysed as examples. In addition, the distribution of the titratable groups by element within the monoprotic outliers was analysed and is depicted in figure 5.22. The distribution of the elements

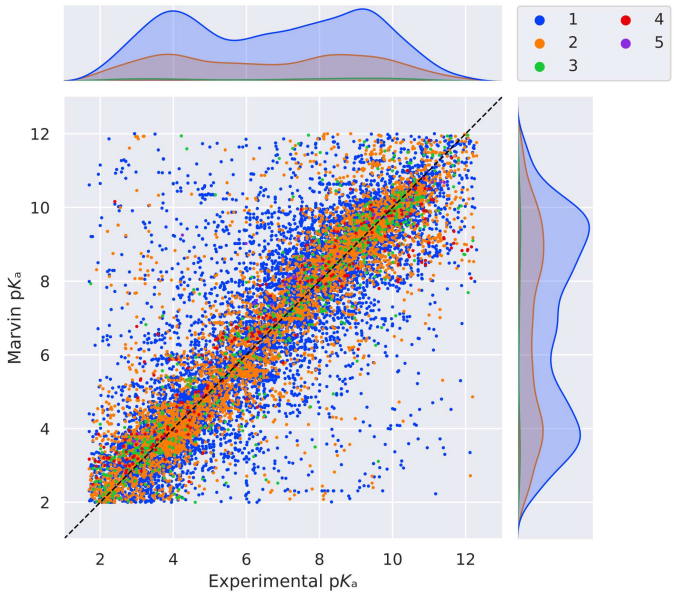


Figure 5.21: Result of value assignment for each titratable group without error cut: The respective experimental values plotted on the X axis and the predicted values of CXM plotted on the Y axis. The colors indicate if the pK_a value belongs to a monoprotic molecule (1), a diprotic molecule (2), and so on. Additionally, the distribution of the pK_a values for each axes are shown in the ridge plots on the top and on the right of the scatter plot. Statistical metrics: MAE=0.859, RMSE=1.375, R^2=0.722

of the titratable groups of the monoprotic outliers with an absolute error of at least 4.0 corresponds closely to the distribution within the monoprotic molecules of the filtered dataset shown in figure 5.24, with the exception that the outliers contain significantly more titratable carbon atoms. This makes it unlikely that there is a systematic problem with erroneously reported pK_b values instead of the pK_a values in the source datasets. In such a case it would be expected that the titratable nitrogens in particular would be overrepresented within the monoprotic outliers. The exemplary analysis of the two selected outlier molecules also reveals other problematic issues.

Figure 5.23 shows two examples of strong outlier molecules and their CXM predicted pK_a values. Molecule (A) has two titration sites with predicted pK_a values of -0.11 and 11.86 according to CXM. For the molecule, three of the available datasets each provide an experimental pK_a value, which is 2.24 in all cases. The datasets are DataWarrior, Hunt *et al.*[105] and Yu *et al.*[109] Due to the restriction of the pK_a range to 2-12 within MPP, the predicted value of -0.11 was not considered, although in this case assigning the experimental value of 2.24 to this particular titration site would have resulted in a significantly lower absolute error of 2.35 instead of 9.62 as in the present case. However, this molecule would still have been rejected as an

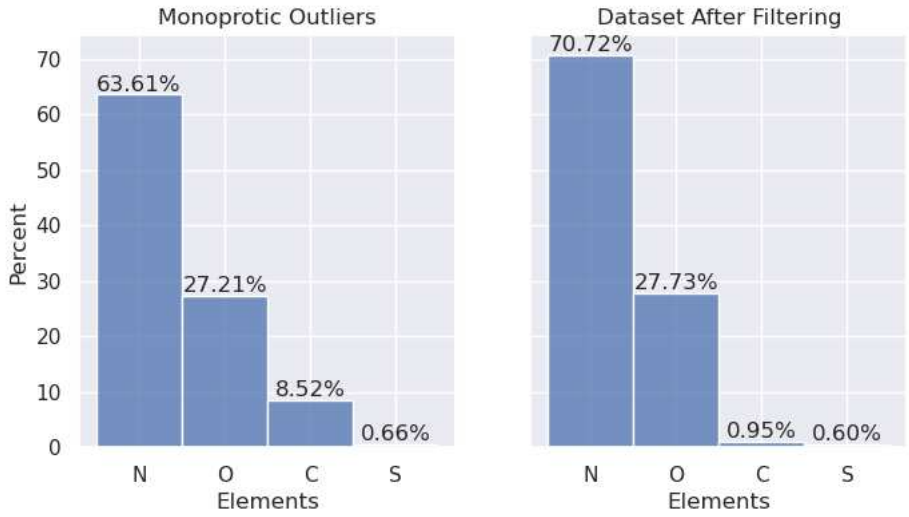


Figure 5.22: Distribution of the elements of the strong outliers' titration sites after value assignment for all monoprotic molecules: In this case, a strong outlier is defined by an absolute error between the experimental pK_a and the CXM prediction of at least 4.0

outlier as the error is still above the threshold of 2.0. The case of molecule (B) is similar. Here two titration sites with predicted pK_a values of 1.84 and 11.25 are given by CXM. The actual experimental pK_a value taken from ChEMBL29[99] is 2.7. Again, the titration site with a pK_a of 1.84 was filtered out by limiting the pK_a range to 2-12 and the experimental value of 2.7 was assigned to the predicted value of 11.25, resulting in an error of 8.55. In this case an assignment to the titration site with a pK_a of 1.84 would have resulted in a significantly lower error of 0.86, which means that the molecule would not have been classified as an outlier. From these examples it can be seen that the strict restriction to the pK_a range of 2-12 could be a cause of the extreme outliers. An extension of the pK_a range considered, at least for the predicted molecules of CXM, to 0-14 would probably have led to a reduction in the number of extreme outliers. The 15 most extreme outliers, including the experimental and predicted pK_a values, are further presented in the supplement in chapter A.5, figure A.11. A complete compilation of all molecules excluded due to an error greater than 2.0 can be found in the electronic appendix (cf. chapter A.6.1).

However, since an individual check of every single experimental value cannot be performed, no generally applicable reasoning can be given. Nevertheless, the resulting dataset should be as accurate and trustworthy as possible in order to be able to use the data for further tasks, such as machine learning. Therefore, all molecules for which the absolute error between the experimentally determined pK_a value and the CXM prediction is greater than 2.0 for at least one titratable group were removed. Given this criterion, 11 983 molecules with 14 354 titratable groups remain,

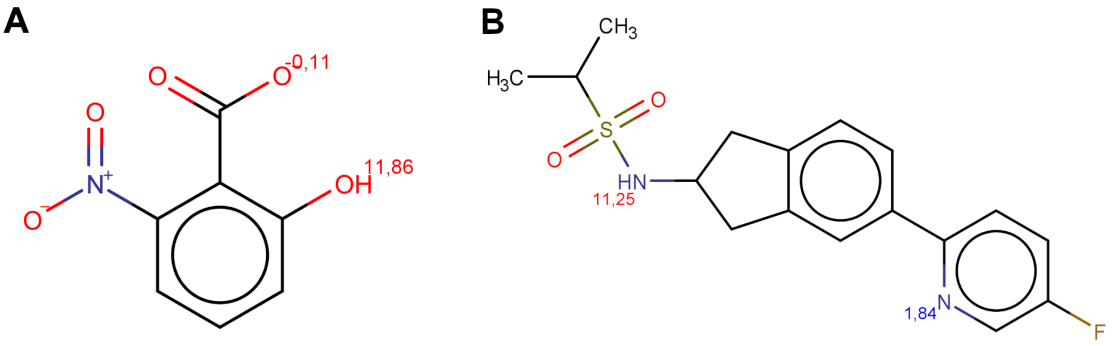


Figure 5.23: Exemplary outlier molecules: Both molecules (A) and (B) are considered monoprotic within the pK_a range of 2-12. In both cases the titration sites outside this range would be closer to the available experimental pK_a values 2.24 (A) and 2.70 (B).

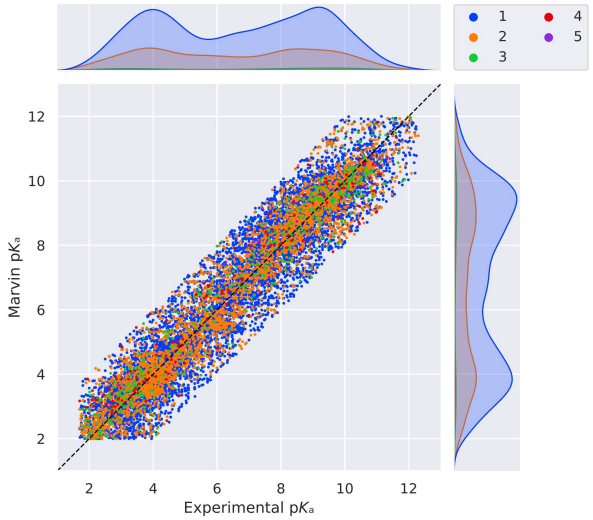


Figure 5.24: Result of value assignment for each titratable group with cut at an absolute error of 2.0: The respective experimental values plotted on the X axis and the predicted values of CXM plotted on the Y axis. The colors indicate if the pK_a value belongs to a monoprotic molecule (1), a diprotic molecule (2), and so on. Additionally, the distribution of the pK_a values for each axes are shown in the ridge plots on the top and on the right of the scatter plot. Statistical metrics: MAE=0.567, RMSE=0.742, R^2=0.919

in which 2 121 multiprotic molecules are included. This now leads to a MAE of 0.567, a RMSE of 0.742 and a R^2 of 0.919. In figure 5.24, analogous to figure 5.21, the distribution of the filtered dataset is given.

In addition to the assignment of the experimentally determined pK_a values to the corresponding titratable groups, it was also investigated to what extent a pK_a value can be approximately inferred from the group itself. For this purpose, the final dataset resulting from the pipeline was taken to analyse how the pK_a values are

distributed when only the titratable groups with a radius of 1 are considered. The results for the 20 most frequent titratable groups are shown in figure 5.25. Here, it can be clearly seen that the pK_a values per group can vary widely, partly within the mean 50% of the data of a group over five pK_a units. The entire range of pK_a values per group from their minimum to their maximum is in most cases spread over the complete pH range from 2 to 12. A valid statement about the actual pK_a value of one of these groups within any molecule can therefore not be made on the basis of the distribution.

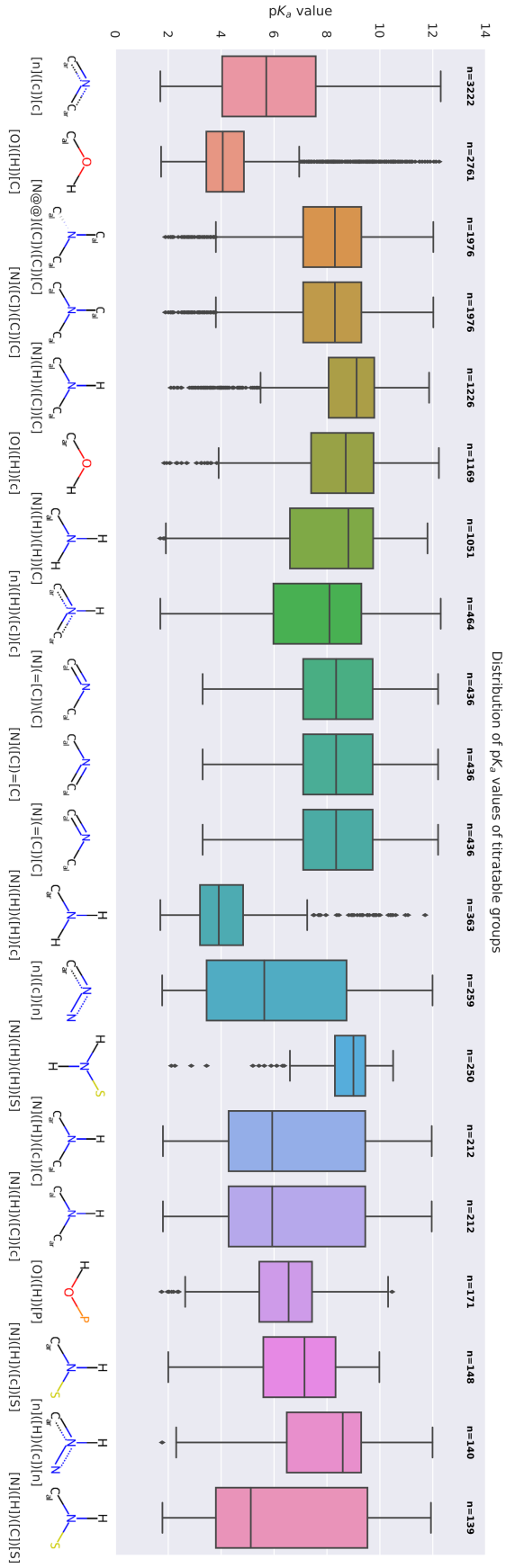


Figure 5.25: Distribution of the pK_a values of the 20 most frequent titratable groups

5.4 Discussion

5.4.1 Data basis and filtering

During the compilation of the different datasets from various sources, it was noticed that in several cases individual information about the experimental parameters of the measurement were not provided. These parameters included temperature, solvent and measurement method. In order not to have to reduce the dataset further 25°C and water were specified for temperature and solvent, respectively. This assumption was verified by evaluating random samples. Nevertheless, it cannot be ruled out that individual measured values were determined under different conditions and could therefore distort the combined measured value of the individual titratable groups after assignment in each case. The information on the measurement method was most frequently missing, which is why no differentiation or filtering was carried out here. Since the measurement method can also have an influence on the experimental value (cf. chapter 2.1.4 and figure 5.10), distortions or noise cannot be ruled out here either.

When filtering the underlying data sources, molecules with more than 35 heavy atoms as well as structures containing an unlisted element (cf. chapter 5.2.1) or an isotope are sorted out. In particular, the first two criteria serve to limit the molecules to structure classes similar to drug-like compounds. This is to map the dataset resulting from the pipeline, which will be used for machine learning in the further course, specifically to the targeted chemical space of drug-like molecules. However, this makes the result of the pipeline or machine learning no longer reliably applicable to molecules that do not meet the aforementioned criteria. In the same context, the limitation of the pK_a range from 2 to 12 must therefore also be addressed. In the human body, a pH value between 1 and 8 applies, depending on the range, whereby the lower end of the scale is only reached within the stomach[113]. Aiming at the use of the resulting data for models within drug discovery, the human pH range is therefore almost covered. The narrowing of the pK_a range from 2 to 12 was done because the laboratory instrument used to verify and generate pK_a readings within the research group has an effective measurement range of 2 to 12. This rough limitation exists for several measurement methodologies independent of the particular instrument used[26].

5.4.2 SMARTS tree and localization strategies

As described, the use of SMARTS trees significantly accelerates the localization and verification of titratable groups. At the same time, the depth within the tree, and

thus the radius in the tree level at which a match is still found for such a particular group, can be used to specify a confidence level. The use of SMARTS patterns for the identification of titratable groups in general, and thus also in the present case with the use of SMARTS trees, carries the risk of overfitting in a metaphorical sense. A SMARTS pattern list or a SMARTS tree, which is based on the data used for generation, can only recognize and localize titratable groups seen in the dataset. For example, if the SMARTS patterns are based on a dataset containing only carboxylic acids, no titratable amines can be found. In context of this work, the use of a relatively large dataset from different sources and chemical domains ensures a relatively broad coverage, but it cannot be excluded that individual titratable groups of an unknown molecule cannot be found. Due to the lack of experimental information on the membership of a pK_a value to a specific group, the prediction and localization capability of ChemAxon Marvin[8] was used, as described. Since the SMARTS pattern generation based on this does not insist on the presence of experimental measurements, the problem can be circumvented by using a large collection of molecules to generate the SMARTS tree. For example, the entire ChEMBL[99] database would be a suitable molecule collection if the described filtering is respected. Along with this, another issue specific to the use of trees as the organizational structure of SMARTS patterns arises. The generation of the tree requires a high computational effort, which is why the use of large databases results in a considerably long runtime within the generation process. By an improvement of the implementation and extended parallelization on computing clusters this problem can be minimized in principle. By the hierarchically forced structure of the tree from above downward however no unrestricted parallelization is possible.

Twelve different localization strategies were tested (cf. chapter 5.2.3), which are divided into three categories as described. In the first category (1.-3.) the extracted SMARTS patterns are tested in the respective size and in case of a hit the titratable group is labeled. This methodology is only suitable for smaller datasets and small radii. Already from a radius of 3, the computational effort increases overproportionally, causing a localization run for a more than 20 000 molecules to take almost 30 minutes. At higher radii of 4 or 5, localization is no longer feasible in an acceptable time if a similarly large and diverse dataset serves as the basis for the SMARTS patterns. Here, a significant gain in speed could be achieved with the second category (4.-7.). In each case, an preceding search was performed with the few radius 1 patterns and then the hits with a higher radius were validated in each case. Only patterns containing the original radius 1 pattern, i.e. its children, were used for validation (cf. figure 5.6). The disadvantage of this method is that one has to decide on a validation radius in advance. This is where the SMARTS tree

method (8.-12.) shows its strengths. Due to the one-time generation of the tree and the subsequent recursive validation from top to bottom within the tree, the time required for all different validation radii is constant. However, the already described computationally expensive generation of the tree arises here.

