



TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Chemie

Bayerisches NMR-Zentrum

Lehrstuhl für biomolekulare NMR Spektroskopie

Modelling False Positives in High Throughput Assays

DIPAN GHOSH

Vollständiger Abdruck der von der Fakultät für Chemie der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

genehmigten Dissertation.

Vorsitzende : Prof. Dr. Angela Casini

Prüfer der Dissertation:

1. Prof. Dr. Michael Sattler

2. Dr. Igor Tetko

Die Dissertation wurde am 24.02.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Chemie am 02.07.2021 angenommen.

*Dedicated to my parents and brother who have been a constant
source of support and joy*

Abstract

Medicine plays a monumental role in modern society, and with it the importance of developing new and safe drugs has become paramount. While therapeutic antibodies and biologics are emerging as novel therapeutic entities, small molecules remain as most important drugs in use, as they usually are advantageous regarding cost, delivery (pills) and storage and distribution. However, developing a novel drug is time- and resource-intensive and expensive. Therefore, it is vital to expedite the process of generating valid lead compounds in the early stages of drug discovery.

With the advancement in automation, high throughput assays have become routine in early-stage drug discovery. Although such methods are instrumental in high-throughput screening (HTS) an extensive collection of compounds, a significant inherent problem is the presence of false positive hits, depending on the specific screening assay used. Different mechanisms of how compounds may interfere with the assay can give rise to a false-positive result. In fact, some compounds are found to show promiscuous activity across different assay types and targets. Such unwanted hits should be identified and eliminated during the initial stages of screening to focus further developments on true hits.

In this thesis, high-performance, state-of-the-art machine learning models have been developed and applied to identify false positives and promiscuous hits for three different assay systems that are commonly used in early-stage drug discovery. First, the problem of assay interference due to inhibition of luciferase, an enzyme widely used as a reporter is discussed. From publicly available luciferase counterscreen assay data, a robust machine learning model with balanced accuracy of 89% is built. Second, the performance of a machine learning model is compared with more traditional scaffold-based filters, that were developed from in-house AlphaScreen data, and then tested against public datasets. It was found that machine learning methods outperform the scaffold-based methods significantly. A new machine learning model for filtering AlphaScreen frequent hitters was also developed which predicts AlphaScreen frequent hitters with a Balanced Accuracy score of 84.7%. Third, publicly available data were analysed to develop a model for flagging frequent hitters in GPCR assays. Assays with GPCRs that are distant in the phylogenetic tree were chosen to avoid identifying GPCR-specific scaffolds. Assays using fluorescence technology were found to contain more frequent hitters than assays using bioluminescence. The model developed using 10 datasets flags potential GPCR frequent hitters with 86.0% balanced accuracy.

All the machine-learning models described were built using the OCHEM platform (<http://ochem.eu>) and are freely available to the public.

The results provide useful tools to enhance the quality and efficiency of early-stage drug discovery.

Zusammenfassung

Die Medizin spielt in der modernen Gesellschaft eine essentielle Rolle. Die Entwicklung neuer und sicherer Medikamente ist hierbei von größter Bedeutung. Während therapeutische Antikörper und Biologika als neuartige therapeutische Moleküle an Bedeutung gewinnen, bleiben kleine Moleküle die wichtigsten verwendeten Medikamente, da sie im Allgemeinen hinsichtlich Kosten, Applikation (Pillen) sowie Lagerung und Verteilung vorteilhaft sind. Die Entwicklung eines neuartigen Arzneimittels ist jedoch Zeit- und Ressourcen intensiv und damit teuer. Daher ist es wichtig, den Prozess der Erzeugung gültiger Leitstrukturen in den frühen Stadien der Wirkstoffentdeckung zu beschleunigen.

Mit dem Fortschritt von Automatisierungstechniken, ist „High Throughput Screening“ (HTS) ein Standardverfahren in der frühen Wirkstofffindung geworden. Ein Problem dieser Verfahren ist, dass oft Moleküle fälschlicherweise als aktiv identifiziert werden. Solche Verbindungen werden als „Falsch-Positive“ bezeichnet. Verschiedene Studien haben die Mechanismen, die zur Interferenz mit der im Assay verwendeten Detektionsmethode und damit zu falsch-positiven Verbindungen führen, aufgeklärt. Ebenso wurden sogenannte „promiske“ Substanzen, also Moleküle, die unabhängig von der verwendeten Assaymethode oder dem Zielprotein aktiv sind, beschrieben. Diese Verbindungen werden in der Literatur auch als „frequent hitter“ bezeichnet. Die vorliegende Arbeit hat zum Ziel, solche Moleküle zu identifizieren und möglichst früh aus dem Drug Discovery Prozess zu entfernen.

In dieser Arbeit wird die Entwicklung von „Machine Learning“ (ML) Verfahren zur Identifizierung solcher Verbindungen für drei verschiedene Assaymethoden beschrieben. (1) Luciferase ist ein in zahlreichen Assays verwendetes Detektorprotein. Verbindungen, die Luciferase inhibieren, erscheinen daher aktiv in einem Luciferase Assay. Unter Verwendung öffentlich zugänglicher Luciferase Assay Daten wurde ein robustes ML Modell mit einer „balanced accuracy“ von 89% entwickelt. (2) ML Modelle wurden auch für Daten aus „Alphascreen“ Kampagnen entwickelt und mit traditionellen, Modellen verglichen, die auf der Identifizierung gemeinsamer Substrukturen basieren. Es konnte für Daten aus proprietären und öffentlichen HTS Kampagnen gezeigt werden, dass die ML Modelle den substruktur-basierten Modellen überlegen sind. Ein neues ML Verfahren zur Identifizierung von falsch-positiven Verbindungen wurde entwickelt. Die „balanced accuracy“ dieses Modells liegt für die Identifizierung falsch-positiver bei 84.7%. (3) Schliesslich wurden ML Modelle zur Identifizierung von „frequent hittern“ in GPCR Assays entwickelt. Hierzu wurden Daten aus der Datenbank Pubchem verwendet. Dabei wurden GPCRs ausgewählt, die phylogenetisch möglichst wenig Ähnlichkeit aufweisen, um zu verhindern, daß Substanzklassen, die infolge ihrer privilegierten Struktur an mehrere verwandte GPCRs binden, fälschlicherweise als „frequent hitter“ identifiziert werden. Die Wahrscheinlichkeit „frequent hitter“ zu identifizieren, war grösser für Assays, die Fluoreszenz Methoden verwenden, als für solche, deren Detektionsmethode auf Biolumineszenz beruht. Die „balanced accuracy“ des Modells zur Vorhersage von „frequent hittern“, das auf 10 GPCR Datensätzen basiert liegt bei 80%.