5.4.3 Value assignment and further analysis

During the value assignment process, a significant portion of approximately 33% of the dataset available up to that point is filtered out. This is due to the chosen restriction that the assignment of pK_a values to titratable groups of the respective molecule must be unambiguous. This means that molecules which, on the basis of the experimental measured values, should have more titratable groups than the selected localization strategy has determined or too few experimental values are available for the specific molecule, were sorted out. The motivation for this procedure is considered to be the machine learning that follows the pipeline run. Here, an unambiguous and trustworthy dataset was required. Provided that both the localization by the pipeline and the localization by ChemAxon Marvin[8] match, the extent of the filtered-out molecules could have been reduced by making the best possible assignment if too many experimental measured values were available and discarding the additional values. Filtering with too few available values measured by the number of titratable groups could not be avoided based on the targeted machine learning strategy. In other machine learning methods, for example node-based prediction using Graph Neural Networks (GNNs), incompletely labeled molecules could have been used, but this was not pursued further in the course of this work. In the same context, the hard cut at an error of $\pm$ 2 pK_a units must also be discussed. The error value used here was the deviation of the assigned experimental value from the value predicted by ChemAxon Marvin[8] for the same titratable group. Even if ChemAxon Marvin[8] itself only makes a prediction, which itself may contain an error, a deviation of more than two pK_a units is at least questionable. This means that the trustworthiness of the assignment and/or the measured value is no longer given. However, there is of course the possibility that the experimental value filtered through this limit was in fact correct and ChemAxon Marvin[8] predicted a value that was strongly in deviation. This can only be verified either by experimental verification of the individual values or by comparison with several pK_a prediction programs. Unfortunately, experimental verification was not possible within the scope of this work, nor was comparison with other pK_a prediction programs possible for licensing reasons. Without further verification, it is not possible to determine which of the values is incorrect.

Finally, the boxplot in figure 5.25 needs to be discussed in more detail. This plot

is based on the values from the MPP pipeline assigned to the individual titratable groups. Therefore, the assumption made about the large range of experimental measured values per unique titratable group can only be made if it is assumed that the assignment was correct and the experimental measured values themselves are correct. Since for most of the titratable groups shown the range between lower and upper whisker covers almost the entire pK_a range of 2-12, the values are relatively more widely scattered, which would suggest a systematic assignment error. However, this could be rejected as unlikely by random checks of individual measurements within the range mentioned. An alternative would be to use pK_a values predicted by ChemAxon Marvin[8] instead of the assignments determined by the pipeline for this particular analysis. However, this would include an error induced by the prediction program, which does not simplify the interpretation of the results, since a systematic prediction error cannot be excluded here either.

5.5 Conclusion

With the Multiprotic pK_a Processor, a comprehensive tool was designed to handle the process of summarizing, preprocessing, analyzing and filtering pK_a datasets. In addition, a methodology was developed to locate titratable groups and assign pK_a values to their associated titratable groups in a comprehensible manner. With the SMARTS trees, a procedure was developed with which titratable groups can be localized efficiently and in an organized manner using hierarchically arranged SMARTS patterns. For the initial construction based on the available data, an external pK_a prediction tool, which in addition to the pK_a values also determines the titration site within the molecule, is required. However, the finally constructed SMARTS tree can then be used without external tools to localize titratable groups in arbitrary molecules. In order to get a trustworthy localization it is nevertheless necessary to stay within the chemical space of the overall dataset used for construction of the SMARTS tree and to follow the filtering rules. In any case, an estimation about the trustworthiness can be made by the confidence value, which is determined by the hit with the largest radius within the tree, and thus the depth of the corresponding SMARTS pattern in the tree. An internal as well as external validation of the assignment results was performed. The assignment of the pK_a values to the titratable groups was tested with several algorithms, and the method with the best assignment quality was used in the further process. Based on this, an analysis of the pK_a value distribution per titratable group was performed and examined.

The quality of the results obtained from the MPP pipeline can be further optimized by steps such as the extension and/or completion of the basic dataset as well as

the improvement of the assignment algorithms. The reliability of the assignment of pK_{a} values to their associated groups can be further improved by building the SMARTS tree separately using a large and diverse dataset and using an external prediction tool such as ChemAxon Marvin[8]. For this purpose, optimization of the algorithm for building the SMARTS tree in terms of runtime is recommended. A broader applicability of the pipeline can be achieved by removing the restrictions in terms of filtering, keeping in mind any side effects in the area of localization and assignment of titratable groups.

Chapter 6

Multiprotic pK_a Prediction

After intensive preprocessing and validation of the datasets and preparation through the procedures described in chapter 5 for using the data through machine learning for multiprotic pK_a value prediction, the machine learning itself now had to be prepared and optimized. After extensive research and preliminary work within the research group, it was decided to use a graph-based neural network for developing a multiprotic pK_a prediction tool.

In the following chapters, a machine learning based pK_a predictor called Graph-based Multiprotic pK_a Predictor (GMPP) will be introduced. GMPP uses Graph Convolutional Neural Networks (GCNs) to provide accurate pK_a predictions within the range of two to twelve. Additionally, not only the most basic and/or the most acidic pK_a value can be predicted, instead up to three titration sites per small molecule can be predicted and localized. The entire tool is written in Python 3.9[74] and uses RDKit[114], Pandas[115], Scikit-Learn[79], Optuna[12], MLflow[13], PyTorch[10] and PyTorch Geometric[11].

6.1 Datasets

For the multiprotic pK_a prediction model, a dataset compilation consisting of 16 individual datasets was used. These include data from public databases, such as ChEMBL[99] and DataWarrior[90], data from the Statistical Assessment of the Modeling of Proteins and Ligands (SAMPL) challenges[16, 17, 100], data from experimental measurements in our own laboratory, data that could be extracted from various publications as well as data kindly provided by Idorsia Pharmaceuticals Ltd.[101], Roche Pharma AG[102], Novartis Pharma AG[15] and OpenEye Scientific Software[103]. The initial dataset compilation used here is the same as

described in chapter 5.1 and table 5.1.

6.2 Preprocessing

6.2.1 Using MPP

The datasets were preprocessed with the Multiprotic pK_a Processor (MPP) tool and therefore using the methodologies described in chapter 5. The Settimo *et al.*[14] and Novartis[15] datasets were used as test datasets. All other datasets together formed the training dataset. Deviating from the standard settings of MPP, *OpenEye QUACPAC/Tautomers*[96] was used for tautomer standardisation and to standardise the charge states of the molecules to pH 7.4. In addition, the minimum confidence for assigning the experimentally determined pK_a values to the corresponding titration sites was set to 4 as well as multiprocessing on 12 cores was activated. This resulted in a training dataset of 11 663 molecules and the Novartis[15] and Settimo *et al.*[14] test datasets with 311 and 48 molecules, respectively.

6.2.2 PyTorch Geometric

The next step was to prepare the training and test datasets for use with GCNs. This includes the unified preparation of the molecular structures, the transformation into graphs as well as the calculation of edge and node features. Due to the limited number of 41 molecules containing 4 titration sites and five molecules with five titratable groups available, it was decided to limit the model to molecules with a maximum of three titration sites. This reduces the initial number of training molecules to 11 617.

#Titration Sites	**Training Dataset #Molecules**	**Novartis[15] #Molecules**	**Settimo *et al.*[14] #Molecules**
1	9 648	202	41
2	1 815	99	7
3	142	10	-
Summary	11 617	311	48

Table 6.1: Dataset composition used for training and testing the GCNs.

First, the datasets were checked for elements that are not contained in the list of allowed atoms H, C, N, O, F, Cl, Br, I, P and S. Molecules that contained excluded elements were removed. Next, the molecules were protonated to pH 0 and all hydrogens were explicitly added. Then their 3D conformation was calculated.

This was done using the Experimental-Torsion Basic Knowledge Distance Geometry (ETKDG)[116] algorithm built into RDKit[114]. This algorithm is an extension of the distance geometry[117, 118] method that uses information such as ideal bond lengths and angles as well as torsion angle preferences from the Cambridge Structural Database (CSD) to correct the conformers generated by the plain distance geometry[116]. As the distance geometry algorithm generates a random distance matrix, a random seed of 666 was set for reproducibility. The resulting conformers were then optimised using the Merck Molecular Force Field 94 (MMFF94)[119] force field. All molecules with errors in any of the above steps were removed.

Node Features	**Edge Features**
Hybridization Type (8 bit) *Other, S, SP, SP2, SP3, SP3D, SP3D2, Unspecified*	Bond Type (4 bit) *Single, Double, Triple, Aromatic*
Element (10 bit) *H, C, N, O, F, Cl, Br, I, P, S*	Is Conjugated (1 bit)
Is Aromatic (1 bit)	Is Rotatable (1 bit)
#Hydrogens (5 bit) *0, 1, 2, 3, 4*	Bond Length (1 float)
Total Valence (7 bit) *1, 2, 3, 4, 5, 6, 7*	
Is In Ring (1 bit)	
Formal Charge (3 bit) *-1, 0, 1*	
Is H Donor (1 bit)	
Is H Acceptor (1 bit)	
Is Titration Site (3 bit) *TS1, TS2, TS3*	
Total: 40	Total: 7

Table 6.2: Node and edge features used for training the GCNs. Features that are represented with more than 1 bit are one-hot encoded except the titration sites. If the node is not a titration site, all three bits are zero. The respective values to the one-hot encoding are additionally shown within the corresponding cell. TS: titration site

Subsequently, the edge and node features were calculated and the adjacency matrices were constructed for each molecule. Each of the adjacency matrices represents a fully bidirectional graph since chemical bonds don't have a direction. An overview of all calculated edge and node features can be found in table 6.2. All categorical non-

binary properties were represented as one-hot encoding. As the only non-categorical property, the length of the chemical bonds of each associated edge normalised between 0 and 1 was added as an attribute. Since the number of titratable groups was limited to a maximum of three, a vector of length three was also used to mark whether the node is a titratable group as well as which group it is. A node that does not belong to a titratable group would have three times 0 entered here as a node property. The maximum of three titratable groups would have the associated properties `[1, 0, 0]`, `[0, 1, 0]` and `[0, 0, 1]`. This numbering is necessary for the assignment of the predicted pK_a values to their respective titratable groups and is obtained through the preprocessing via MPP (cf. chapter 5.2.2 and 5.2.3). The target pK_a values (Y values) for each molecule were represented as a vector of length three. For molecules with fewer titration sites a mask value was added for the missing values. The pK_a values were normalized to the range of zero to one over all molecules including all titration sites. After all, the dataset compositions shown in table 6.1 were used for training and testing.

6.3 Machine learning

6.3.1 Base architecture and settings

The base architecture of the GCN used is shown in figure 6.1. The coloured markings divide the network into its individual functional parts, the red annotations represent parameters that are optimized in the hyperparameter search and architectural optimization. The parts of the network that feed data into the network from outside are shown in green. These are the input layers for edge and node features and the connectivity matrix, also called adjacency matrix, which flows into all graph convolutional layers. The grey part is a preceding multilayer perceptron (MLP) for processing the edge properties, whose output then flows into the NNConv layer[72]. This layer also receives the node properties and thus forms the input layer to the convolutional part of the network. This is followed by one or more FeaStConv layers[73], each followed by a dropout layer. The output of the convolutional part is then pooled and transferred to the subsequent MLP. Finally, the output layer follows, which is composed of three independent output layers. Depending on the number of titratable groups of the processed molecule(s), either the output layer with one, two or three output neurons is used and trained.

As optimizer Adaptive Moment Estimation (Adam) was used, the optimizer's settings were part of the hyperparameter search, though. The number of epochs each run was trained was set to 300, whereas the best epoch over all epochs was saved.

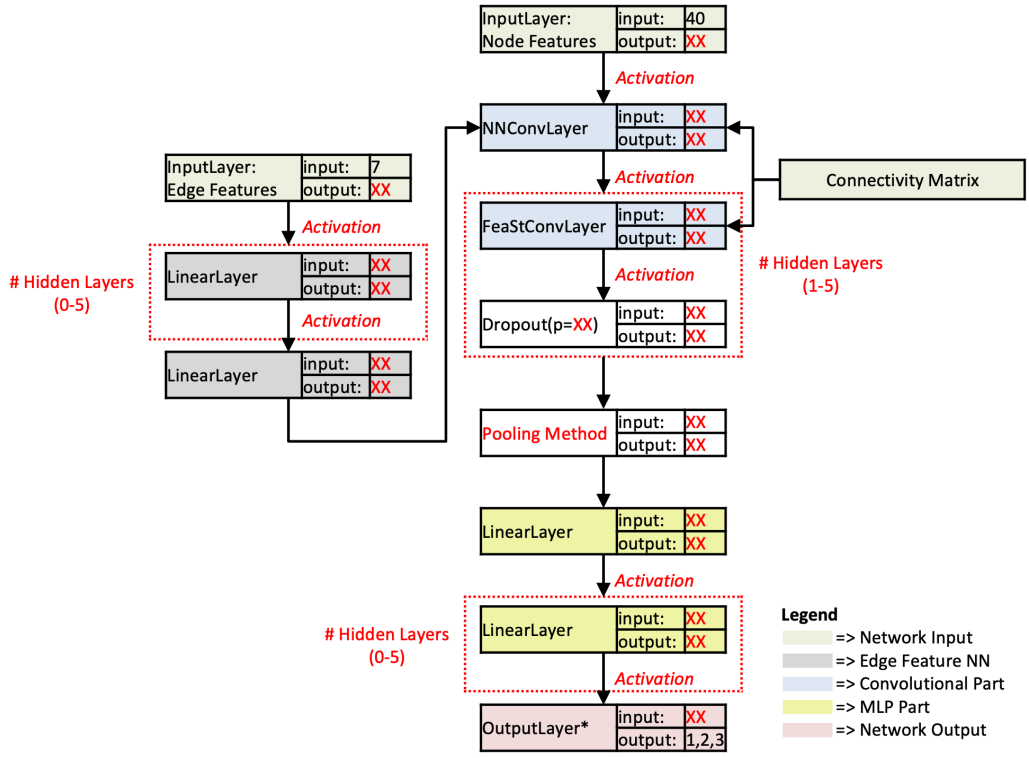


Figure 6.1: Base architecture used in hyperparameter and architecture optimization: The red annotations represent parameters that are optimized in the hyperparameter search and architectural optimization. The output layer has either one, two or three output neurons depending on the number of titratable groups of the processed molecule(s).

The best model was chosen by the average mean absolute error (MAE) of both testsets. A fixed random seed of 666 was set and a scheduler to reduce the learning rate on a plateau with a patience of 10 epochs was used. The reduction factor of the scheduler was also part of the hyperparameter search. As loss function L1 loss (MAE) was chosen.

6.3.2 Hyperparameter search

The hyperparameter search was carried out using a tree-structured Parzen estimator for parameter selection performing a Bayesian optimization. To do so the tool Optuna[12] was used. To monitor all performed training runs MLflow[13] was utilized to track the resulting models including the used parameters and calculated metrics as well as generated logs and figures. Additionally, the models can be easily compared afterwards based on all metrics through a web interface.

The hyperparameters that were optimized can be divided into two categories: architecture related settings and parameters related to the optimizer and scheduler. The only exception from this builds the batch size. This setting is basically a dataset

Parameters / Settings	**Values**
Activation Function	ReLU, LReLU, RReLU, tanh
Dropout Ratio	0.0 - 0.5 (continuous)
Hidden Conv. Layers	1 - 5
Hidden Edge Feature MLP Layer	0 - 5
Hidden MLP Layers	0 - 5
Neurons per Layer	16 - 256 (continuous)
Network Width	1 - 3
Pooling Method	Max Pooling, Mean Pooling, Add Pooling
Batch Size	16 - 512 (continuous)
Learning Rate	0.00001 - 0.1 (continuous, log scale)
Scheduler Reduction Factor	0.01 - 0.99 (continuous)
Weight Decay	0.00001 - 0.1 (continuous, log scale)

Table 6.3: Parameters and training settings improved during hyperparameter and architecture optimization

related parameter. For the scheduler only the learning rate reduction factor was investigated. The optimizer was adjusted in terms of the learning rate and the weight decay (L2 regularization). The architecture related parameters affect the amount of hidden layers in the edge feature MLP, the convolutional section as well as the final Multi-layer Perceptron (MLP) part of the network. These parts are marked with dashed red boxes in figure 6.1. Additionally, four different activation functions were examined: Hyperbolic tangent (tanh), rectified linear unit (ReLU), leaky ReLU (LReLU) with a negative slope of 0.01, and randomized leaky ReLU (RReLU) with a lower bound of $\frac{1}{8}$ and a upper bound of $\frac{1}{3}$. Also the number of neurons for all hidden layers was investigated. Here an additional parameter comes into place: the width of the network. This width is a factor between one and three to be multiplied with the number of neurons for the inner part of the hidden layers within the convolutional part of the network architecture. Please refer to the figure 6.2 as an example. Finally, the pooling method and the dropout ratio of all dropout layers were improved.

In total 500 different training configurations were trained on the Linux Cluster TU Dortmund 3 (LiDO3). A detailed overview of all parameters and their ranges can be viewed in table 6.3. The best model was selected based on the prediction performance for the Settimo *et al.*[14] and Novartis[15] external test datasets. Each trained model from the hyperparameter optimisation was used to predict these two external test datasets, followed by the calculation of the mean absolute error (MAE)

for the respective mono-, di- and triprotic sub-datasets. The calculated error values were then arithmetically averaged. The model with the best performance based on this averaged value, i.e. the lowest averaged MAE, was selected as the best model.

6.4 Results

6.4.1 Hyperparameter search

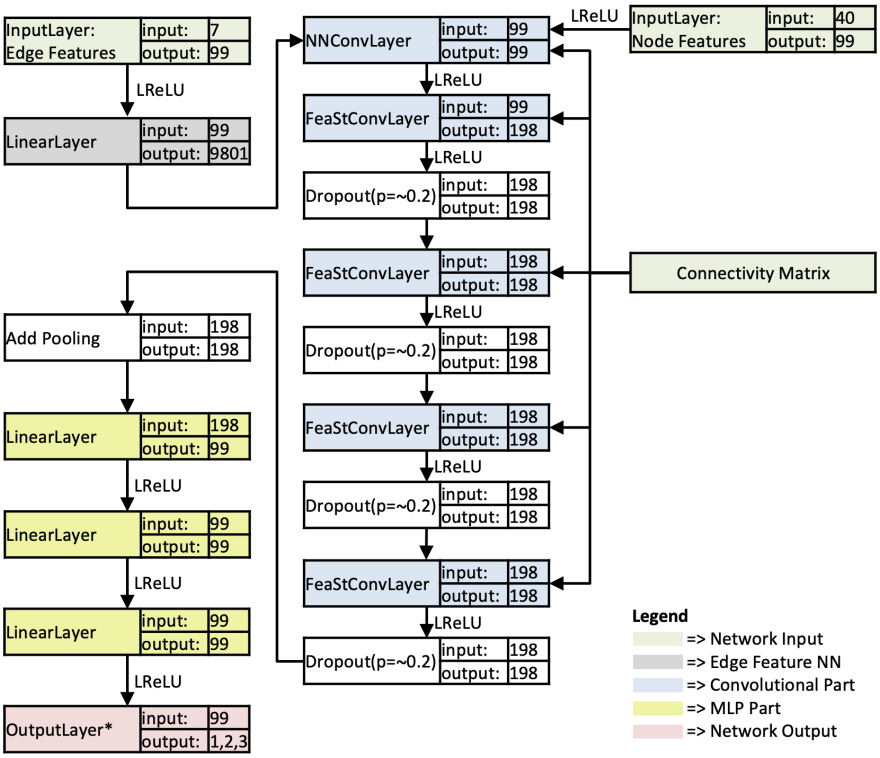


Figure 6.2: Architecture of best model found during hyperparameter search: In the edge feature part of the network the number of neurons that feed the NNConv layer[72] are by definition of the NNConv layer[72] input neurons times output neurons which leads to 9 801 neurons.