Alle ML Modelle wurden unter Verwendung der OCHEM Plattform (<http://ochem.eu>) erstellt und stehen der Öffentlichkeit frei zur Verfügung.

Table of Contents

Modelling False Positives in High Throughput Assays	i
Abstract	iii
Zusammenfassung	iv
Introduction	1
1.1. A brief overview of the drug discovery pipeline	2
1.1.1. Target Identification	2
1.1.2. Screening	3
1.1.3. Hit Validation	3
1.1.4. Lead Optimization	3
1.1.5. Clinical Trials	3
1.2. High Throughput Assays:	3
1.3. False positives and Frequent Hitters:	4
1.4. Mechanism of False Positives:	4
1.4.1. Interference with the assay:	4
1.4.2. Reactive Species:	5
1.4.3. Aggregation:	5
1.5. Pan Assay Interference Compounds:	6
1.6. Identification of false positives:	6
1.6.1. Biochemical Methods:	6
1.6.2. Biophysical Methods:	7
1.6.3. Computational approaches:	9
1.7. Machine Learning:	11
1.7.1. What is Machine Learning?	11
1.7.2. Classification of ML Implementations	11
1.8. About this Thesis	13
Methods	14
2.1. Data Collection	15
2.1.1. Gathering Assay Data	15

2.1.2. Gathering Structural Data	16
2.2. Data Analysis	16
2.3. Tools used in the study:	19
2.3.1. Programming Language: Python	19
2.3.2. Scaffold-Hunter ⁶⁵	21
2.3.3. LigandScout ⁸⁰	21
2.3.4. AutoDock	22
2.4. Machine Learning	22
2.4.1. OCHEM: Online Chemical Modelling Database	22
2.4.2. Overview of Common ML algorithms	23
2.4.1. Variations of the ANN Algorithm	30
2.4.2. Available Descriptors in OCHEM:	31
2.1. Model Training in OCHEM	35
2.1.1. Data Upload	35
2.1.2. Building a model	36
2.2. Model evaluation parameters	37
2.2.1. Sensitivity and specificity	37
2.2.2. Accuracy	40
2.2.3. Balanced Accuracy	40
2.2.4. Matthews Correlation Coefficient	41
2.2.5. ROC-AUC	41
2.2.6. Test set Evaluation	41
2.2.7. Cross validation	42
2.2.8. Bagging Validation	43
2.3. Applicability Domain of QSAR Models	43
2.3.1. Distance to model	43
2.3.2. Model Performance with Applicability Domain	45
Motivation	47
Modelling False Positive Hits in Luciferase HTS Assays	48
3.1. Project Introduction	49
3.2. Data	50
3.3. Methods	51
3.3.1. Docking studies	51
3.3.2. Pharmacophore Analysis	52
3.3.3. Machine learning methods.	52
3.3.4. Molecular descriptors.	52

3.3.5. Statistical coefficients.	52
3.4. Results	53
3.4.1. Molecular Docking	53
3.4.2. Scaffold Analysis	54
3.4.3. Pharmacophore analysis	56
3.4.4. Machine Learning Models	57
3.4.5. Sensitivity of existing filters	61
3.5. Discussion	62
3.6. Project Conclusions	62

Machine Learning model to filter Frequent Hitters for AlphaScreen assays

.....	63
4.1. Introduction	64
4.2. Data	65
4.3. Methods	66
4.3.1. Machine learning methods	66
4.3.2. Molecular descriptors	66
4.3.3. Statistical coefficients	67
4.4. Results	67
4.4.1. Frequent Hitter Analysis	67
4.4.2. Machine-learning Models:	67
4.4.3. Comparison between Scaffold based Filter and Machine-learning Models:	68
4.4.4. Machine learning models to identify mechanism of action of FHs	71

Modelling False Positives in GPCR assays

5.1. Introduction	76
5.2. Data	76
5.2.1. Data description	76
5.2.2. Data Collection	76
5.2.3. Frequent Hitter Flagging	78
5.3. Methods	78
5.3.1. Data Gathering:	78
5.3.2. Activity Cross-check:	78
5.4. Machine Learning	78
5.5. Results and Discussion	79
5.5.1. Compound Activity Profile:	79

5.5.2. Scaffold Identification:	79
5.5.3. Machine Learning	80
5.6. Conclusion	81
Comparison with other tools	82
Thesis Discussion	83
Thesis Conclusion	86
Acknowledgments	87
References	88
Appendix I: PCA Pipeline with Jupyter Notebook	106
Appendix II	123

Chapter 1
Introduction

In the year 1938, E. Chain, Howard Florey, and their collaborators selected penicillin, a metabolite from a penicillium for further study as an antibiotic. Penicillin proved to be a success in treating bacterial infections and started turning the wheels of pharmacology and drug discovery^{1,2}.

Over the past century, humanity has made enormous progress in understanding the world around it. As each branch of science develops, we are provided with better tools with which to understand and potentially override natural design. With a proper understanding of chemistry, and the molecular structure of novel drugs and their endpoints, the synthetic accessibility of new compounds are increasingly within our grasp. On the other hand, a better understanding of the cell and its vast array of molecular machinery and pathways, leads to new ways of targeting said machinery, and raises new questions on a daily basis. New developments in microscopy have led to the direct visualization of living cells in real-time³⁻⁵, and X-Ray crystallography has led to the elucidation of complex biological structures, such as DNA⁶ and proteins^{7,8}. Over the last few decades, automation and computers have become a key part of the drug discovery effort. Drug discovery is a multi-disciplinary effort, with experts participating from almost all fields of science^{9,10}. Biotech and pharma industries are delving into fundamental research, often collaborating actively with academic partners. Analysis of Big Data using cheminformatics and machine learning methods contribute to the fast development of the field^{11,12}.

1.1. A brief overview of the drug discovery pipeline

Early in human history, people developed remedies from natural sources. This was mostly governed by observations and serendipity, and there are many known instances of such remedies actually being extremely harmful to the point of lethality; eg, Mercury. As scientific discoveries led to an understanding of natural systems, it propelled forward an understanding of these traditional remedies. Isolation of the active ingredient, and establishing a causal relationship between the drug and its effects became a major part of pharmaceutical research. More recently, screening libraries of chemical compounds or natural products against intact cells or live organisms became popular for finding a desired effect or phenotype. This is known as phenotypic screening and is still popular today. However, the chances of finding a lead that is worth pursuing in such non-targeted screening is very slim, and therefore a more refined, targeted, rational approach is often called for.

1.1.1. Target Identification

The first step in the drug discovery process is identification of a biological target, the modulation of which may lead to a desired phenotype, such as alleviation of a symptom. Such target identification often comes from basic scientific research, or a systematic genomic analysis of human genetic data deposited in biobanks. The target of a modern drug are usually the protein machineries, often enzymes. However, only 10% of the human genome are known to be drug-gable, and thus, even if the cause of a disease is known it may be challenging to find a good drug target for that disease.

1.1.2. Screening

Once a druggable target is identified, the next step involves finding chemical or biological entities that will interact with the target. A virtual screening may be a first step in identifying a list of compounds for *in-vitro* screening. A robot driven high-throughput or ultra-high-throughput screening may also be used to identify potential lead compounds from a huge library of commercially available or *in-house* compound library. Much of this thesis is concerned with refining the hitlist from such large screens and identifying potential bad actors early on in the process in order to minimize cost and effort. We will discuss them in detail in the following section.

1.1.3. Hit Validation

The list of hits from such screens must be verified, and this is known as validation of hits. A variety of techniques including biochemical assays, structural and spectroscopic techniques are used to probe, elucidate and validate the interaction between the hit and the target. We will discuss some of such techniques in the following sections.

1.1.4. Lead Optimization

Once the hit is confirmed, it may be called a *lead*. The most promising leads are then subjected to a wide range of pharmacological and toxicological studies and often the molecular design is optimized considerably to improve their pharmacokinetics or address key properties such as solubility or oral availability. The compounds are tested both on live cells (animal or human cell lines) and in animal (typically mouse) models. Only when their efficacy is demonstrated *in-vivo*, can a compound be subjected to clinical trials.

1.1.5. Clinical Trials

Preclinical research and pharmacological parameters are only an indication of how a given drug will interact with the human body. Clinical trials refer to the studies that are done on increasingly larger groups of people in order to ascertain the effectiveness of drug and eliminate off-target activity or toxicity, before the drug is made commercially available. There are three phases of clinical trials, and only 1 out of 20 compounds that enter a clinical trial are released commercially as a drug. Being at the very end of the pipeline, failure at this stage is very expensive and it is the major contributor for the overall expense of a drug development pipeline.

In this thesis, we are focussed on the early stage of the drug discover pipeline, namely screening and hit selection. We will discuss them briefly below.

1.2. High Throughput Assays:

A critical step in discovering new drugs is to test candidate molecules against the intended target or otherwise-related targets and look for an effect. This process is known as a biochemical assay. There are many different kinds of well-established assays, with various detection tech-

nologies each suited for different scenarios¹³. With the advent of automation, high-throughput assay systems have become popular¹⁴⁻¹⁶. In such systems, hundreds of thousands, sometimes millions of compounds can be tested against given endpoints using automated systems in a relatively short time.

1.3. False positives and Frequent Hitters:

High throughput Screening (HTS) enables direct testing of more compounds than ever, but analysing these hits and finding good results is a challenging task. There always exists the possibility of false positives mixed in with ‘true’ hits¹⁷; there can also be compounds that are generally promiscuous and are not a specific binder to our intended target¹⁸. Identifying such molecules is the primary aim of this project.

The two different kinds of molecules that are the subject of this project are *false positives* and *frequent hitters*. Sometimes, the distinction between the two types of molecules is not clear, so before proceeding, what is meant by each should be clearly defined. In binary classification, a *false positive* (FP) is where the test result improperly indicates the presence of an affirmative assay result, in our case, a hit. These molecules, therefore, usually do not produce a desirable interaction in the assay, despite giving a positive result in the assay detection mechanism. A frequent hitter, on the other hand, is a molecule that shows up as a hit in many different biological assays covering a wide range of targets¹⁸. These molecules may produce the intended interaction but are not specific to any particular target, and hence, are considered a poor drug candidate. Both classes of molecules are potentially unwanted, and their identification is an important part of recognising unwanted hits from an HTS.

1.4. Mechanism of False Positives:

Biochemical assays involve testing the effect of a compound on an assay system: The assay systems can typically be anything from a single enzyme to multilayer binding cascades, but the results of such events are usually optical measurements, such as absorbance, fluorescence, luminescence, or scintillation counting. The assays are typically carried out in microtiter plates, and automated systems take the measurements from each well, to be analysed later.

1.4.1. Interference with the assay:

Choosing compatible assay systems and targets is the first step in avoiding false positives in an assay. It is essential to understand that biochemical assays do not measure biological activity, but molecular interactions. If the compounds, the target, and the assay system are not compatible with each other, unwanted interactions may lead to false positives.

1. Interference with detection technology:

Test compounds may directly interfere with the readout signal. Fluorescence quenchers can foil an assay using fluorescence as the detection mechanism^{19,20}. If a compound inhibits luciferase,

assays using luciferase as a reporter enzyme will fail²¹⁻²³.

2. Interference with assay technology:

Target proteins may be sensitive to a class of compounds, or their activity may be modulated indirectly by the assay system employed. For example, nucleophilic protease assays are particularly susceptible to protein-reactive electrophiles. Similarly, phosphatase assays are sensitive to phosphorylated compounds, salts, metals, and polyions²⁴.