A table with all tested parameter combinations and their associated performance metrics can be viewed in the electronic appendix (see chapter A.6.1 for more details). Of the 500 trials, run 489 was the one with the best prediction performance defined as described in chapter 6.3.2. This is worth mentioning because the hyperparameter search is not a grid or random search, but uses Bayesian optimization to continuously optimise the selection of hyperparameters. The best model should therefore be one of the latter runs, as is the case here. The parameters and settings used for the best model can be found in table 6.4. The resulting architecture of the GCN is shown in figure 6.2. Here the effects of the network width parameter can be seen: The

Parameters / Settings	**Values**
Activation Function	LReLU
Dropout Ratio	~ 0.20358
Hidden Conv. Layers	4
Hidden Edge Feature MLP Layer	0
Hidden MLP Layers	2
Neurons per Layer	99
Network Width	2
Pooling Method	Add Pooling
Batch Size	109
Learning Rate	~ 0.00079
Scheduler Reduction Factor	~ 0.46415
Weight Decay	~ 0.00003

Table 6.4: Parameters and training settings of best model found during hyperparameter search: The values marked with a tilde are rounded to five decimal places. Since Optuna[12] works with double-precision floating-point values (float64) the actual value has 15 decimal places.

hidden FeaStConv layers[73] in this case have twice as many neurons as the other layers of the network. In addition, the number of 9 801 neurons as the output of the edge property MLP is remarkable. This number, by definition of the NNConv layer[72], is the number of input neurons times the number of output neurons[72]. In the further course of this work, only this specific model will be referred to.

6.4.2 Prediction performance

The optimal hyperparameters found in table 6.4 were then used to train a new model with the same data and settings, which was used below to evaluate model performance and to which the following values refer. The model achieved its best predictive performance at epoch 178. The statistical metrics calculated were root-mean-square error (RMSE), coefficient of determination (R^2) and MAE. Since mono-, di- and triprotic molecules were fed separately into the network during training each subset has its own performance values. For better understanding, the following statistical values were calculated over all three subsets combined for each training and test dataset, e.g. for the training dataset, all mono-, di- and triprotic pK_a values of all titration sites were concatenated separately for the experimental and predicted values and then the individual metrics were calculated.

	Train (1)	**Train (2)**	**Train (3)**	**Settimo (1)**	**Settimo (2)**	**Novartis (1)**	**Novartis (2)**	**Novartis (3)**
MAE	0.295	0.385	0.416	0.453	0.299	0.646	0.930	0.849
RMSE	0.456	0.592	0.628	0.624	0.461	0.879	1.192	1.079
$\mathbf{R^2}$	0.969	0.949	0.955	0.906	0.970	0.844	0.773	0.819

Table 6.5: Performance metrics of retrained model with optimal hyperparameters: All values were rounded to three decimal places. The number in brackets within the table header indicate the number of titration sites per molecule in this subset. Settimo *et al.*: [14]; Novartis: [15]

The training dataset reached a MAE of 0.323, a RMSE of 0.502 and a R^2 of 0.963. The Settimo *et al.*[14] testset performed with a MAE of 0.414, a RMSE of 0.587 and a R^2 of 0.929. Finally, the Novartis[15] testset got a MAE of 0.791, a RMSE of 1.048 and a R^2 of 0.808. All together the MAE of all testsets combined is at 0.748. All individually computed metrics for each dataset and for each mono-, di-, and triprotic subset can be viewed in table 6.5.

To gain a better understanding of the performance of the best model, the correlations between the experimental pK_a values and the predicted pK_a values for the training dataset and the two external test datasets are shown in figures 6.4 and 6.3. Corresponding to table 6.5, the correlation is shown separately for each of the columns contained therein. For the multiprotic datasets (figure 6.3 (B), (C), (E), (F), figure 6.4 (B)) a point plotted in the correlation plot corresponds to a titration site within the respective molecule. Sub-figures 6.3 (A) - (C) show the performance of the training dataset. A satisfactory correlation between prediction and experiment can be seen, although there are still significant outliers. For the Novartis test dataset, the scatter is relatively large, especially for the diprotic molecules, which is also reflected in the statistical values given in table 6.5. The Settimo *et al.*[14] diprotic and Novartis[15] triprotic sub-datasets should be treated with caution when evaluating the model performance, as the corresponding dataset size is relatively small (cf. table 6.1).

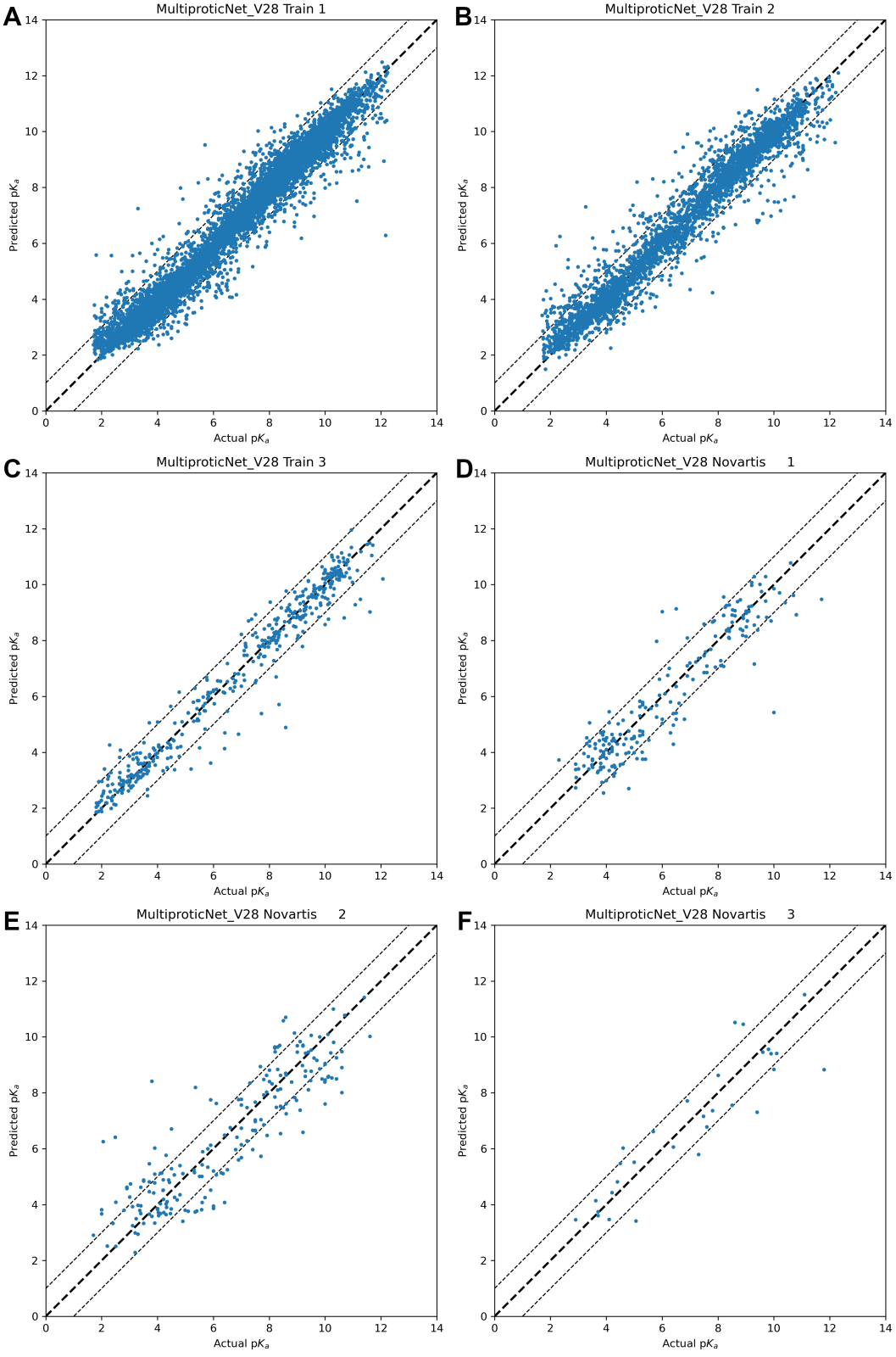


Figure 6.3: Correlation plots of training datasets and Novartis[15] testset for best model: Plots (A) - (C) show the performance of the training dataset regarding the mono-, di- and triprotic molecules, respectively. Analogous, plots (D) - (F) show the performance of the Novartis[15] test dataset. For the plots that show the correlation of multiprotic molecules, each point corresponds to one titration site of a molecule.

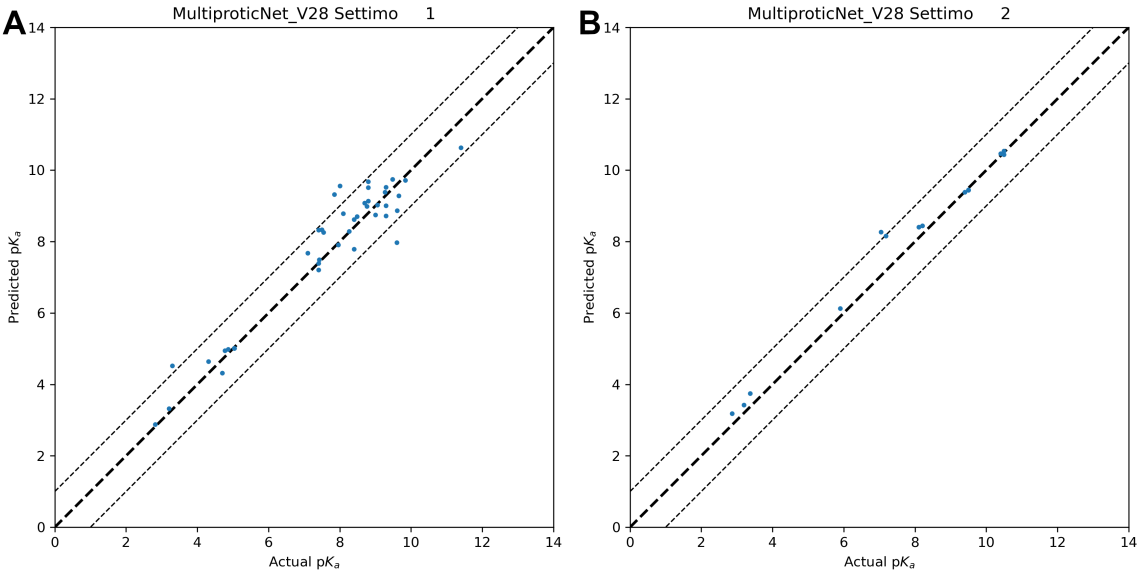


Figure 6.4: Correlation plots of Settimo *et al.*[14] testset for best model: Plots (A) and (B) show the performance of the mono- and diprotic molecules within the Settimo *et al.*[14] test dataset. In plot (B) each point corresponds to one titration site of a molecule.

6.4.3 Evaluation of SAMPL6 and 7 on retrained model

In order to evaluate the performance of the model in a more comparable way, an additional evaluation was carried out using the SAMPL6[16] and SAMPL7[17] datasets. These datasets were originally part of the training dataset (cf. chapters 6.1 and 6.2.1) and were included in the hyperparameter search. For a more meaningful evaluation of the model performance, a new model was trained in which, in contrast to the model discussed in chapter 6.4.2, the SAMPL6[16] and SAMPL7[17] datasets were used as external test datasets and the Novartis[15] and Settimo *et al.*[14] datasets were part of the training dataset. For this, a new preprocessing by MPP was performed as described in chapter 5. The model was then trained with the resulting overall training dataset and evaluated with the two SAMPL datasets. The same hyperparameters were used for training as were used in the hyperparameter search for the best model (cf. chapter 6.4.1 and table 6.4). The dataset composition is shown in table 6.6, analogous to table 6.1.

The results are shown analogous to the results of the best model presented in chapter 6.4.2. Table 6.7 shows the statistical values of the training and external test datasets, and figure 6.5 shows the correlation plots.

the following statistical values were calculated over all three subsets combined for each training and test dataset, e.g. for the training dataset, all mono-, di- and triprotic pK_a values of all titration sites were concatenated separately for the experimental

#Titration Sites	Training Dataset #Molecules	SAMPL6[16] #Molecules	SAMPL7[17] #Molecules
1	9 854	17	20
2	1 935	1	-
3	153	-	-
Summary	11 942	18	20

Table 6.6: Dataset composition used for training and testing the retrained GCNs.

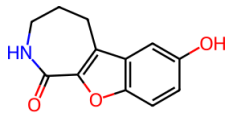
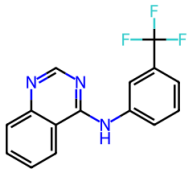
and predicted values and then the individual metrics were calculated.

Combining the set of mono-, di- and triprotic molecules of the training dataset and calculating the statistical metrics over all titration sites, i.e. all mono-, di- and triprotic pK_a values of all titration sites were concatenated separately for the experimental and predicted values, led to a MAE of 0.352, a RMSE of 0.539 and a R^2 of 0.957. The filtered monoprotic SAMPL6[16] testset performed with a MAE of 0.442, a RMSE of 0.592 and a R^2 of 0.915. The filtered diprotic SAMPL6[16] subset consisting only of a single molecule reached a MAE of 0.136, a RMSE of 0.145 and a R^2 of 0.995. Combining the set of monoprotic and diprotic molecules and calculating the statistical metrics over all titration sites led to a MAE of 0.410, a RMSE of 0.561 and a R^2 of 0.926. Due to the filtering, the molecules SM11, SM12, SM14, SM18, SM21 and SM22 were not included in the calculations. Additionally, the molecules SM06 and SM16 were considered to be monoprotic since the predicted value by ChemAxon Marvin[8] (CXM) for one of the titration sites of each molecule was outside of the pK_a range of 2-12. Therefore, these statistical values cannot be compared with the published statistical values of SAMPL6[16]. Finally, the SAMPL7[17] testset consisting only of monoprotic molecules got a MAE of 0.722, a RMSE of 0.907 and a R^2 of 0.851. All together the mean MAE of all testsets is at 0.433. It is worth noting that the statistical values for the training datasets are only slightly different from those of the best model from the previous chapter (see table 6.5). This equal performance is also reflected in the correlations (A) - (C) shown in figure 6.5. Here, an equally intense scatter can be seen in comparison to figure 6.3. This comparable performance shows that the hyperparameters optimized in chapter 6.4.1 ensure a good and reproducible fit of the model to the data, even if the training dataset is slightly changed due to the replacement of the external test datasets. Also noteworthy is the very good performance for the diprotic SAMPL6[16] data subset. However, as this subset contains only a single molecule, the performance based on this single molecule is of very little relevance in terms of the overall performance of the model.

	Train (1)	Train (2)	Train (3)	SAMPL6 (1)	SAMPL6 (2)	SAMPL7 (1)
MAE	0.329	0.398	0.461	0.442	0.136	0.722
RMSE	0.498	0.614	0.692	0.592	0.145	0.907
R^2	0.962	0.945	0.945	0.915	0.995	0.851

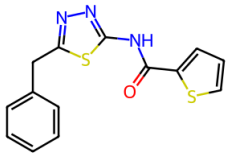
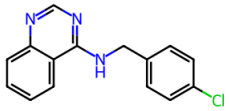
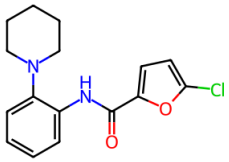
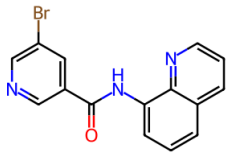
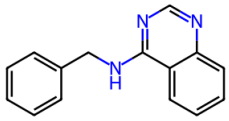
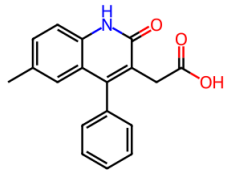
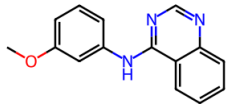
Table 6.7: Performance metrics of retrained model: All values were rounded to three decimal places. The number in brackets within the table header indicate the number of titration sites per molecule in this subset. SAMPL6: [16]; SAMPL7: [17]

In order to facilitate a more comprehensive understanding and comparison with other pK_a prediction techniques and the participants of the SAMPL6[16] and SAMPL7[17] pK_a challenges, the results of the individual molecules from the two aforementioned challenges are presented in table 6.8. The table includes the molecule IDs from the respective challenges, an image of the corresponding molecule, the experimental pK_a value(s), the predictions of CXM, and the prediction of GMPP. In the event that a prediction of GMPP was not possible or that the molecules were discarded during the preprocessing stage, a corresponding comment can be found in the final column of the table.