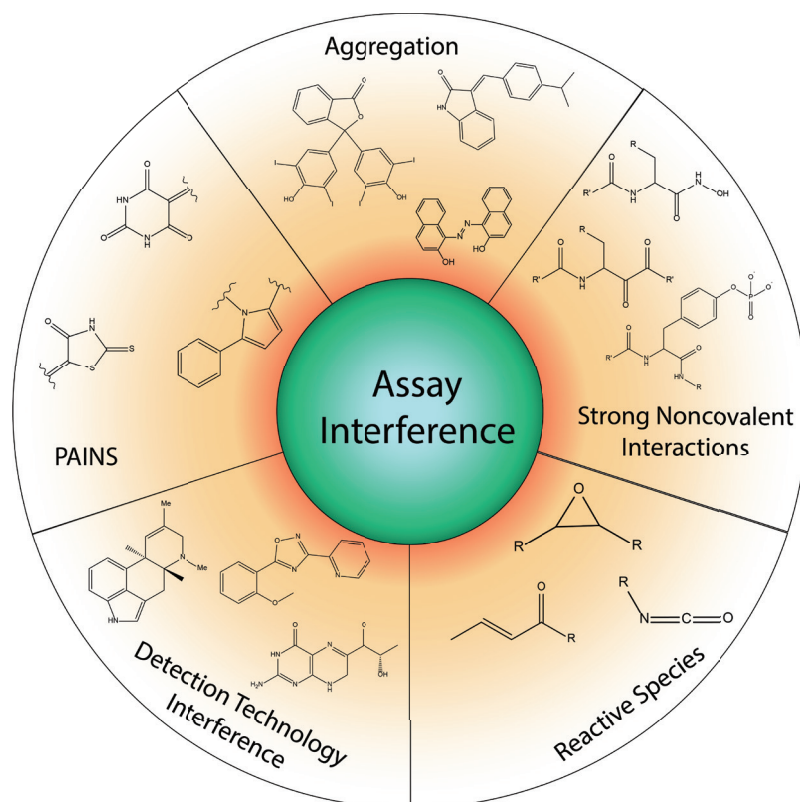


Figure 1.1: Different modes of assay interference and a few examples of each.

1.4.2. Reactive Species:

Although some drugs act by binding to their target covalently²⁵, in an assay system, the same kind of reactive species can indiscriminately attach themselves to proteins or DNA irreversibly. Reactive species may also oxidize susceptible target proteins, such as metalloenzymes and cysteine-using enzymes. Such oxidation may lead to, say, inhibition of the target enzyme, but such indiscriminate reactive molecules are not useful for therapeutic purposes. Therefore, such reactive species lead to false positives. Such compounds can often be identified based on their chemical structures and inhibition pattern²⁶⁻²⁹. Being protein reactive, they are often active on the counterscreen enzymes. Therefore, a counterscreen combined with the irreversibility of the inhibition and mass spectrometry of the target protein may help identify the presence of such molecules in an assay system.

1.4.3. Aggregation:

In 2002 Shoichet *et al.* first reported that aggregates formed in solution by certain compounds

could non-specifically inhibit enzymes³⁰. Since the initial report, the same phenomenon has been observed in kinase inhibitors³¹, cruzain inhibitors³², kinesin motor protein inhibitors³³, amyloid polymerization³⁴, and non-nucleoside reverse transcriptase inhibitors (NNRTIs)³⁵. Above a particular concentration known as critical aggregation concentration (CAC), typically in the low-to-mid micromolar compound concentration range, these molecules form solid colloids composed of up to 10⁸ small-molecules and are several hundred nanometers in mean diameter. For enzyme systems, a reversible association between the aggregate surface and the protein was observed^{30, 36}. Aggregates formed by promiscuous compounds reversibly sequester enzyme, resulting in apparent inhibition³⁶.

1.5. Pan Assay Interference Compounds:

In 2010, Jonathan B. Baell and Georgina A. Holloway reported a new class of compounds that turn up as hits in different assays³⁷. They termed them **Pan Assay Interference Compounds** or PAINS. They also published common scaffolds of such compounds implemented as a filter in Sybyl notation. The report received wide recognition by the scientific community, and the term PAINS became synonymous with unwanted compounds in an assay. In our current context, such compounds are defined as frequent hitters. The filters the Baell *et al.* published became widespread and are still in use today to weed out potential offenders, sometimes even before doing an assay³⁸⁻⁴³.

However, the PAINS filter is not a magic bullet to solve the frequent hitter problem. In 2018, J. B. Baell published a review⁴⁴ on PAINS where he states “It has become increasingly clear that overzealous or simplistic use of these filters may inappropriately exclude a useful compound from consideration and inappropriately tag a useless compound as worthy of development.”

Baell *et al.* developed the PAINS filter from a set of six independent high-throughput assays, all of which were Alphascreen assays. The compound set was relatively small, only about 100,000 compounds. The filters were derived with observation alone, not taking pharmacokinetics or toxicology into account. These factors combined limits where applying a PAINS filter is effective, versus where it can be potentially misleading. Nevertheless, the seminal work by Baell and Holloway paved the way for computational filters for identifying false positives and frequent hitters in a high throughput assay campaign.

1.6. Identification of false positives:

1.6.1. Biochemical Methods:

1. Detergent Sensitivity

Detergent such as Triton-X100 impede the formation of aggregates. Performing assays both with and without detergent might help identify potential aggregators that may contribute to the false-positive signal^{45, 46}.

2. Sensitivity to Concentration

The ability of a nanomolar inhibitor to titrate at micromolar concentration implies the formation of an aggregate^{30, 36}. Titrating under varying concentrations may help identify compounds with such properties⁴⁷

Preincubation

Steep inhibition curves are often associated with undesirable screening hits. Performing two assays with identical conditions, but incubating one for longer may identify compounds that have slow on-rates, binds covalently, or form aggregates^{30, 31, 36}.

3. Counterscreen with dissimilar Enzymes

A compound interfering with assay technology will show up as positive even with completely unrelated targets. Performing multiple assays and finding frequent hits may help filter out assay technology-related artifacts.

4. Reactivity Profiling

Careful consideration of functional groups and reactivity may help identify compounds that bind to the target covalently and lead to false hits. For example, reactivity towards glutathione and toward thiols in general, can be measured using a competitive assay system developed by Epps *et al.*²⁹ Similar assay systems are also available for checking redox activity of small molecules⁴⁸, which are particularly important when the target is susceptible to oxidation.

1.6.2. Biophysical Methods:

Several biophysical methods can be used to verify the interaction between the ligand and the protein directly. Such experiments take effort and time to perform but may provide conclusive evidence of protein-ligand interaction. Some of the methods, such as dynamic light scattering and electron microscopy, can be used to observe the aggregates directly.

1. Dynamic Light Scattering

Aggregates can be identified directly using Dynamic Light Scattering (DLS)⁴⁹. In this technique, monochromatic light scatters off the particles suspended in a solution, and the scattered light is detected as the signal and analyzed. The larger the particle, the slower its Brownian motion, which leads to a larger variation in the scattering intensities. Because of this, aggregate-forming inhibitors yield an autocorrelation function with well-defined decay on the microsecond timescale and can be easily identified. Using DLS, the particle size of the aggregates can also be calculated.

2. Surface Plasmon Resonance

Surface plasmon Resonance or SPR can be used to calculate the binding constant between a protein target and a ligand, among a variety of other uses. In this technique, light is focused on a

sensor chip containing free electrons known as surface plasmons. At a specific angle, resonance is observed between the incident electromagnetic wave and the surface plasmon. This angle is sensitive to the local environment of the SPR chip. For determining a binding constant, target proteins are immobilized on to the chip surface, and ligands are passed using a flow system. If binding occurs, a resonance angle shift is observed. Then the ligand is washed off with some buffer, and another shift is observed. Monitoring these shifts with respect to time, association and dissociation constants (k_{on} and k_{off} respectively) can be calculated, which can then be used to calculate the binding constant. This method provides a conclusive and quantitative estimation of the interaction between the target protein and the ligand. However, this technique is labor and cost-intensive, therefore not suitable for testing more than a few ligands.

3. NMR and Mass Spectrometry

Multidimensional Nuclear Magnetic Resonance can be used to identify interactions between

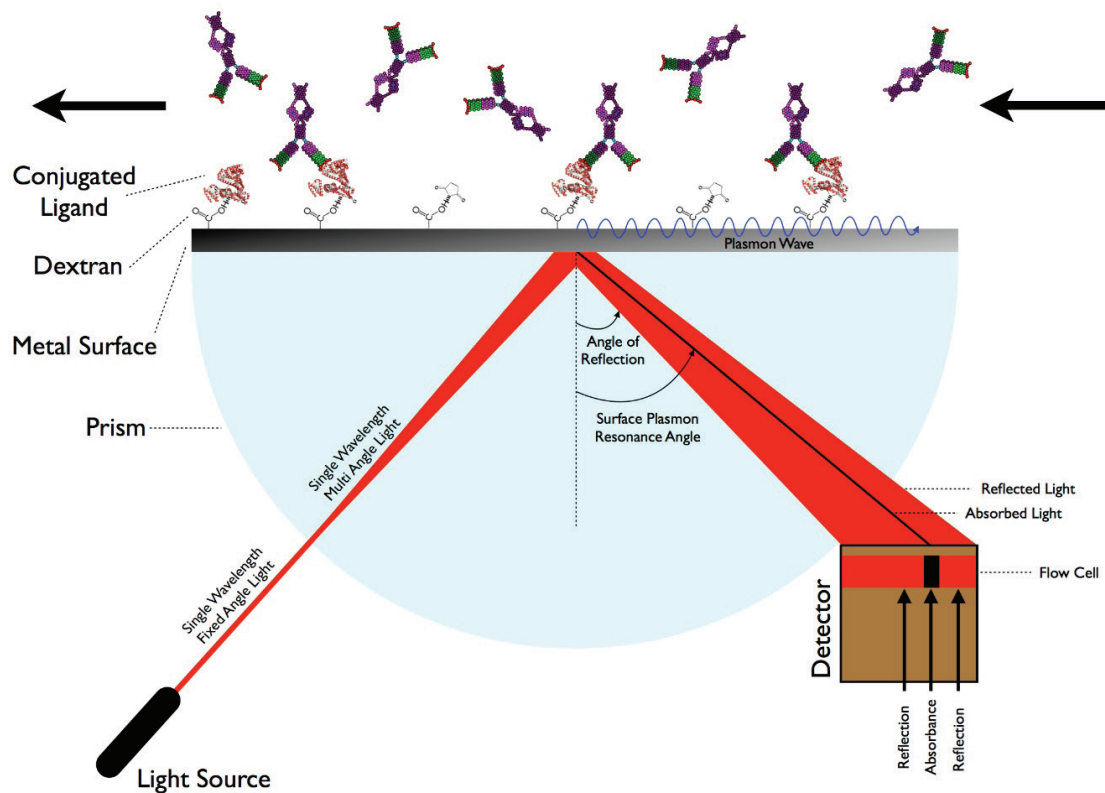


Figure 1.2: Setup for determining protein-ligand binding using Surface Plasmon Resonance.

Source: Wikipedia

two separate molecules, such as a target protein and a ligand. Mass spectrometry can measure the molecular mass of a species with extreme precision, validating or ruling out covalent bond formation. Such methods provide conclusive evidence, but are not suitable for high throughput applications.

4. Electron Microscopy

Electron microscopy can be used to observe any aggregate formation. Confocal microscopy has

been used to characterize interactions between aggregates and protein molecules. Aggregation of proteins that are associated with pathological conditions, such as Amyloid, have been characterized using Transmission Electron Microscopy (TEM)^{50, 51}.

1.6.3. Computational approaches:

Over the past few years, efforts to build a computational filter to flag potential false positives and frequent hitters has intensified. In contrast with the biochemical and biophysical techniques described above, there are certain advantages to a computational filter. However, such filters do have their limitations, and using one blindly can lead one down the wrong path.

The most common type of cheminformatic filter is a substructure-based filter. Over the last 25 years, many such filters have been developed, aimed at solving a wide range of cases^{37, 52-56}. Most of these filters are based on observation. A general ruleset is derived from a set of data, which is then implemented as a filter that looks for substructures, functional groups and calculates various molecular properties to filter against the given ruleset.

In literature, there are a plethora of articles describing various rulesets for filtering out potentially unwanted compounds^{13, 37, 39, 57-61}. Such rulesets have been applied most successfully in detecting and eliminating reactive species. Because such reactivity often depends on the functional groups present, relatively simple functional group filters can flag the molecules successfully¹⁷.