ID	Image	Exp. pK_a	CXM pred. pK_a	GMPP pred. pK_a	Comment
SM01		pK_a1=9.53	pK_a1=9.43	pK_a1=9.60	
SM02		pK_a1=5.03	pK_a1=4.01	pK_a1=5.12	

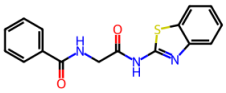
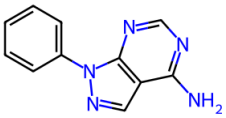
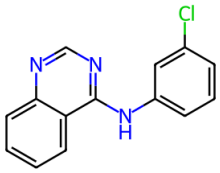
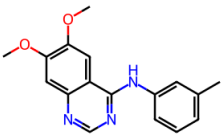
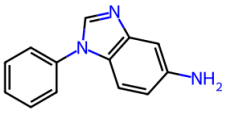
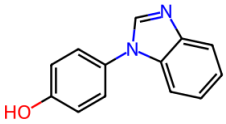
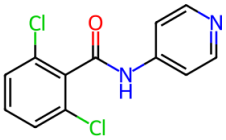
Continued on next page

Table 6.8: Overview of SAMPL6[16] molecules SM01 to SM24 and SAMPL7[17] molecules SM25 to SM46 including the CXM and GMPP predictions. The molecules for which no GMPP predictions are given were not included in the calculation of the statistical metrics.

ID	Image	Exp. pK_a	CXM pred. pK_a	GMPP pred. pK_a	Comment
SM03		pK_a1=7.02	pK_a1=7.68	pK_a1=6.81	
SM04		pK_a1=6.02	pK_a1=4.83	pK_a1=5.68	
SM05		pK_a1=4.59	pK_a1=4.19	pK_a1=4.11	
SM06		pK_a1=3.03 pK_a2=11.74	pK_a1=3.42 pK_a2=13.85	pK_a1=4.28	CXM predicted pK_a2 outside of the range 2-12, therefore the titration site was discarded during MPP preprocessing
SM07		pK_a1=6.08	pK_a1=4.83	pK_a1=5.80	
SM08		pK_a1=4.22	pK_a1=4.20	pK_a1=4.09	
SM09		pK_a1=5.37	pK_a1=4.00	pK_a1=5.52	

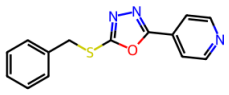
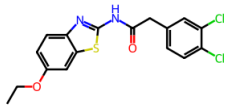
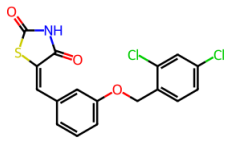
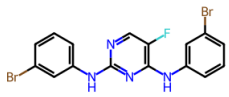
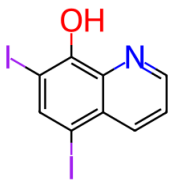
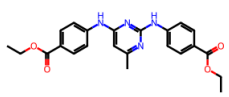
Continued on next page

Table 6.8: Overview of SAMPL6[16] molecules SM01 to SM24 and SAMPL7[17] molecules SM25 to SM46 including the CXM and GMPP predictions. The molecules for which no GMPP predictions are given were not included in the calculation of the statistical metrics. (Continued)

ID	Image	Exp. pK_a	CXM pred. pK_a	GMPP pred. pK_a	Comment
SM10		pK_a1=9.02	pK_a1=7.77	pK_a1=8.85	
SM11		pK_a1=3.89	pK_a1=3.42	-	Molecule was part of the training set through another dataset (cf. table 6.6), therefore it was discarded
SM12		pK_a1=5.28	pK_a1=4.01	-	Molecule was part of the training set through another dataset (cf. table 6.6), therefore it was discarded
SM13		pK_a1=5.77	pK_a1=4.68	pK_a1=6.11	
SM14		pK_a1=2.58 pK_a2=5.30	pK_a1=0.93 pK_a2=6.40	-	CXM predicted pK_a1 outside of the range 2-12, therefore the titration site was discarded during MPP preprocessing
SM15		pK_a1=4.70 pK_a2=8.94	pK_a1=4.91 pK_a2=10.64	pK_a1=4.89 pK_a2=9.02	
SM16		pK_a1=5.37 pK_a2=10.65	pK_a1=5.61 pK_a2=13.97	pK_a1=4.89	CXM predicted pK_a2 outside of the range 2-12, therefore the titration site was discarded during MPP preprocessing

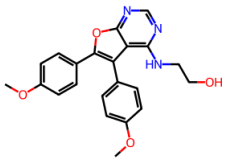
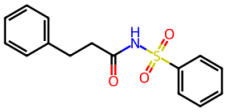
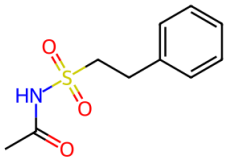
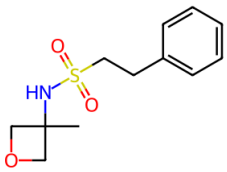
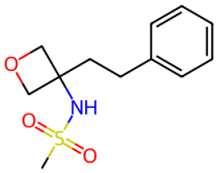
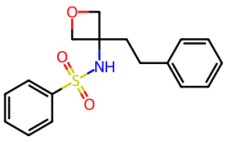
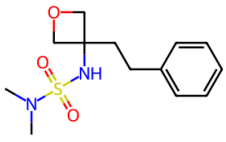
Continued on next page

Table 6.8: Overview of SAMPL6[16] molecules SM01 to SM24 and SAMPL7[17] molecules SM25 to SM46 including the CXM and GMPP predictions. The molecules for which no GMPP predictions are given were not included in the calculation of the statistical metrics. (Continued)

ID	Image	Exp. pK_a	CXM pred. pK_a	GMPP pred. pK_a	Comment
SM17		pK_a1=3.16	pK_a1=2.48	pK_a1=3.67	
SM18		pK_a1=2.15 pK_a2=9.58 pK_a3=11.02	pK_a1=5.40 pK_a2=8.04 pK_a3=9.93	-	The generated SMARTS tree (cf. chapter 5.2.2) was not able to identify all three titration sites, therefore the molecule was discarded
SM19		pK_a1=9.56	pK_a1=7.89	pK_a1=9.62	
SM20		pK_a1=5.70	pK_a1=6.90	pK_a1=7.13	
SM21		pK_a1=4.10	pK_a1=1.95	-	CXM predicted pK_a1 outside of the range 2-12, therefore the molecule was discarded during MPP preprocessing
SM22		pK_a1=2.40 pK_a2=7.43	pK_a1=3.34 pK_a2=7.39	-	Molecule was part of the training set through another dataset (cf. table 6.6), therefore it was discarded
SM23		pK_a1=5.45	pK_a1=5.01	pK_a1=6.28	

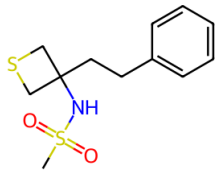
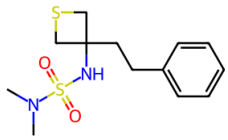
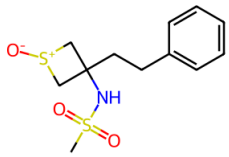
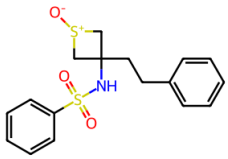
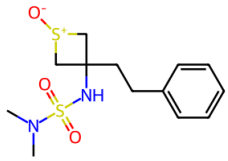
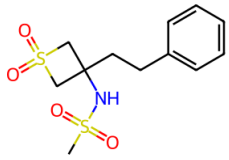
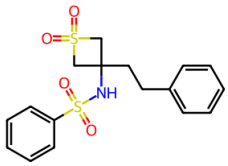
Continued on next page

Table 6.8: Overview of SAMPL6[16] molecules SM01 to SM24 and SAMPL7[17] molecules SM25 to SM46 including the CXM and GMPP predictions. The molecules for which no GMPP predictions are given were not included in the calculation of the statistical metrics. (Continued)

ID	Image	Exp. pK_a	CXM pred. pK_a	GMPP pred. pK_a	Comment
SM24		pK_a1=2.60	pK_a1=3.09	pK_a1=3.29	
SM25		pK_a1=4.49	pK_a1=4.24	pK_a1=5.07	
SM26		pK_a1=4.91	pK_a1=4.09	pK_a1=5.01	
SM27		pK_a1=10.45	pK_a1=9.59	pK_a1=10.15	
SM29		pK_a1=10.05	pK_a1=9.59	pK_a1=10.76	
SM30		pK_a1=10.29	pK_a1=10.15	pK_a1=10.36	
SM31		pK_a1=11.02	pK_a1=9.43	pK_a1=10.43	

Continued on next page

Table 6.8: Overview of SAMPL6[16] molecules SM01 to SM24 and SAMPL7[17] molecules SM25 to SM46 including the CXM and GMPP predictions. The molecules for which no GMPP predictions are given were not included in the calculation of the statistical metrics. (Continued)

ID	Image	Exp. pK_a	CXM pred. pK_a	GMPP pred. pK_a	Comment
SM32		pK_a1=10.45	pK_a1=10.70	pK_a1=11.20	
SM34		pK_a1=11.93	pK_a1=10.54	pK_a1=10.90	
SM35		pK_a1=9.87	pK_a1=10.16	pK_a1=9.87	
SM36		pK_a1=9.80	pK_a1=10.16	pK_a1=9.66	
SM37		pK_a1=10.33	pK_a1=10.07	pK_a1=9.47	
SM38		pK_a1=9.44	pK_a1=9.07	pK_a1=9.75	
SM39		pK_a1=10.22	pK_a1=10.15	pK_a1=9.52	

Continued on next page

Table 6.8: Overview of SAMPL6[16] molecules SM01 to SM24 and SAMPL7[17] molecules SM25 to SM46 including the CXM and GMPP predictions. The molecules for which no GMPP predictions are given were not included in the calculation of the statistical metrics. (Continued)

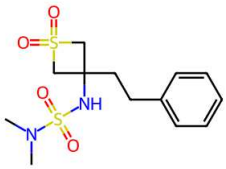
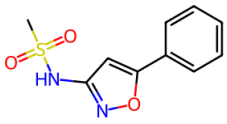
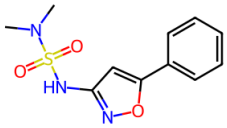
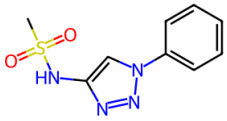
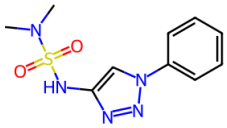
ID	Image	Exp. pK_a	CXM pred. pK_a	GMPP pred. pK_a	Comment
SM40		pK_a1=9.58	pK_a1=9.00	pK_a1=9.25	
SM41		pK_a1=5.22	pK_a1=6.44	pK_a1=4.12	
SM42		pK_a1=6.62	pK_a1=5.82	pK_a1=4.30	
SM43		pK_a1=5.62	pK_a1=7.17	pK_a1=4.50	
SM44		pK_a1=6.34	pK_a1=6.48	pK_a1=5.00	
SM45		pK_a1=5.93	pK_a1=5.72	pK_a1=5.26	
SM46		pK_a1=6.42	pK_a1=7.20	pK_a1=5.02	

Table 6.8: Overview of SAMPL6[16] molecules SM01 to SM24 and SAMPL7[17] molecules SM25 to SM46 including the CXM and GMPP predictions. The molecules for which no GMPP predictions are given were not included in the calculation of the statistical metrics. (Continued)

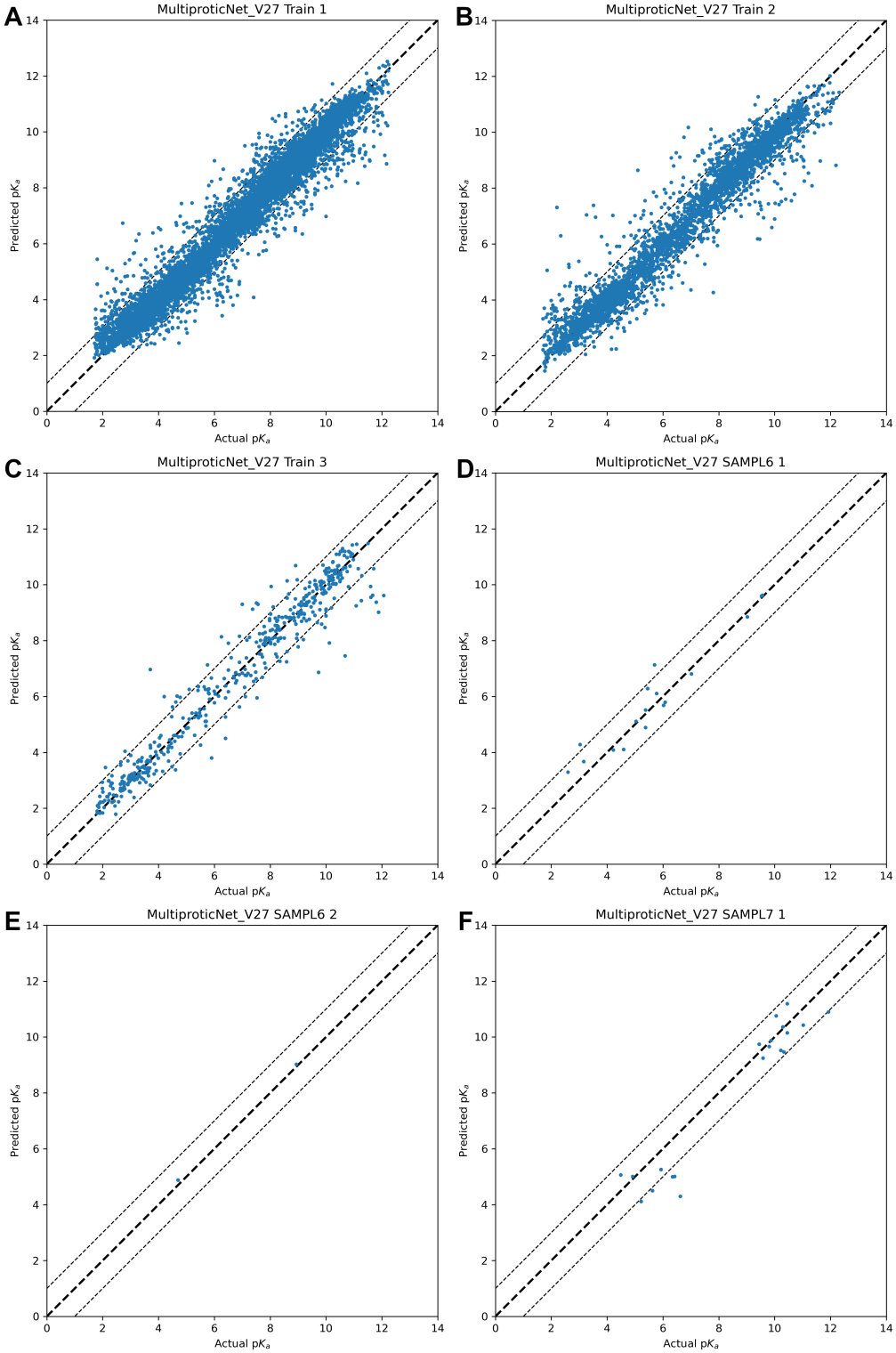


Figure 6.5: Correlation plots of training datasets and SAMPL6[16] and SAMPL7[17] testsets for retrained model: Plots (A) - (C) show the performance of the training dataset regarding the mono-, di- and triprotic molecules, respectively. Analogous, plots (D) - (F) show the performance of the SAMPL6[16] and SAMPL7[17] test datasets. For the plots that show the correlation of multiprotic molecules, each point corresponds to one titration site of a molecule.

6.5 Discussion

Already during the definition of the research goal, the decision was made that the research and the resulting machine learning models should be based exclusively on experimentally determined data. This principle has been consistently adhered to, but this inevitably has the disadvantage of having less data available for evaluation and training. Other pK_a prediction tools published in the meantime either additionally or exclusively use predicted pK_a values from commercial programs for training, such as MolGpKa[120]. While this provides significantly more training data, it does not allow the model to become better than the underlying pK_a prediction program used. However, a combination of experimental and predicted pK_a values could certainly result in an improvement in model quality by using the significantly larger dataset of predicted values for pre-training and the experimental values for fine-tuning the model.

Due to the use of the MPP pipeline for preprocessing of the datasets and assignment of the experimentally determined values to the individual titratable groups, it cannot be excluded that incorrect assignments may have taken place within this step. This has the consequence that the model trained with it could also adapt these errors and replicate them in the prediction. Because of the necessarily equal preprocessing of the test datasets, such errors would not be verifiable by this means. By carefully evaluating the MPP results and filtering data points where uncertainty is too high, an attempt was made to reduce such errors to an absolute minimum.

For training the machine learning model, the molecules were used in their protonated form, which in this case is protonated to pH 0. Thus, each basic titration site is in a positively charged form and each acidic titration site is in a neutral form. This protonation form was chosen because the molecules are thus present with the highest number of total atoms and thus the graph for machine learning is larger, respectively contains more information. Other options would be to use the molecules either in neutral, uncharged form or in the form where each titration site is charged. In the first method, no information about any titration sites within the molecule and the pK_a range used can be taken directly from the molecular structure itself and is therefore less suitable compared to the other possibilities. In the second method, all titration sites are present in charged form, i.e. acids in negatively charged form and bases in positively charged form. This means that the model can already recognize the titration sites on the basis of the charge and is therefore also a suitable method for preprocessing. However, the molecule may thus be present in a naturally non-occurring protonation state, which is why this method was also not used. Since the protonation state has a crucial influence on the model to solve the underlying

problem, the prediction of the model is particularly dependent on the protonation state of the input molecule. During training, the protonation sites and their class (basic or acidic) were available from the CXM prediction due to the preprocessing in chapter 5. In addition to the available experimental pK_a values, the molecule was "correctly" protonated or deprotonated to pH 0 based on these values. Of course, this information is not available for a subsequent prediction of the pK_a values of a molecule, which is why the protonation state can only be determined approximately. Due to the aforementioned significant influence of the protonation state on the prediction, the accuracy can therefore vary significantly if the approximated protonation state is not the "correct" state at pH 0. Although the use of neutral molecules for training, as described above, would solve this problem, it is likely that the overall quality of the model could be reduced as a result. Nevertheless, this solution would be preferable in a further optimization iteration of the model due to the improved usability of the model.