In 1999, Walters *et al.* published REOS (Rapid Elimination of Swill), which is a program developed to filter out potentially unwanted compounds using a hybrid method⁶². REOS combines functional group filters with Lipinski's Rule of 5. The initial filtering is based on a set of property filters in conjunction with another set of more than 200 rules based on chemical functionalities known to be problematic. REOS allows users to specify a maximum allowed quantity for each functional group role, rather than just accept or reject it. REOS is one of the first efforts towards creating a computational filter for eliminating artifacts from an assay.

In 2002, Gisbert Schneider and colleagues developed a virtual screening method to identify compounds that are frequently active, and coined the term frequent hitter¹⁸. They developed a scoring scheme from substructural analysis and multivariate linear and nonlinear statistical methods applied to several sets of one and two-dimensional molecular descriptors. The final model, based on a three-layer neural network, was able to correctly classify 90% of the test set molecules in a 10-times cross-validation study. This report is the first application of machine learning to identify frequent hitters.

In 2010, Baell and Holloway published the now-famous PAINS filter. Along with its widespread usage, its applicability sparked controversy. Numerous reports were published showing that the PAINS filter has many shortcomings, and should be applied in context. One of the most significant report in this context was published by Tropsha *et al.* in 2017⁶³. They found that the majority (97%) of all compounds containing PAINS alerts were infrequent hitters in AlphaScreen assays measuring protein-protein interaction inhibition. They also reported that the presence of PAINS alerts did not correlate with any heightened assay activity trends across all

assays in PubChem including AlphaScreen, luciferase, beta-lactamase, or fluorescence-based assays. Also, 109 PAINS alerts were present in 3570 extensively assayed, but consistently inactive compounds called Dark Chemical Matter. In 2018, J. B. Baell published a review discussing and highlighting the considerations that should be taken into account when using the PAINS filter⁴⁴.

In 2016, Bologna *et al.* developed a system for identifying likely promiscuous compounds via associated scaffolds, which they called Badapple (bioassay-data associative promiscuity pattern learning engine)⁶⁴. Badapple is a scaffold-based filter for flagging frequent hitters. The authors defined “promiscuity” as the multiplicity of positive non-duplicate bioassay results. At the core, Badapple employs a “Bayesian-like” method to calculate a score for a given scaffold in a dataset by the formula

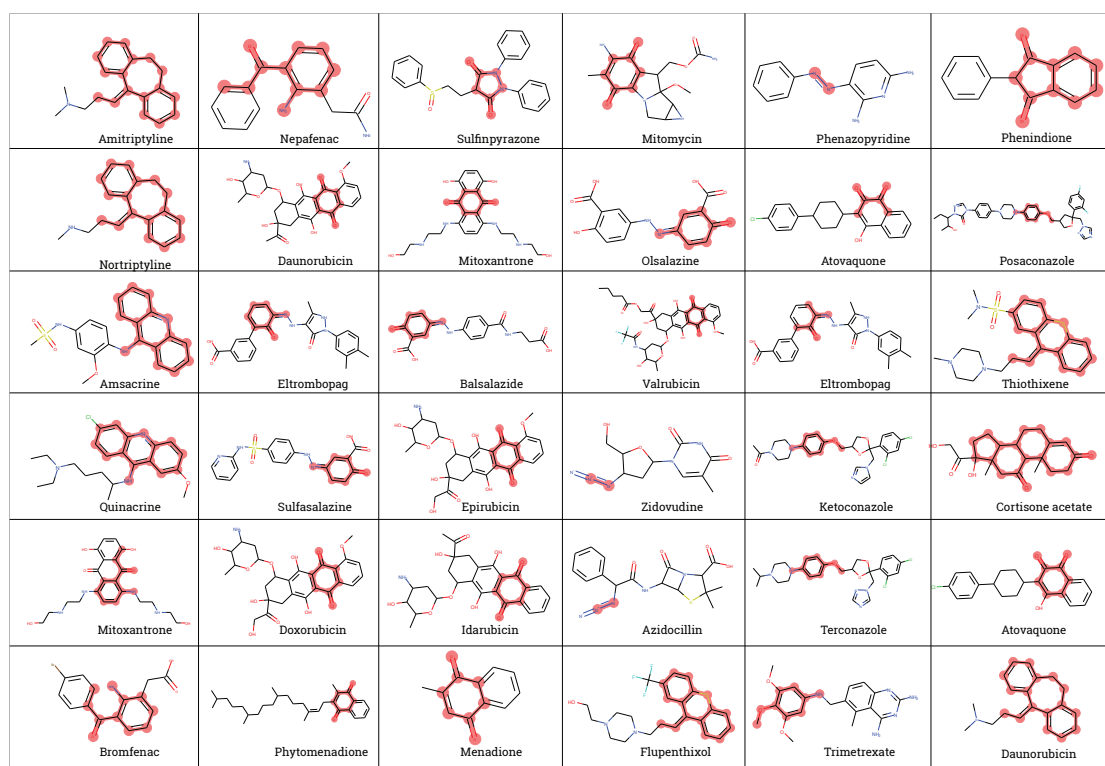


Figure 1.3 : Approved Drugs that contain PAINS scaffolds. Data collected from the article by Tropsha *et al.*⁶³.

$$Score = \frac{s_A}{s_T + med(s_T)} \times \frac{a_A}{a_T + med(a_T)} \times \frac{w_A}{w_T + med(w_T)} \times 10^5 \quad [1]$$

where s_T = tested substances with scaffold, s_A = active substances with scaffold, a_T = assays with tested compounds with scaffold, a_A = assays with active compounds with scaffold, w_T = tested samples with scaffold, w_A = active samples with scaffold, med = median. For a high score, all three terms have to be high. Noisy data requires sufficient sampling to produce evidence, and the Badapple formula mitigates noise by aggregating across samples and substances. Doing this reduces the noise in the data and increases confidence in the score as more of the same scaffolds

are found active repeatedly. On the other hand, Badapple remains sceptical of scanty evidence.

In 2013, Hadian *et al.* published a scaffold-based filter for identifying frequent hitters of AlphaScreen assays¹⁹. From four primary in-house assays, the authors identified common active molecules, and then looked for overrepresented scaffolds among them. They used Scaffold Hunter⁶⁵, ISIDA⁶⁶, and Silicos-IT scaffolds⁶⁷ using the SetCompare⁶⁸ utility of the online chemical modeling environment (OCHEM) platform and identified overrepresented structural elements in the set of promiscuous compounds. In total, 60 molecules were identified to be frequent hitters of the Alphascreen technology, and 25 scaffolds were identified to recognize them. The scaffolds were encoded as SMARTS strings and implemented as a filter in the freely available Online Chemical Modelling Environment⁶⁹ (OCHEM).

The latest development in the field of computational chemistry is Machine Learning. In late 2017, Kirchmair *et al.* reported the first machine learning model for identifying false positives and frequent hitters in a high throughput assay⁷⁰. The model, named HitDexter was developed from 427 657 unique compounds with activity data on a total of 653 unique proteins available in PubChem. The authors used two extremely randomized tree-based classifiers for the prediction of compounds that are likely to be frequently active. Their final model reached MCC and AUC values of up to 0.67 and 0.96 on an independent test set, respectively. The model is freely available as web-service, and alongside the prediction score, the service also provides five nearest neighbors from the training data, which helps the user validate the model score.

1.7. Machine Learning:

Machine learning is involved in a major part of this thesis. So, it is useful to review the basics of Machine Learning, various techniques of Machine Learning, including Deep Learning and learn about various applications of Machine learning in the field of cheminformatics and drug discovery. Machine learning is an emergent field and is under heavy research. The basic definitions will be discussed here, details of various algorithms used throughout this thesis can be found in the Methods section.

1.7.1. What is Machine Learning?

It is difficult to generalize all of machine learning in a simple sentence, but stated simply, machine learning is the study of a set of algorithms that allows computers to learn and make decisions based on the given data, without being explicitly programmed⁷¹. Machine learning can be considered a key component of Artificial Intelligence (AI), which aims to develop intelligent autonomous computer systems.

1.7.2. Classification of ML Implementations

Based on the learning approach, machine learning can be classified into three types: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

1. Supervised Learning:

In supervised learning, the model is trained using labelled data. The input parameters and the required output are both given as the training data. The algorithm then tries to come up with a function to map the input parameters to the required output.

2. Unsupervised Learning

In unsupervised learning, the system learns without any associated response or target, thus identifying general patterns in the data. Such systems are useful in identifying trends and clustering unknown data to discover new features. The features thus discovered can then be used as a descriptor in a supervised learning algorithm. Many recommendation systems on the internet, such as in marketing automation, use such implementations.

3. Reinforcement Learning

In reinforced learning, the machine learns to make an appropriate decision based on the input data, by trial and error. For any decision the machine takes, a reward or punishment is issued based on whether the decision was correct. By repeating the process and

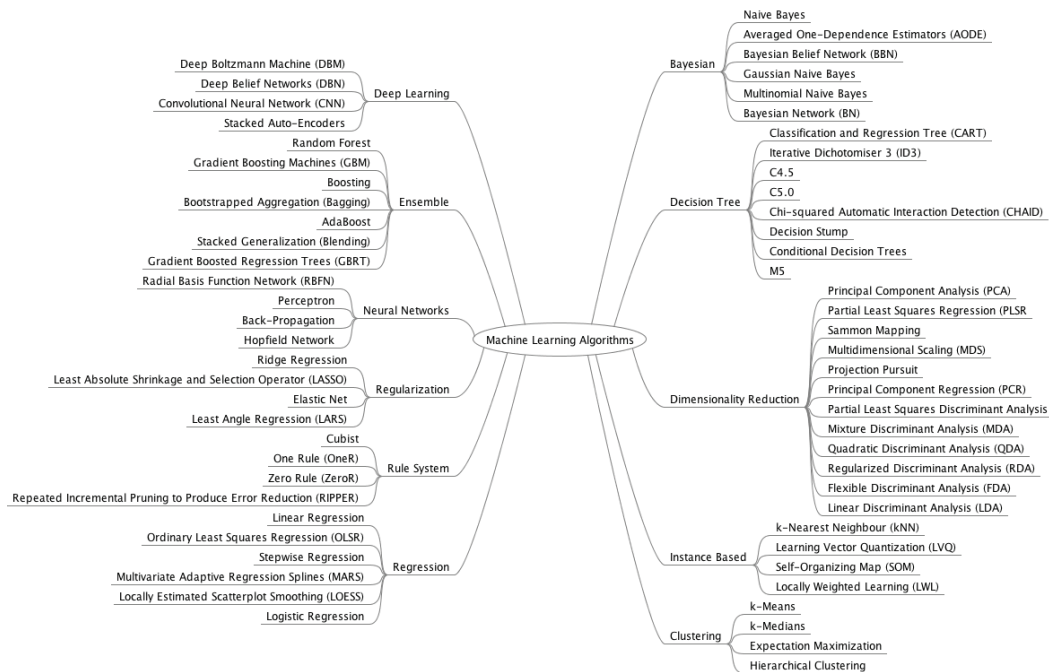


Figure 1.4: Chart showing many of the available machine learning algorithms available today. Source: machinelearningmastery.com

trying to maximize the reward, the machine slowly learns to make the correct decision based on the given data.

Based on the desired goal, machine learning tasks can be divided into three categories: Classification, Regression, and Clustering. Note that these are not types of an algorithm, but types of

problems or tasks that the machine learning system aims to solve.

Classification

For classification models, the task is to separate the data into two or more known groups or classes. For example, determining if a given compound is a frequent hitter would be a classification task, with two classes. In classification tasks, the outputs are discrete. Such tasks are generally supervised.

Regression

In Regression, the output of the model is continuous, rather than discrete. Predicting molecular properties such as melting point, water solubility, etc. are examples of such models. Generally, these tasks are also supervised.

4. Clustering

Clustering involves grouping the input data based on their underlying pattern. For example, data points can be grouped based on their similarity. Such tasks are generally unsupervised.

1.8. About this Thesis

In this thesis, the development of three different frequent hitter filters, involving Luciferase, GPCR assays, and AlphaScreen will be discussed. In the following chapter, the techniques and methods used in this study will be outlined. In the third chapter, the development of a machine learning model for identifying luciferase inhibitors, along with a few other approaches of identifying luciferase inhibitors will follow. The fourth chapter is about comparing machine learning methods with scaffold-based filtering techniques. Performances of machine learning filters are compared to an existing scaffold-based filter in the fourth chapter. In the final chapter, the development of a machine learning model to identify frequent hitters in GPCR assays is discussed.