The resulting machine learning model for predicting pK_a values is limited to molecules with a maximum of three titration sites. In principle, it can of course also predict up to three pK_a values for a molecule with more than three titratable groups, but these could subsequently no longer be reliably assigned to the associated group. The limitation was necessary, however, because the dataset does not allow training for molecules with more than three titration sites. As shown in table 6.1, the training dataset already contains only 142 molecules with three titratable groups. This represents only 1.22% of the total training dataset available. This limitation can only be overcome with more experimentally determined data for molecules with more than three titration sites, by expanding the dataset with predicted pK_a values, or by using an entirely different prediction approach where the pK_a values of the titratable groups are predicted individually and not as part of the total molecule. The limited size of the training dataset is also the reason why a separate validation dataset was not used for model training. An additional reduction of the training dataset by, for example, 10% would have presumably had a negative impact on model performance, especially for multiprotic molecules. It was therefore decided to select the best model within the hyperparameter optimization based on the performance of the external test sets. Since both the Settimo *et al.*[14] literature dataset and the Novartis[15] pharmaceutical research dataset were the key landmarks with respect to the overall goal of this work in terms of a competitive pK_a prediction tool, this decision has been justifiable after weighing the advantages and disadvantages.

The final model is, as explained in the previous chapters, a graph convolutional neural network and is based in its fundamentals on the research results of David Bushiri, which were developed in the course of his master's thesis within the re-

search group on the topic of monoprotic pK_a prediction. Due to the promising results, the basic structure as well as the selection of the convolutional layer implementations NNConv[72] and FeaStConv[73] were kept. All other architectural details and parameters, the transformation of the molecules into graphs and the selection of node and edge features as well as training parameters themselves were adapted or optimized using hyperparameter search. Additionally, the network was modified for use with multiprotic molecules. Other machine learning algorithms such as Random Forest or Support Vector Machine as well as other types of neural networks such as classical MLPs or Message Passing Neural Networks (MPNNs) were not pursued further for multiprotic pK_a prediction in the course of this work. Especially MPNNs, which also work with graphs as input data, could be a promising approach for this problem as well.

An important aspect of the final neural network architecture chosen that could be improved is that the 3D structure in terms of the spatial position of the atoms is not directly incorporated into the training. The reason for this is that at the time of the research during this thesis, there was no convolutional layer implemented within PyTorch Geometric that included edges, edge features, nodes, node features, and 3D coordinates together for computation within the layer. Implemented layers in PyTorch Geometric that include 3D coordinates are by definition intended for use with point clouds, i.e., a collection of nodes without edges, and are therefore unsuitable for the purpose here. Indirectly, the 3D information was used by adding the bond lengths as edge features, however, a full use of the 3D coordinates should be able to improve the model performance. For this purpose, either a dedicated layer can be implemented or an existing layer can be extended to handle the 3D coordinates, or the coordinates could be added in normalized form to the nodes as node features. In addition, it would be possible to model intramolecular interactions using the RadiusGraph transformation, where edges are added between nodes if they are closer than a defined distance within 3D space.

For evaluation of model quality and selection of the model with the best performance during hyperparameter search, the two external testsets from Settimo *et al.*[14] and Novartis[15] were used, as described earlier. This is intended to look at the prediction quality for datasets from the literature as well as for datasets from pharmaceutical research. For both the developed monoprotic model (cf. chapter 4), the final multiprotic model, and ChemAxon Marvin[8], the Novartis[15] dataset appeared to be more challenging, resulting in poorer model performance in all cases than for the Settimo *et al.*[14] dataset. This may be either because the experimentally determined values are incorrect or inaccurate, or the molecules are more complex and thus located elsewhere in the chemical space studied. Upon closer examination of

the molecules from both sources, the latter hypothesis can be considered more likely, as the molecules within the pharmaceutical datasets differ significantly in complexity from those in the literature datasets. Looking more closely at the results of the model evaluation from table 6.5, it can be seen that the diprotic molecules from Settimo *et al.*[14] and the triprotic molecules from Novartis[15] perform better than the monoprotic and diprotic molecules from the respective datasets. A possible rationale for this is provided by the dataset size of the two aforementioned subsets. The diprotic Settimo *et al.*[14] subset counts only seven molecules, and the triprotic Novartis[15] subset counts only ten molecules. With such a small number of test molecules, the reported performance can only be extrapolated with caution to diprotic molecules from the literature and triprotic molecules from the pharmaceutical industry in general. An evaluation of the model with larger external test datasets is necessary for a more robust statement about the model quality for multiprotic molecules from the mentioned fields.

During the development of the model, we have always focused on reproducibility and consistency. Unfortunately, some parts of the implementation have limited reproducibility and will be discussed in more detail in the following. The first element concerns the generation of 3D conformations during the conversion of molecules into graphs for training the GCN. In the implementation used in RDKit[114], ETKDG[116] is used to generate the conformers, which uses a random distance matrix. This distance matrix is different for each run, so the resulting 3D conformers may be slightly different. One way to counteract this behaviour in favour of reproducibility is to set a fixed random seed for each function call. The use of a random seed was already implemented in the final version of the code and is therefore described in chapter 6.2.2, although this change came into effect after the hyperparameter search and training of the best model whose results were described in chapters 6.4.1 and 6.4.2, but before the re-training of the model described in chapter 6.4.3. The second element concerns the calculation of bond lengths for use as edge attributes. By default, distances are normalised to an interval of $[0, 1]$, as this range of values is better for training neural networks. This normalisation is based on the lowest and highest distance value within the individual graph. This can lead to different normalised values for different 3D conformers, as well as for different molecules for the same bond length, which can affect the resulting prediction. In this case it is advisable to set the minimum and maximum values used for normalisation to a reasonable range for bond lengths. The third element refers to the protonation states used for training. To generate the most appropriate protonation for training, this was determined by the pK_a value and the type of titration site (acidic or basic). Obviously, there is no reference pK_a available for the prediction of

new molecules. Therefore, protonation is performed on the basis of fixed rules per element. As a result, the protonation state of the molecule can change, which in turn affects the quality of the prediction. This can be improved by using neutralised molecules for training and prediction. This ensures reproducibility, but the training itself may produce less favourable results.

In order to make a direct comparison with the reference tool CXM, the statistical performance of this tool was calculated for the training dataset used in chapter 6.4.2 and for the two external test datasets Settimo *et al.*[14] and Novartis[15]. The calculation was performed using the same methodology as in chapters 6.4.2 and 6.4.3: All pK_a values of all titration sites of the mono-, di- and triprotic molecules were concatenated separately for the experimental values used for training and evaluation of the model, as well as for the values predicted by CXM. The statistical metrics were then calculated over these values. For the training dataset, CXM achieves a MAE of 0.565, a RMSE of 0.740 and a R^2 of 0.919. For Settimo *et al.*[14], CXM achieves a MAE of 0.619, a RMSE of 0.767, and a R^2 of 0.879. Finally, for the Novartis[15] test dataset, CXM achieves a MAE of 0.656, a RMSE of 0.827 and a R^2 of 0.880. Using identical methodology, an analysis was also performed on the SAMPL6[16] and SAMPL7[17] challenge molecules. For SAMPL6[16], CXM achieves a MAE of 0.809, a RMSE of 0.960 and a R^2 of 0.784. To complete the analysis, CXM achieves a MAE of 0.620, a RMSE of 0.781 and a R^2 of 0.890 for the SAMPL7[17] molecules. Compared to the values reported in chapter 6.4.2 and 6.4.3, GMPP performs better in the statistical evaluation for the Settimo *et al.*[14] literature dataset as well as for the SAMPL6[16] dataset. CXM performs better than GMPP for the Novartis[15] industry dataset along with the SAMPL7[17] dataset. The better performance of CXM on the industry dataset with slightly worse performance on the literature dataset was already observed in chapter 4 in the context of model development for monoprotic molecules. Since GMPP was not trained exclusively but mainly on literature data, this could explain the better performance for Settimo *et al.*[14] compared to CXM, while CXM presumably has a larger training dataset available, which probably contains more data from the chemical space of the pharmaceutical industry and therefore performs better for the Novartis[15] test dataset. It is also possible that the model behind CXM achieves better generalization and therefore varies less between datasets from different chemical spaces. A similar explanation could also account for the better performance of CXM on SAMPL7[17]. The sulfonyl group and the prominent four-atomic heterocycle are characteristic for the SAMPL7[17] structures. These structural fragments are probably underrepresented in the training dataset for GMPP, which may be why the model performs worse here. CXM may have an advantage here due to the presumably larger and more

diverse training dataset. However, since CXM does not provide any information about the training dataset used or its composition, it is not possible to investigate this further.

6.6 Conclusion

With GMPP a pK_a prediction tool was developed that can predict pK_a values of up to three titratable groups per molecule in the range of two to twelve. It is based on a graph convolutional neural network and was trained using only experimentally determined data and evaluated with external datasets from both the literature and the pharmaceutical industry. The neural network architecture and training parameters were optimized with an extensive hyperparameter search using Optuna. In this process, all the studied architectures and their training parameters were evaluated using the external test datasets and the results as well as all the related settings and mappings were tracked and stored using MLflow. The model can be further optimized in building work and research by removing the limitation to a maximum of three titration sites per molecule and extending the training dataset. An evaluation of further basic architectures such as MPNNs or the integration of the 3D coordinates of the individual atoms per molecule into the training of the model could further increase the prediction quality of the model. Nevertheless, with GMPP a pK_a prediction tool has been developed that can compete with commercial applications in its precision and enriches the set of open-source pK_a predictors by a model that is not limited to most acidic/most basic pK_a predictions.

Chapter 7

Conclusion and outlook

As part of this thesis and associated research, a machine learning model for multiprotic pK_a prediction was developed. The entire process, from data collection and standardisation, through elaboration and preprocessing, to model conceptualisation and development, was carried out independently and from scratch. The experimental pK_a data were collected and standardised through extensive literature searches, searches in public databases and by establishing and maintaining collaborations with industrial partners. Great care was taken to ensure the quality of the data, which was analysed and evaluated at every stage of the research.

The next step was to focus on monoprotic molecules in order to gain initial experience with the datasets collected so far in the context of machine learning. In addition to analysing the monoprotic subset of the datasets, several machine learning algorithms were tested and their suitability for predicting monoprotic pK_a values was investigated. Several abstractions necessary for the selected machine learning algorithms, i.e. the representation in terms of molecular fingerprints or descriptors, were also evaluated and combined. The qualitative assessment of the performance of the individual combinations of machine learning algorithms and input data was carried out using 5-fold cross-validation and external test sets from the literature and the pharmaceutical industry. The Random Forest algorithm performed best and was able to match the performance of the commercial tool ChemAxon Marvin[8] (CXM). The analysis results, datasets, model and all associated code have been made publicly available and have successfully passed a peer review process.

After the successful development of a pK_a prediction model for monoprotic molecules, the extension to multiprotic molecules was started. For this purpose, all the collected datasets had to be standardised and the problem of the assignment of the experimentally determined pK_a values to the corresponding titration sites had to be

solved. For this purpose, the Multiprotic pK_a Processor (MPP) tool was developed, which covers the entire process from the starting point of a collection of different datasets from different sources to the final processed training and test datasets that can be used directly for machine learning. The MPP pipeline includes combining the datasets, preparing, standardising and filtering the data points according to defined criteria, statistical analysis, building a SMARTS tree to identify and locate titratable groups, and assigning pK_a values to the associated groups. The results are high quality datasets that have been used to develop a multiprotic pK_a prediction model.

Finally, a machine learning model has been developed that is capable of predicting the pK_a values of up to three titration sites per molecule. For this purpose, the Graph-based Multiprotic pK_a Predictor (GMPP) tool was developed which, based on the basic architecture of a Graph Convolutional Neural Network (GCN), performs architecture and hyperparameter optimisation to find the best possible model. A detailed tracking of all tested models and their evaluation results is also performed. GMPP is extensively parameterizable and modularly extensible through individual network architectures implemented in PyTorch and PyTorch Geometric[11] (PyG). In addition, the conversion of the molecules of the used datasets into graphs and the calculation of all edge and node features are handled, as well as the graphical processing of the training process and its results.

In further work some elements can be further optimised. Within MPP, the algorithm for assigning pK_a values to titration sites should be improved so that fewer data points need to be filtered out. It would be possible to retain in the final dataset not only molecules for which all titratable groups could be assigned, but also molecules for which experimental pK_a values for individual titratable groups are not available. Such molecules could still be useful for training the GCN and further improving the performance of the model. However, the quality and accuracy of the dataset should not be compromised. Another possibility would be to use MPP together with other or even several commercial prediction tools. Currently, CXM prediction plays an important role in assigning pK_a values to their corresponding titration sites. Other prediction tools should also be evaluated or combined to provide a more robust and less biased result. The quality of the data base can be further improved by targeted validation of outliers in the laboratory using a pK_a measuring instrument. However, such a procedure should always be carried out in consideration of the cost-benefit ratio.

The trained GCN for multiprotic pK_a prediction can presumably also be improved by a larger hyperparameter search as well as by evaluating different basic archi-

tectures such as Message Passing Neural Networks (MPNNs). Furthermore, the limitation to a maximum of three titration sites per molecule should be addressed. This could be done by switching from graph-level predictions to node-level predictions. Involving the 3D coordinates in the training should also result in an increase in prediction quality. The 3D structures can also be optimised in this process by a quantum mechanical level calculation with e.g. Gaussian[121]. The use of quantum mechanical descriptors as node or edge properties also represents an optimisation option for the GCN.

Appendix A

Supplementary material

A.1 Value Assignment Algorithms

The following Python[74] code snippets describe the three different value assignment algorithms used in chapter 5.2.4 to assign the pK_a values to their respective titration sites based on the prediction of ChemAxon Marvin[8] (CXM). Each of the individual methods illustrates the process for a single molecule. In the context of these codes, the variable `pred` contains the pK_a values predicted by CXM for each titration site as a list, and `atoms` contains the corresponding atom IDs as a list in the same order. The variable `exp` contains the experimental pK_a values in ascending order as a further list. The methods or constants beginning with `np.` are taken from the Python library NumPy[122], the method `mean_absolute_error()` comes from the Python library Scikit-learn[79]. All other functions used belong to the Python standard library. The result of the assignment is saved in the variable `assigned_exp`.

A.1.1 Method (1)

```
pred = dict(zip(atoms, pred))
assigned_exp = np.full(len(atoms), np.nan)
for ev in exp:
    last_err = 100
    last_ai = -1
    best_pos = -1
    for i, ai in enumerate(atoms):
        err = abs(pred[ai] - ev)
        if err < last_err:
```

```
            last_err = err
            best_pos = i
            last_ai = ai
    if np.isnan(assigned_exp[best_pos]) or \
      last_err < abs(pred[last_ai] - assigned_exp[best_pos]):
        assigned_exp[best_pos] = ev
```

A.1.2 Method (2)

```
n_pred, n_exp = len(pred), len(exp)
assigned_exp = np.empty(shape=n_pred)
err_mat = np.empty(shape=(n_exp, n_pred))
for ei in range(n_exp):
    for pi in range(n_pred):
        err_mat[ei, pi] = abs(pred[pi] - exp[ei])
# For keeping the index while reducing err_mat
x_ix = list(range(n_exp))
y_ix = list(range(n_pred))
for _ in range(n_pred):
    m = np.argmin(err_mat)  # Returns the index of the flattened matrix
    x = m // err_mat.shape[1]
    y = m % err_mat.shape[1]
    assigned_exp[y_ix[y]] = exp[x_ix[x]]
    err_mat = np.delete(err_mat, x, 0)  # Remove row
    err_mat = np.delete(err_mat, y, 1)  # Remove col
    del x_ix[x]
    del y_ix[y]
```

A.1.3 Method (3)

```
permut_exp = list(permutations(exp, r=len(pred)))
err = [mean_absolute_error(pred, pe) for pe in permut_exp]
best_i = err.index(min(err))
assigned_exp = np.array(permut_exp[best_i])
```

A.2 Full validation results

In figure A.1 and figure A.2 the full results of the validation of the location strategies including the timings are shown. Within the strategies from `R1 with R2 valid.` to `R1 with R5 valid.` a plain radius 1 search was performed following a validation of all found matches with the SMARTS pattern of a higher radius that contain the radius 1 pattern as substructure. This is also a SMARTS tree based approach but the trees only have two levels and the single strategies need to be calculated separately.

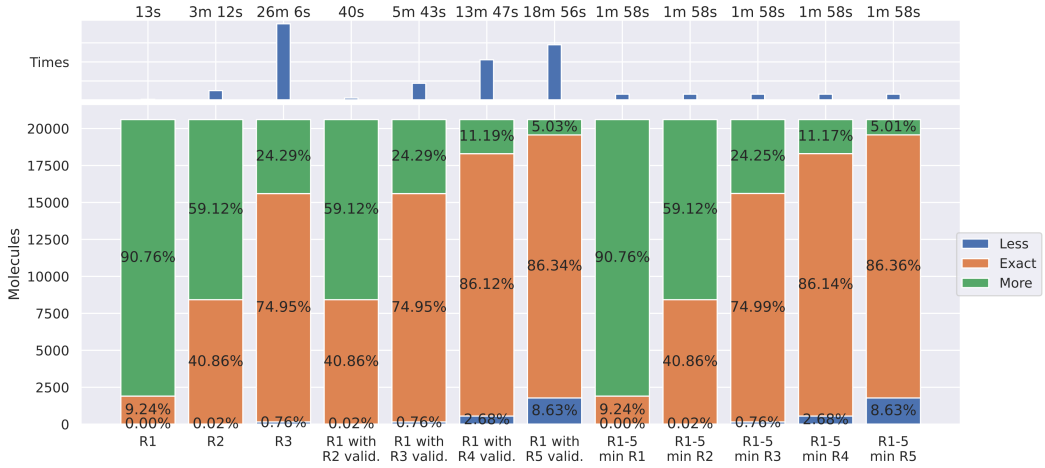


Figure A.1: Visualization of the full evaluation of the different location strategies for the grouping dataset

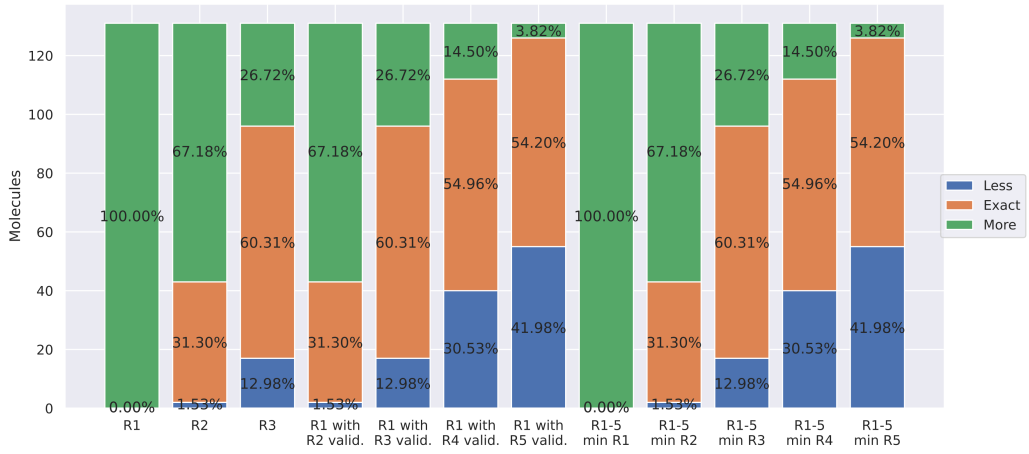


Figure A.2: Visualization of the full evaluation of the different location strategies for the validation dataset

A.3 ChemAxon Marvin SMARTS tree radii 2-5

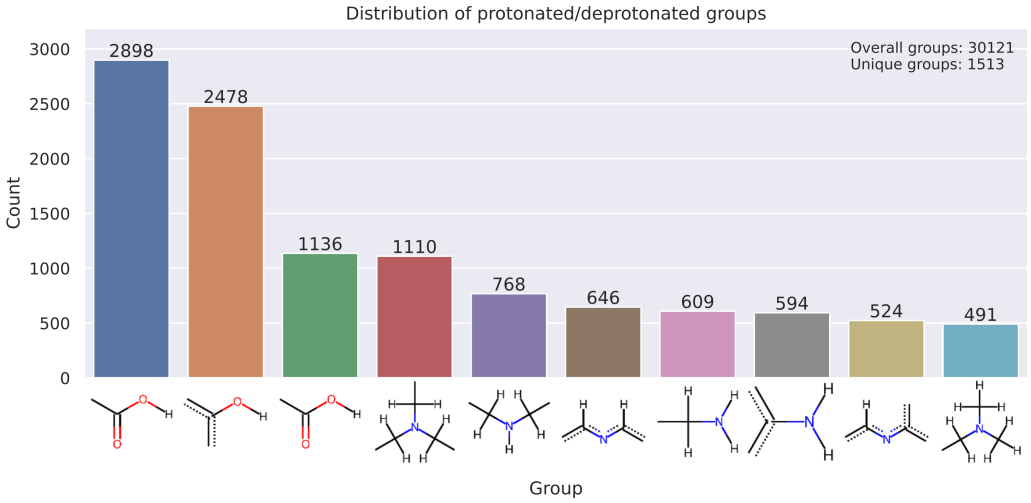


Figure A.3: Distribution of the top 10 titratable groups with radius 2 calculated with ChemAxon Marvin[8]

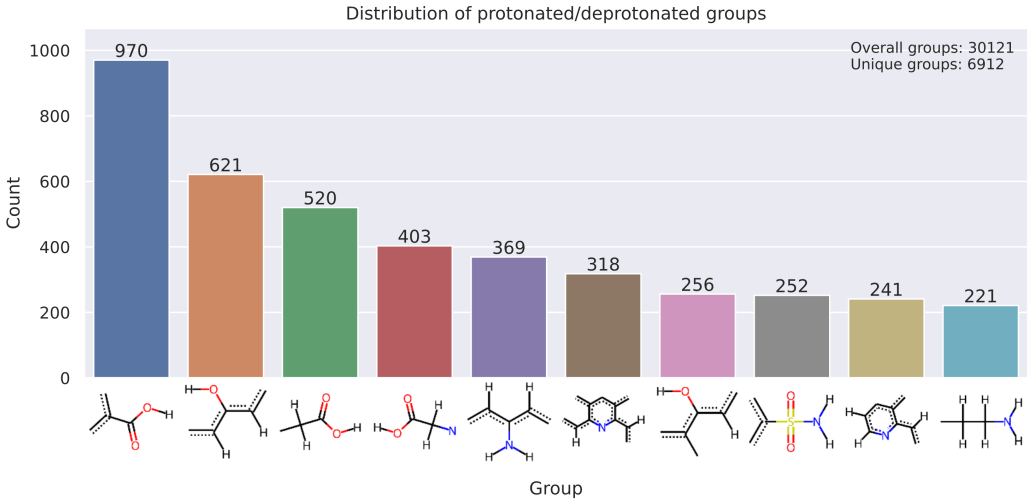


Figure A.4: Distribution of the top 10 titratable groups with radius 3 calculated with ChemAxon Marvin[8]

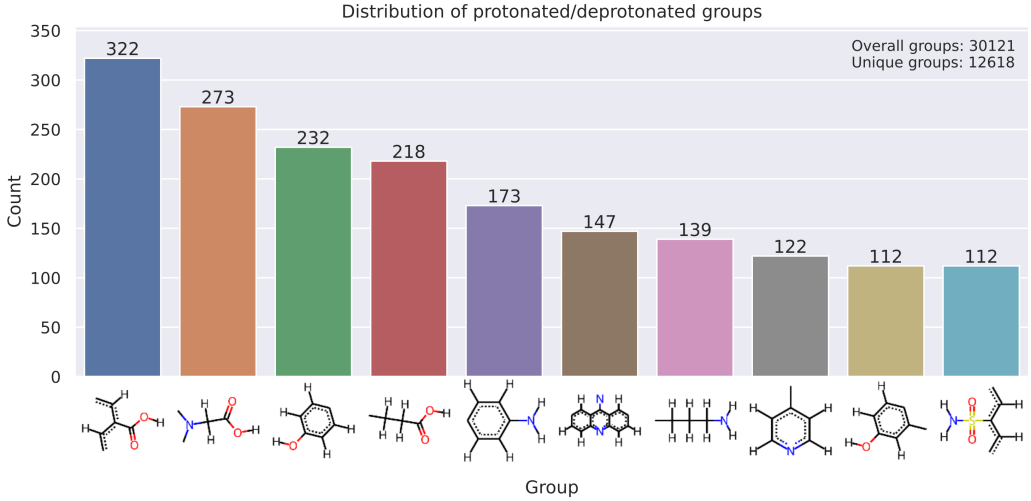


Figure A.5: Distribution of the top 10 titratable groups with radius 4 calculated with ChemAxon Marvin[8]

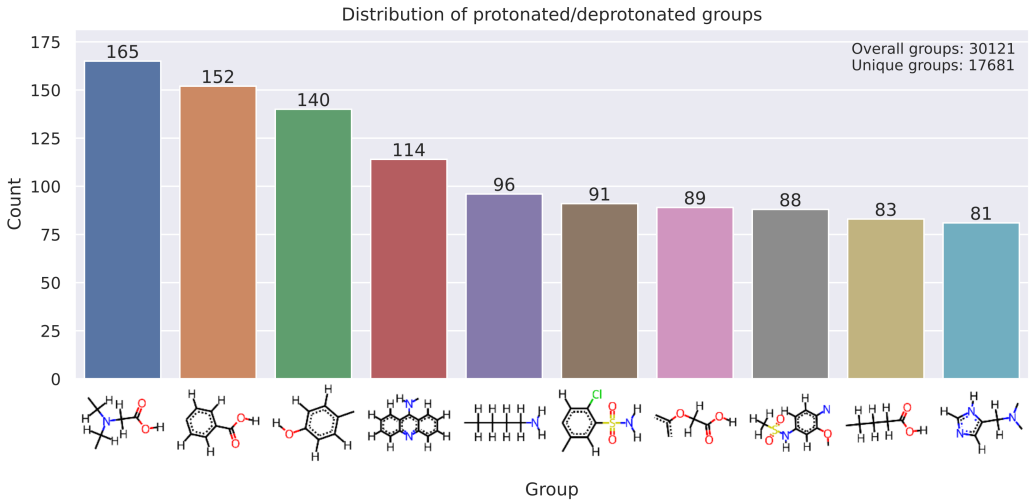


Figure A.6: Distribution of the top 10 titratable groups with radius 5 calculated with ChemAxon Marvin[8]

A.4 Dimorphite-DL SMARTS tree radii 2-5

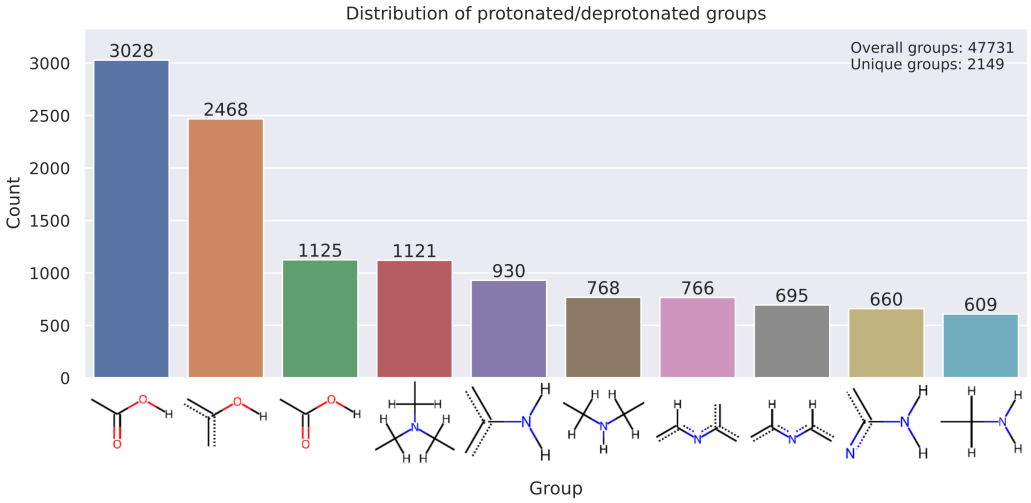


Figure A.7: Distribution of titratable groups with radius 2 calculated with Dimorphite-DL[111]

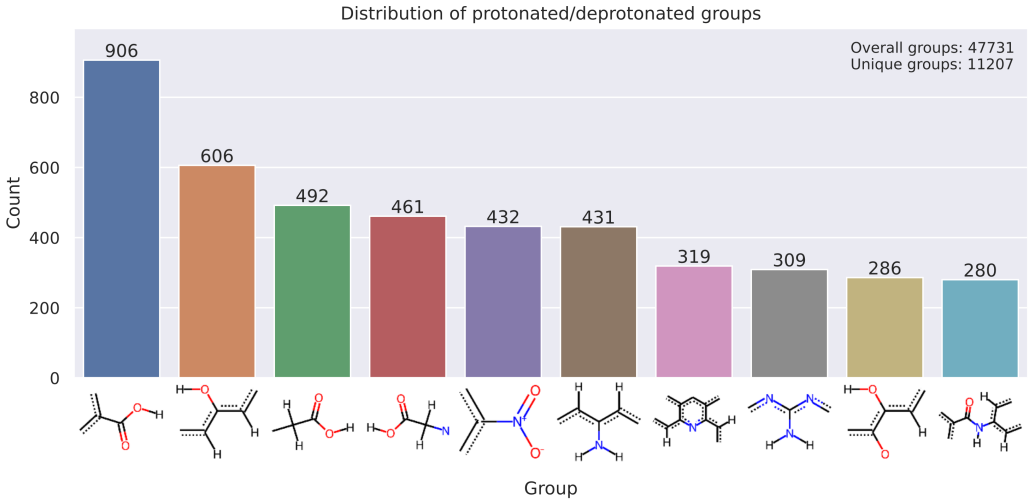


Figure A.8: Distribution of titratable groups with radius 3 calculated with Dimorphite-DL[111]

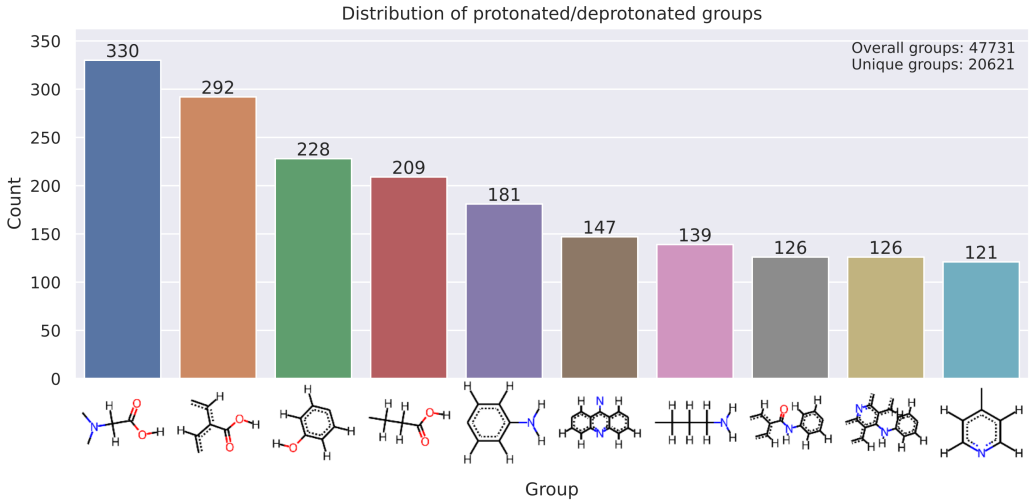


Figure A.9: Distribution of titratable groups with radius 4 calculated with Dimorphite-DL[111]

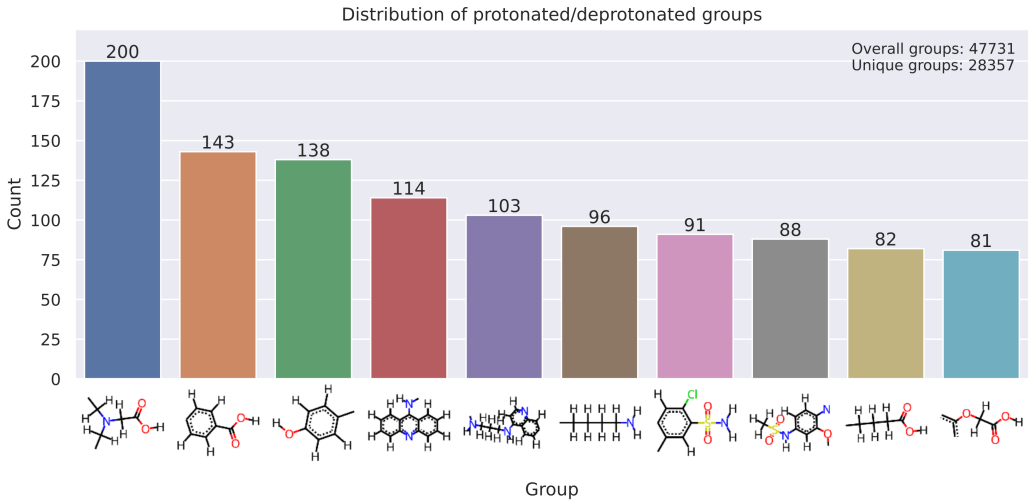


Figure A.10: Distribution of titratable groups with radius 5 calculated with Dimorphite-DL[111]

A.5 Top 15 strongest outliers from value assignment

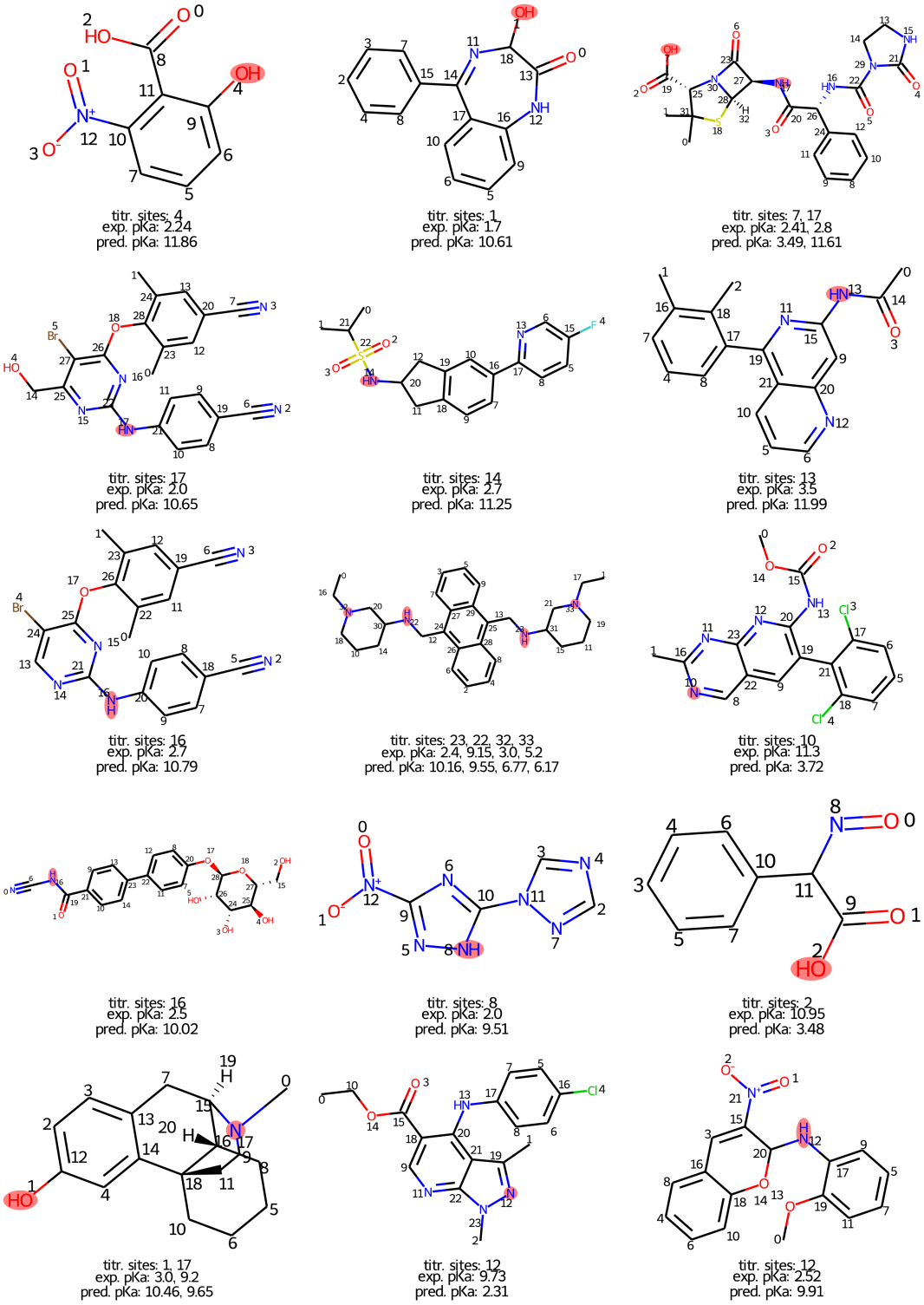


Figure A.11: Top 15 strongest outliers after value assignment (cf. chapter 5.3.4)

A.6 Instructions for pK_a model usage

The attached folder contains the code for the Graph-based Multiprotic pK_a Predictor (GMPP) tool as well as all corresponding datasets and result files belonging to this PhD thesis. Used datasets and files containing data that can not be published due to licensing restrictions or confidential agreements are not included in this repository. These files are replaced with an empty file containing the suffix ".missing".

A.6.1 Folder overview

- **datasets/**: Contains all datasets used for training and testing the models.
- **Machine-learning-meets-pKa/**: Contains the code and files published with the paper *Machine Learning Meets pK_a*[9].
- **mpp_results/**: Contains all results of the Multiprotic pK_a Processor (MPP) tool used to generate the training and test datasets analyzed in chapter 5.3 and used for hyperparameter search and training in chapters 6.4.1 and 6.4.2.
- **mpp_results_sampl/**: Contains all results of the MPP tool used to generate the training and test datasets used for training in chapter 6.4.3.
- **multiprotic-modelling/**: Contains the code for the GMPP tool.
- **multiprotic-pka-processing/**: Contains the code for the MPP tool.
- **dissertation_baltruschat_2024-06-23.pdf**: PDF file of this PhD thesis.
- **Dockerfile**: Dockerfile for building a docker image to use for prediction with the model evaluated in chapter 6.4.3.
- **environment_linux64.yml**: Conda environment file for creating a Conda environment to use for prediction.
- **model_config.json**: Configuration file for the GMPP tool containing all settings that where used for generating the model evaluated in chapter 6.4.3.
- **sma_tree_r1-5.pkl**: Pickle file containing the SMARTS trees used for locating titration sites.
- **state_dict_MultiproticNet_V27.pt**: PyTorch state dictionary file containing the model weights and further model related settings of the model evaluated in chapter 6.4.3.

- **state_dict_MultiproticNet_V28.pt**: PyTorch state dictionary file containing the model weights and further model related settings of the model evaluated in chapter 6.4.2.

A.6.2 Installation (Linux x86_64 only)

This tool is currently only available for Linux x86_64 systems since the environment file and the docker image are only available for this platform. The tool can be installed using either Conda or Docker. The Conda installation is recommended since it is easier to use, does not require root privileges and makes it possible to use *OpenEye QUACPAC/Tautomers*[96] if a license is available.

Using Mamba

A conda-based installation, e.g. Anaconda, Miniconda, Miniforge or Mambaforge, must be available on your system and `mamba` must be installed in the base environment. If no existing conda installation is available, visit `https://github.com/conda-forge/miniforge#mambaforge` for more information.

To create the conda environment, run the following command:

```
mamba env create -f environment_linux64.yml
```

Now you can install the two packages contained in this folder:

```
mamba activate GMPP
pip install ./multiprotic-pka-processing ./multiprotic-modelling
```

Using Docker

Docker must be installed and running on your system. See `https://www.docker.com/` for more information. To build the docker image, run the following command:

```
docker build --tag gmpp .
```

Important note: If you use the docker way of installation, you can not use the *OpenEye QUACPAC/Tautomers*[96] for tautomer and charge standardization since it is not installed in the docker image. In that case you must change the config value `use_openeye` to `false` in the `model_config.json` file **before** building the docker image. This will use *MolVS* integrated in *RDKit*[77] for tautomer and charge standardization. This has an impact on the prediction quality.

A.6.3 Usage

Using Mamba

Assuming you are in this folder you can make a prediction for an structure-data file (SDF) using the following command:

```
predict.py model_config.json \
    datasets/settimo_et_al.sdf prediction.sdf
```

Using Docker

To run the docker image, run the following command:

```
docker run --rm -v $(pwd):/data gmpp \
    /data/datasets/settimo_et_al.sdf /data/prediction.sdf
```

Abbreviations

ACS American Chemical Society

Adam Adaptive Moment Estimation

API Application Programming Interface

BBB blood-brain barrier

BSD Berkeley Software Distribution

CDK Chemistry Development Kit

CPU central processing unit

CSD Cambridge Structural Database

CSV comma-separated values

CXM ChemAxon Marvin[8]

DDL Dimorphite-DL[111]

DFT Density Functional Theory

ETKDG Experimental-Torsion Basic Knowledge Distance Geometry

GCN Graph Convolutional Neural Network

GMPP Graph-based Multiprotic pK_{a} Predictor

GNN Graph Neural Network

GPU graphics processing unit

HDF5 Hierarchical Data Format 5

ITC Isothermal Titration Calorimetry

JSON JavaScript Object Notation

LFER Linear Free-Energy Relationship

LiDO3 Linux Cluster TU Dortmund 3

LReLU leaky rectified linear unit (ReLU)

MAE mean absolute error

MD molecular dynamics

MIT Massachusetts Institute of Technology

MLP Multi-layer Perceptron

MMFF94 Merck Molecular Force Field 94

MPNN Message Passing Neural Network

MPP Multiprotic pK_{a} Processor

MSE mean square error

NMR Nuclear Magnetic Resonance

PCA principle component analysis

PSF Python Software Foundation

PyG PyTorch Geometric[11]

QSPR Quantitative Structure-Property Relationship

$\mathbf{R^2}$ coefficient of determination

ReLU rectified linear unit

RF Random Forest

RMSE root-mean-square error

RReLU randomized leaky ReLU

SAMPL Statistical Assessment of the Modeling of Proteins and Ligands

SDF structure-data file

SGD Stochastic Gradient Descent

SMARTS SMILES Arbitrary Target Specification

SMILES Simplified Molecular-Input Line-Entry System

SQL Structured Query Language

SVC Support Vector Classification

SVM Support Vector Machine

SVR Support Vector Regression

t-SNE t-distributed stochastic neighbour embedding

tanh hyperbolic tangent

TS titration site

XGB XGradientBoost

List of Figures

List of Tables

References

[1] David T. Manallack. "The pKa Distribution of Drugs: Application to Drug Discovery". In: *Perspectives in Medicinal Chemistry* 1 (Jan. 2007). DOI: 10.1177/1177391X0700100003.

[2] David T. Manallack, Elizabeth Yuriev, and David K. Chalmers. "The influence and manipulation of acid/base properties in drug discovery". In: *Drug Discovery Today: Technologies* 27 (2018), pp. 41–47. ISSN: 1740-6749. DOI: 10.1016/j.ddtec.2018.04.003.

[3] David T. Manallack et al. "The significance of acid/base properties in drug discovery". In: *Chem. Soc. Rev.* 42 (2 2013), pp. 485–496. DOI: 10.1039/C2CS35348B.

[4] Paul S. Charifson and W. Patrick Walters. "Acidic and Basic Drugs in Medicinal Chemistry: A Perspective". In: *Journal of Medicinal Chemistry* 57.23 (Dec. 2014), pp. 9701–9717. DOI: 10.1021/jm501000a.

[5] Nicholas A. Meanwell. "Improving Drug Candidates by Design: A Focus on Physicochemical Properties As a Means of Improving Compound Disposition and Safety". In: *Chemical Research in Toxicology* 24.9 (2011), pp. 1420–1456. DOI: 10.1021/tx200211v.

[6] M. Paul Gleeson. "Generation of a Set of Simple, Interpretable ADMET Rules of Thumb". In: *Journal of Medicinal Chemistry* 51.4 (2008), pp. 817–834. DOI: 10.1021/jm701122q.

[7] Paul D. Leeson, Stephen A. St-Gallay, and Mark C. Wenlock. "Impact of ion class and time on oral drug molecular properties". In: *Med. Chem. Commun.* 2 (2 2011), pp. 91–105. DOI: 10.1039/C0MD00157K.

[8] ChemAxon. *Marvin*. Version 21.3.0. 2021. URL: https://chemaxon.com.

[9] Marcel Baltruschat and Paul Czodrowski. "Machine learning meets pKa". In: *F1000Research* 9 (Feb. 2020), p. 113. ISSN: 2046-1402. DOI: 10.12688/f1000research.22090.1.

[10] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: (2019). DOI: 10.48550/ARXIV.1912.01703.

[11] Matthias Fey and Jan Eric Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *arXiv* (Mar. 2019). DOI: 10.48550/ARXIV.1903.02428.

[12] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[13] Andrew Chen et al. "Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle". In: *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*. DEEM'20. Portland, OR, USA: Association for Computing Machinery, 2020. DOI: 10.1145/3399579.3399867.

[14] Luca Settimo, Krista Bellman, and Ronald M. A. Knegtel. "Comparison of the Accuracy of Experimental and Predicted pKa Values of Basic and Acidic Compounds". In: *Pharmaceutical Research* 31.4 (Apr. 2014), pp. 1082–1095. DOI: 10.1007/s11095-013-1232-z.

[15] Richard A. Lewis and Stephane Rodde. *Novartis Pharma AG*. Basel, Switzerland.

[16] Mehtap Isik et al. *MobleyLab/SAMPL6: SAMPL6 Part II - Release the evaluation results of log *P* predictions*. Version v1.16. Apr. 2019. DOI: 10.5281/zenodo.2651393.

[17] Teresa Danielle Bergazin et al. *samplchallenges/SAMPL7: Correct two submissions, update analysis, and finalize logD analysis*. Version 1.0. Apr. 2021. DOI: 10.5281/zenodo.4706021.

[18] Kenneth W. Whitten. *General Chemistry*. 4th ed. Fort Worth, Tex.: Saunders College Pub., 1992. ISBN: 9780030751561.

[19] John William Hill and Ralph H. Petrucci. *General Chemistry*. 3rd ed. New Jersey: Upper Saddle River, N.J., 2002. ISBN: 9780130334459.

[20] Edithe Selwa et al. "SAMPL6: calculation of macroscopic pK a values from ab initio quantum mechanical free energies". In: *Journal of Computer-Aided Molecular Design* 32.10 (Oct. 2018), pp. 1203–1216. ISSN: 15734951. DOI: 10.1007/s10822-018-0138-6.

[21] Henry N. Po and N. M. Senozan. "The Henderson-Hasselbalch Equation: Its History and Limitations". In: *Journal of Chemical Education* 78.11 (2001), p. 1499. DOI: 10.1021/ed078p1499.

[22] Art D. Bochevarov et al. "Multiconformation, Density Functional Theory-Based pKa Prediction in Application to Large, Flexible Organic Molecules with Diverse Functional Groups". In: *Journal of Chemical Theory and Computation* 12.12 (2016), pp. 6001–6019. DOI: 10.1021/acs.jctc.6b00805.

[23] Mehtap Işık et al. “Overview of the SAMPL6 pKa challenge: evaluating small molecule microscopic and macroscopic pKa predictions”. In: *Journal of Computer-Aided Molecular Design* (Jan. 2021), pp. 1–36. ISSN: 15734951. DOI: 10.1007/s10822-020-00362-6.

[24] Dallas L. Rabenstein and Thomas L. Sayer. “Determination of microscopic acid dissociation constants by nuclear magnetic resonance spectrometry”. In: *Analytical Chemistry* 48.8 (May 2002), pp. 1141–1146. DOI: 10.1021/ac50002a019.

[25] M. R. Gunner et al. “Standard state free energies, not pKas, are ideal for describing small molecule protonation and tautomeric states”. In: *Journal of Computer-Aided Molecular Design* (Feb. 2020), pp. 1–13. ISSN: 15734951. DOI: 10.1007/s10822-020-00280-7.

[26] Jetse Reijenga et al. “Development of methods for the determination of pKa values”. In: *Analytical Chemistry Insights* 8 (2013). DOI: 10.4137/ACI.S12304.

[27] Frans M. Everaerts, Jo L. Beckers, and Theo P. E. M. Verheggen. *Isotachophoresis: Theory, Instrumentation and Applications*. Vol. 6. Elsevier Scientific Publishing Co., Jan. 1976, pp. 1–418. ISBN: 9780080858067.

[28] Lisa Samuelsen et al. “Buffer solutions in drug formulation and processing: How pKa values depend on temperature, pressure and ionic strength”. In: *International Journal of Pharmaceutics* 560 (Apr. 2019), pp. 357–364. ISSN: 18733476. DOI: 10.1016/j.ijpharm.2019.02.019.

[29] C. D. Kennedy. “Ionic strength and the dissociation of acids”. In: *Biochemical Education* 18 (1 Jan. 1990), pp. 35–40. ISSN: 0307-4412. DOI: 10.1016/0307-4412(90)90017-I.

[30] Mandeep Dalal. *A Textbook of Physical Chemistry – Volume 1*. 1st ed. Dalal Institute, 2018. ISBN: 9788193872017.

[31] A. Baird Hastings and Julius Sendroy. “THE EFFECT OF VARIATION IN IONIC STRENGTH ON THE APPARENT FIRST AND SECOND DISSOCIATION CONSTANTS OF CARBONIC ACID”. In: *Journal of Biological Chemistry* 65 (2 Sept. 1925), pp. 445–455. ISSN: 0021-9258. DOI: 10.1016/S0021-9258(18)84852-9.

[32] Robert L. Jones and W. David Wilson. “Effect of ionic strength on the pKa of ligands bound to DNA”. In: *Biopolymers* 20 (1 Jan. 1981), pp. 141–154. ISSN: 0006-3525. DOI: 10.1002/bip.1981.360200110.

[33] Young Kee Kang, George Némethy, and Harold A. Scheraga. “Free energies of hydration of solute molecules. 2. Application of the hydration shell model to nonionic organic molecules”. In: *Journal of Physical Chemistry* 91.15 (1987), pp. 4109–4117. DOI: 10.1021/j100299a033.

[34] Emanuele Rossini, Art D. Bochevarov, and Ernst Walter Knapp. "Empirical Conversion of pKa Values between Different Solvents and Interpretation of the Parameters: Application to Water, Acetonitrile, Dimethyl Sulfoxide, and Methanol". In: *ACS Omega* 3.2 (Feb. 2018), pp. 1653–1662. DOI: 10.1021/acsomega.7b01895.

[35] Stephen T. Heller and Todd P. Silverstein. "pKa values in the undergraduate curriculum: introducing pKa values measured in DMSO to illustrate solvent effects". In: *ChemTexts* 6 (2 2020), p. 15. ISSN: 2199-3793. DOI: 10.1007/s40828-020-00112-z.

[36] Walter C. Holmes and Edward F. Snyder. "Spectrophotometric Determination of Hydrogen-Ion Concentrations and of the Apparent Dissociation Constants of Indicators IV. 1-Naphthol-2-Sodium Sulfonate Indophenol". In: *Journal of the American Chemical Society* 47 (8 May 1925), pp. 2232–2236. DOI: 10.1021/ja01685a023.

[37] ChemAxon Ltd. *pKa calculation — Chemaxon Docs*. Accessed: 11.06.2023. 2023. URL: https://docs.chemaxon.com/display/docs/pka-calculation.md.

[38] József Szegezdi and Ferenc Csizmadia. "Prediction of dissociation constant using microconstants". In: *American Chemical Society National Meeting*. Accessed: 11.06.2023. ChemAxon Ltd, 2004. URL: https://docs.chemaxon.com/display/docs/attachments/attachments_1814016_1_Prediction_of_dissociation_constant_using_microconstants.pdf.

[39] József Szegezdi and Ferenc Csizmadia. "A method for calculating the pKa values of small and large molecules". In: *American Chemical Society Spring Meeting*. Accessed: 11.06.2023. ChemAxon Ltd, 2007. URL: https://docs.chemaxon.com/display/docs/attachments/attachments_1814017_1_Calculating_pKa_values_of_small_and_large_molecules.pdf.

[40] Johann Gasteiger and Mario Marsili. "Iterative partial equalization of orbital electronegativity-a rapid access to atomic charges". In: *Tetrahedron* 36.22 (1980), pp. 3219–3228. DOI: 10.1016/0040-4020(80)80168-2.

[41] Kenneth J. Miller and John A. Savchik. "A New Empirical Method to Calculate Average Molecular Polarizabilities". In: *Journal of the American Chemical Society* 101.24 (1979), pp. 7206–7213. DOI: 10.1021/ja00518a014.

[42] Ferenc Csizmadia et al. "Prediction of distribution coefficient from structure. 1. Estimation method". In: *Journal of Pharmaceutical Sciences* 86.7 (1997), pp. 865–871. DOI: 10.1021/js960177k.

[43] Chenzhong Liao and Marc C. Nicklaus. "Comparison of nine programs predicting pKa values of pharmaceutical substances". In: *Journal of Chemical*

Information and Modeling 49.12 (Dec. 2009), pp. 2801–2812. ISSN: 15499596. DOI: 10.1021/ci900289x.

[44] John Manchester et al. "Evaluation of pKa Estimation Methods on 211 Drug-like Compounds". In: *Journal of Chemical Information and Modeling* 50.4 (Mar. 2010), pp. 565–571. DOI: 10.1021/ci100019p.

[45] György T. Balogh, Ákos Tarcsay, and György M. Keserű. "Comparative evaluation of pKa prediction tools on a drug discovery dataset". In: *Journal of Pharmaceutical and Biomedical Analysis* 67-68 (Aug. 2012), pp. 63–70. DOI: 10.1016/J.JPBA.2012.04.021.

[46] José L. Castro et al. "Enhancement of Oral Absorption in Selective 5-HT1D Receptor Agonists: Fluorinated 3-[3-(Piperidin-1-yl)propyl]indoles". In: *Journal of Medicinal Chemistry* 41.15 (1998), pp. 2667–2670. DOI: 10.1021/jm980204e.

[47] Yi Fan et al. "Insights for Predicting Blood-Brain Barrier Penetration of CNS Targeted Molecules Using QSPR Approaches". In: *Journal of Chemical Information and Modeling* 50.6 (2010), pp. 1123–1133. DOI: 10.1021/ci900384c.

[48] Jialu Wu et al. "Machine learning methods for pKa prediction of small molecules: Advances and challenges". In: *Drug Discovery Today* 27.12 (2022), p. 103372. ISSN: 1359-6446. DOI: 10.1016/j.drudis.2022.103372.

[49] Ibon Alkorta and Paul L. A. Popelier. "Linear Free-Energy Relationships between a Single Gas-Phase Ab Initio Equilibrium Bond Length and Experimental pKa Values in Aqueous Solution". In: *ChemPhysChem* 16.2 (2015), pp. 465–469. DOI: 10.1002/cphc.201402711.

[50] Junming Ho. "Predicting pKa in Implicit Solvents: Current Status and Future Directions". In: *Australian Journal of Chemistry* 67.10 (2014), pp. 1441–1460.

[51] Paul G. Seybold and George C. Shields. "Computational estimation of pKa values". In: *WIREs Computational Molecular Science* 5.3 (2015), pp. 290–297. DOI: 10.1002/wcms.1218.

[52] Art D. Bochevarov et al. "Jaguar: A high-performance quantum chemistry software program with strengths in life and materials sciences". In: *International Journal of Quantum Chemistry* 113.18 (2013), pp. 2110–2142. DOI: 10.1002/qua.24481.

[53] A. P. Harding, D. C. Wedge, and P. L. A. Popelier. "pKa Prediction from "Quantum Chemical Topology" Descriptors". In: *Journal of Chemical Information and Modeling* 49.8 (2009), pp. 1914–1924. DOI: 10.1021/ci900172h.

[54] J. Michael Word and Anthony Nicholls. "Application of the Gaussian dielectric boundary in Zap to the prediction of protein pKa values". In: *Proteins: Structure, Function, and Bioinformatics* 79.12 (2011), pp. 3400–3409. DOI: 10.1002/prot.23079.

[55] Donald Bashford and Martin Karplus. "pKa's of ionizable groups in proteins: atomic detail from a continuum electrostatic model". In: *Biochemistry* 29.44 (1990), pp. 10219–10225. DOI: 10.1021/bi00496a010.

[56] Abdulaziz H. Alkhzem, Timothy J. Woodman, and Ian S. Blagbrough. "Individual pKa Values of Tobramycin, Kanamycin B, Amikacin, Sisomicin, and Netilmicin Determined by Multinuclear NMR Spectroscopy". In: *ACS Omega* 5.33 (Aug. 2020), pp. 21094–21103. ISSN: 2470-1343. DOI: 10.1021/acsomega.0c02744.

[57] G. Matthias Ullmann. "Relations between Protonation Constants and Titration Curves in Polyprotic Acids: A Critical View". In: *The Journal of Physical Chemistry B* 107.5 (2003), pp. 1263–1271. DOI: 10.1021/jp026454v.

[58] Alexey Onufriev, David A. Case, and G. Matthias Ullmann. "A Novel View of pH Titration in Biomolecules". In: *Biochemistry* 40.12 (2001), pp. 3413–3419. DOI: 10.1021/bi002740q.

[59] Plamen Dobrev et al. "Probing the Accuracy of Explicit Solvent Constant pH Molecular Dynamics Simulations for Peptides". In: *Journal of Chemical Theory and Computation* 16.4 (2020), pp. 2561–2569. DOI: 10.1021/acs.jctc.9b01232.

[60] Noora Aho et al. "Scalable Constant pH Molecular Dynamics in GROMACS". In: *Journal of Chemical Theory and Computation* 18.10 (2022), pp. 6148–6160. DOI: 10.1021/acs.jctc.2c00516.

[61] Pavel Buslaev et al. "Best Practices in Constant pH MD Simulations: Accuracy and Sampling". In: *Journal of Chemical Theory and Computation* 18.10 (2022), pp. 6134–6147. DOI: 10.1021/acs.jctc.2c00517.

[62] Scott Hartshorn. *Machine Learning With Random Forests And Decision Trees*. Aug. 2016.

[63] Leo Breiman. "Random Forests". In: *Machine Learning* 45 (1 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[64] Chris Smith and Mark Koning. *Decision Trees and Random Forests: A Visual Introduction For Beginners*. Oct. 2017. ISBN: 978-1549893759.

[65] Alex J Smola and Bernhard Schölkopf. "A tutorial on support vector regression". In: *Statistics and Computing* 14 (3 2004), pp. 199–222. ISSN: 1573-1375. DOI: 10.1023/B:STCO.0000035301.49549.88.

[66] Andreas Christmann and Ingo Steinwart. *Support Vector Machines*. 1st ed. Springer New York, 2008. ISBN: 978-0-387-77241-7. DOI: 10.1007/978-0-387-77242-4.

[67] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[68] Pragati Baheti. *The Essential Guide to Neural Network Architectures*. Accessed: 17.11.2022. Oct. 2022. URL: https://www.v7labs.com/blog/neural-network-architectures-guide.

[69] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, Nov. 2016.

[70] Si Zhang et al. "Graph convolutional networks: a comprehensive review". In: *Computational Social Networks* 6 (1 2019), p. 11. ISSN: 2197-4314. DOI: 10.1186/s40649-019-0069-y.

[71] Benjamin Sanchez-Lengeling et al. "A Gentle Introduction to Graph Neural Networks". In: *Distill* (2021). DOI: 10.23915/distill.00033.

[72] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *arXiv* (2017). DOI: 10.48550/ARXIV.1704.01212.

[73] Nitika Verma, Edmond Boyer, and Jakob Verbeek. "FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis". In: *arXiv* (2017). DOI: 10.48550/ARXIV.1706.05206.

[74] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[75] TIOBE Software BV. *TIOBE Index*. Accessed: 21.11.2022. Nov. 2022. URL: https://www.tiobe.com/tiobe-index/.

[76] Python Software Foundation. *Applications for Python*. Accessed: 21.11.2022. 2022. URL: https://www.python.org/about/apps/.

[77] *RDKit: Open-source cheminformatics*. URL: https://www.rdkit.org.

[78] *GitHub: rdkit/rdkit*. Accessed: 21.11.2022. 2022. URL: https://github.com/rdkit/rdkit.

[79] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[80] The pandas development team. *pandas-dev/pandas: Pandas*. Version v1.5.1. Oct. 2022. DOI: 10.5281/zenodo.7223478.

[81] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[82] The Linux Foundation. *Features — PyTorch*. Accessed: 22.11.2022. 2022. URL: https://pytorch.org/features/.

[83] Horace He. "The State of Machine Learning Frameworks in 2019". In: *The Gradient* (Oct. 2019). URL: https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/.

[84] Francesca Milletti et al. “New and Original pKa Prediction Method Using Grid Molecular Interaction Fields”. In: *Journal of Chemical Information and Modeling* 47.6 (Oct. 2007), pp. 2172–2181. DOI: 10.1021/ci700018y.

[85] Advanced Chemistry Development, Inc., Toronto, On, Canada. *ACD/Percepta*. 2021. URL: www.acdlabs.com.

[86] John C. Shelley et al. “Epik: a software program for pKa prediction and protonation state generation for drug-like molecules”. In: *Journal of Computer-Aided Molecular Design* 21 (12 Dec. 2007), pp. 681–691. DOI: 10.1007/s10822-007-9133-z.

[87] Robert Fraczkiewicz et al. “Best of Both Worlds: Combining Pharma Data and State of the Art Modeling Technology To Improve in Silico pKa Prediction”. In: *Journal of Chemical Information and Modeling* 55.2 (Dec. 2014), pp. 389–397. DOI: 10.1021/ci500585w.

[88] Rafał Roszak et al. “Rapid and Accurate Prediction of pKa Values of C–H Acids Using Graph Convolutional Neural Networks”. In: *Journal of the American Chemical Society* 141.43 (Oct. 2019), pp. 17142–17149. DOI: 10.1021/jacs.9b05895.

[89] Kamel Mansouri et al. “Open-source QSAR models for pKa prediction using multiple machine learning approaches”. In: *Journal of Cheminformatics* 11.1 (Dec. 2019), p. 60. ISSN: 1758-2946. DOI: 10.1186/s13321-019-0384-1.

[90] Thomas Sander et al. “DataWarrior: An Open-Source Program For Chemistry Aware Data Visualization And Analysis”. In: *Journal of Chemical Information and Modeling* 55.2 (Feb. 2015), pp. 460–473. DOI: 10.1021/ci500588j.

[91] Marcel Baltruschat, czodrowskilab, and Paul Czodrowski. *czodrowskilab/Machine-learning-meets-pKa article*. Version article. Feb. 2020. DOI: 10.5281/zenodo.3662245.

[92] Anna Gaulton et al. “The ChEMBL database in 2017”. In: *Nucleic Acids Research* 45.D1 (Nov. 2016), pp. D945–D954. ISSN: 0305-1048. DOI: 10.1093/nar/gkw1074.

[93] Alex Avdeef. *Absorption and Drug Development: Solubility, Permeability, and Charge State*. Aug. 2003. ISBN: 9780471423652. DOI: 10.1002/047145026X.

[94] Feng Luan et al. “Prediction of pKa for Neutral and Basic Drugs Based on Radial Basis Function Neural Networks and the Heuristic Method”. In: *Pharmaceutical Research* 22 (9 2005), pp. 1454–1460. ISSN: 1573-904X. DOI: 10.1007/s11095-005-6246-8.

[95] Martin Morgenthaler et al. *Predicting and tuning physicochemical properties in lead optimization: Amine basicities*. Aug. 2007. DOI: 10.1002/cmdc.200700059.

[96] OpenEye Scientific Software, Santa Fe, NM. *QUACPAC*. Version 2.1.1.2. 2021. URL: http://www.eyesopen.com.

[97] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785.

[98] Christophe Dardonville. "Automated techniques in pKa determination: Low, medium and high-throughput screening methods". In: *Drug Discovery Today: Technologies* 27 (July 2018), pp. 49–58. ISSN: 17406749. DOI: 10.1016/j.ddtec.2018.04.001.

[99] David Mendez et al. "ChEMBL: towards direct deposition of bioassay data". In: *Nucleic Acids Research* 47.D1 (2018), pp. D930–D940. ISSN: 0305-1048. DOI: 10.1093/nar/gky1075.

[100] David L. Mobley and amezcum1. *samplchallenges/SAMPL8: SAMPL8 CB8 "drugs of abuse" challenge inputs*. Version 0.1.0. Sept. 2020. DOI: 10.5281/zenodo.4029560.

[101] Julien Hazemann. *Idorsia Pharmaceuticals Ltd*. Basel, Switzerland.

[102] Christian Kramer. *Roche Pharma AG*. Basel, Switzerland.

[103] OpenEye Scientific Software, Santa Fe, NM. *Aqueous pKa Database*. Version 1.13. 2017. URL: http://www.eyesopen.com.

[104] Cecilia B. Castells et al. "Effect of temperature on pH measurements and acid-base equilibria in methanol-water mixtures". In: *Journal of Chromatography A* 1002.1-2 (June 2003), pp. 41–53. ISSN: 00219673. DOI: 10.1016/S0021-9673(03)00644-7.

[105] Peter Hunt et al. "Predicting pKa Using a Combination of Semi-Empirical Quantum Mechanics and Radial Basis Function Methods". In: *Journal of Chemical Information and Modeling* 60.6 (June 2020), pp. 2989–2997. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.0c00105.

[106] Javad Noroozi and William R. Smith. "Prediction of Alkanolamine pKa Values by Combined Molecular Dynamics Free Energy Simulations and ab Initio Calculations". In: *Journal of Chemical & Engineering Data* 65.3 (Mar. 2020), pp. 1358–1368. DOI: 10.1021/acs.jced.9b00927.

[107] Adam C. Lee, Jing-yu Yu, and Gordon M. Crippen. "pKa Prediction of Monoprotic Small Molecules the SMARTS Way". In: *Journal of Chemical Information and Modeling* 48.10 (Oct. 2008), pp. 2042–2053. DOI: 10.1021/ci8001815.

[108] Jasna J. Klicić et al. "Accurate Prediction of Acidity Constants in Aqueous Solution via Density Functional Theory and Self-Consistent Reaction Field

Methods". In: *Journal of Physical Chemistry* 106.7 (2002), pp. 1327–1335. DOI: 10.1021/JP012533F.

[109] Haiying Yu et al. "Comparative Analysis of QSAR Models for Predicting pKa of Organic Oxygen Acids and Nitrogen Bases from Molecular Structure". In: *Journal of Chemical Information and Modeling* 50.11 (Oct. 2010), pp. 1949–1960. DOI: 10.1021/ci100306k.

[110] Greg Landrum et al. *rdkit/rdkit: 2021_09_2 (Q3 2021) Release*. Version Release_2021_09_2. Oct. 2021. DOI: 10.5281/zenodo.5589557.

[111] Patrick J. Ropp et al. "Dimorphite-DL: an open-source program for enumerating the ionization states of drug-like small molecules". In: *Journal of Cheminformatics* 11.1 (Feb. 2019), p. 14. ISSN: 1758-2946. DOI: 10.1186/s13321-019-0336-9.

[112] Pengju Ji. *iBonD 2.0-the most comprehensive pKa and BDE database so far*. July 2016. DOI: 10.13140/RG.2.1.4632.9841.

[113] Ehrhardt Proksch. "pH in nature, humans and skin". In: *The Journal of Dermatology* 45.9 (2018), pp. 1044–1052. DOI: 10.1111/1346-8138.14489.

[114] Greg Landrum et al. *rdkit/rdkit: 2022_03_1 (Q1 2022) Release*. Version Release_2022_03_1. Mar. 2022. DOI: 10.5281/zenodo.6388425.

[115] Jeff Reback et al. *pandas-dev/pandas: Pandas 1.4.2*. Version v1.4.2. Apr. 2022. DOI: 10.5281/zenodo.6408044.

[116] Sereina Riniker and Gregory A. Landrum. "Better Informed Distance Geometry: Using What We Know To Improve Conformation Generation". In: *Journal of Chemical Information and Modeling* 55.12 (2015), pp. 2562–2574. DOI: 10.1021/acs.jcim.5b00654.

[117] Jeffrey M. Blaney and J. Scott Dixon. *Distance Geometry in Molecular Modeling*. John Wiley & Sons, Ltd, 1994, pp. 299–335. ISBN: 9780470125823. DOI: 10.1002/9780470125823.ch6.

[118] Timothy F. Havel. "Distance Geometry: Theory, Algorithms, and Chemical Applications". In: *Encyclopedia of Computational Chemistry* (2002). DOI: 10.1002/0470845015.cda018.

[119] "Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94". In: *Journal of Computational Chemistry* 17.5-6 (1996), pp. 490–519. DOI: 10.1002/(SICI)1096-987X(199604)17:5/6<490::AID-JCC1>3.0.CO;2-P.

[120] Xiaolin Pan et al. "MolGpka: A Web Server for Small Molecule pKa Prediction Using a Graph-Convolutional Neural Network". In: *Journal of Chemical Information and Modeling* 61.7 (July 2021), pp. 3159–3165. DOI: 10.1021/acs.jcim.1c00075.

[121] M. J. Frisch et al. *Gaussian˜16 Revision C.01*. Gaussian Inc. Wallingford CT. 2016.

[122] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

Eidesstattliche Versicherung (Affidavit)

_____ _____

Name, Vorname
(Surname, first name)

Matrikel-Nr.
(Enrolment number)

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden, § 63 Abs. 5 Hochschulgesetz NRW.

Die Abgabe einer falschen Versicherung an Eides statt ist strafbar.

Wer vorsätzlich eine falsche Versicherung an Eides statt abgibt, kann mit einer Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft werden, § 156 StGB. Die fahrlässige Abgabe einer falschen Versicherung an Eides statt kann mit einer Freiheitsstrafe bis zu einem Jahr oder Geldstrafe bestraft werden, § 161 StGB.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:

Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offence can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offences of this type is the chancellor of the TU Dortmund University. In the case of multiple or other serious attempts at deception, the candidate can also be unenrolled, Section 63, paragraph 5 of the Universities Act of North Rhine-Westphalia.

The submission of a false affidavit is punishable.

Any person who intentionally submits a false affidavit can be punished with a prison sentence of up to three years or a fine, Section 156 of the Criminal Code. The negligent submission of a false affidavit can be punished with a prison sentence of up to one year or a fine, Section 161 of the Criminal Code.

I have taken note of the above official notification.

_____ _____

Ort, Datum
(Place, date)

Unterschrift
(Signature)

Titel der Dissertation:
(Title of the thesis):

_____

_____

_____

Ich versichere hiermit an Eides statt, dass ich die vorliegende Dissertation mit dem Titel selbstständig und ohne unzulässige fremde Hilfe angefertigt habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht.
Die Arbeit hat in gegenwärtiger oder in einer anderen Fassung weder der TU Dortmund noch einer anderen Hochschule im Zusammenhang mit einer staatlichen oder akademischen Prüfung vorgelegen.

I hereby swear that I have completed the present dissertation independently and without inadmissible external support. I have not used any sources or tools other than those indicated and have identified literal and analogous quotations.

The thesis in its current version or another version has not been presented to the TU Dortmund University or another university in connection with a state or academic examination.*

*Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the PhD thesis is the official and legally binding version.

_____ _____

Ort, Datum
(Place, date)

Unterschrift
(Signature)