Chapter 2
Methods

For finding frequent hitters, multiple assays must be analyzed and their results compared. Therefore, this thesis involves analysis of multiple large datasets, with several hundreds of thousands of entries in each. Needless to say, such analysis requires powerful computer systems, and an efficient approach. In this section of this thesis, the tools and methods that were employed in accomplishing this are examined. First, the nature of the data being handled, and the methods by which it is handled, are discussed. Then, the various techniques used for analyzing and building models from this data are examined.

2.1. Data Collection

Before one can start analyzing data and building models, one must collect data. For gathering in-house data, typically the in-house database was queried. The implementation of such queries was generally done in python because the upstream processing and modelling was also implemented in python. The majority of data used in this study, however, were collected from on PubChem⁷².

2.1.1. Gathering Assay Data

PubChem provides excellent utilities for downloading data from their server. For downloading data regarding any assay, the assay ID must be known. For one or a few assays, the assay data can be downloaded manually by visiting <https://pubchem.ncbi.nlm.nih.gov/> and searching by the assay ID. If the assay ID is not known, a keyword search can be performed. If one is looking for luciferase counterscreen data for example, one can search for “Luciferase”, and filter the results for counterscreen assays. This is often enough, but for our purposes, many assays must be retrieved, and this work was done programmatically. To do this, first search PubChem for relevant assays by using keywords. The query general must be kept general, so as to not exclude any relevant assays. For example, if GPCR assays are being sought, one can just search for “GPCR”. The search result is then downloaded as a text file, and parsed with a custom python script, which generates a table with all the relevant data, including the Assay IDs (See **Error! Reference source not found.** S2 at page 119). Looking through the table, it is then easy to determine what kind of data is on PubChem, and pass that table on to other scripts for downloading the data.

Once the assay IDs are obtained, data can then be downloaded from PubChem in various ways.

1. PubChem Assay Download service available at the PUBCHEM website at URL <https://pubchem.ncbi.nlm.nih.gov/assay/assaydownload.cgi>. Here or more assay IDs can be specified, and the data obtained in various formats such as .xml or .csv
2. PubChem has PUG or the Power User Gateway for programmatic access to its data. There are various web-services within the umbrella of PUG including PUG-REST: A Representational State Transfer (REST)-style web service that supplies specific bits of information on one or more PubChem records. It is intended to handle short,

synchronous requests - that is, the result is given in a single call that may last at most 30s. For the majority of applications, this service was used to download data programmatically.

3. Assay data can also be downloaded from the assay page on PubChem manually.

For gathering assay data, generally the assay download service was used. However, sometimes it was necessary to look up assay data dynamically, and this is where the PUG-REST service was very useful. All the implementations for the various API calls and web requests were done in python. As a helper tool, the python package PubChemPy was used. This tool provides wrapper functions for many of the services available through PUG-REST and is often easier to deal with than PUG API.

2.1.2. Gathering Structural Data

The datasets in this thesis were often comprised of millions of data points, and hundreds of thousands of unique molecules. Like assays, molecules in PubChem also have IDs. There are two different types of IDs, Substance IDs or SIDs and Compound IDs or CIDs. A PubChem substance can be a mixture of different compounds, so we used Compound IDs for our studies. Similar to assays, once the ID is known, the data stored in PubChem about that molecule can be accessed. Compound IDs generally come from the assays. Once a list of compounds IDs with relevant assays is obtained, structural data for the molecules from PubChem can be downloaded. The PubChem Structure Download service at https://pubchem.ncbi.nlm.nih.gov/pc_fetch/pc_fetch.cgi was often used for downloading structures of one or more compounds. For our purposes, we usually supplied a list of IDs and downloaded gunzipped SMILES.

However, it is very useful to do the structure download programmatically. Then, as the assays to identify potential frequent hitters are analysed, PubChem can be queried *in-situ* for their structure. For this, the PUG-REST service was used, combined with PubChemPy. For smaller lists (up to a few thousand) this method works well. For larger sets, the PUG-REST API cannot be used because of its request volume limitations. PubChem recommends the Structure Download service for bulk downloads and for very large sets (>50000). For this, the list of compound IDs were exported, and the structures were downloaded separately.

The activity data in all the assays for a particular molecule can also be checked. This was used to look for the activity of compounds across different assays and identify frequent hitters. Python was used to implement this in order for checking cross activity of compounds found to be frequently actives in GPCR assays. The results are discussed in detail in chapter 4.

2.2. Data Analysis

For a majority of our projects, our objective was to find false positives or frequent hitters from a set of assays. Our dataset often involved a number of high-throughput assays, with over 5 million data points. Needless to say, efficient ways of sifting through such data were called for

and python was chosen for doing this.

Data exploration was the first step in the projects described here. Information must be gathered about the chemical space involved, the total number of unique compounds, the total number of compounds with any activity, before frequent hitters can be identified. For such exploration, python has excellent tools that are very efficient and therefore can be applied to big data. As a first step, extent of overlap between the assay compounds provides a good initial assessment. Many HTS campaigns come from the same source. Also, many HTS campaigns share compound li-

Compounds	Assay data						Fingerprint	Frequency (Sum/Length)
	□	■	□	□	□	□	0 1 0 0 0	0.2
		□	□	□	□	□	0 0 0 1	0.25
	□	□	□	□	□	□	0 0 1 0 0	0.2
	□	□	□	□	□	□	0 0 0 0 0	0.0
	□	□	□	□	□	■	0 0 0 0 1	0.2
	□	□	□	□	□	□	0 0 0 0	0.0
	□	□	□	□	□	□	0 1 0 0	0.25
	■	□	□	□	□	□	1 0 0 0 0 0	0.17
	■	□	□	□	□	□	0 0 0 0 0	0.0
	■	□	■	□	□	□	1 0 1 1 0	0.6
	□	□	□	□	□	□	0 0 0 0	0.0
	□	□	□	□	□	□	0 0 0 1 0 0	0.17
	□	□	□	□	□	□	0 0 0 0 0	0.0
	□	□	□	□	□	□	0 0 0 0	0.0
	□	□	□	□	□	□	1 1	1.0
	■	□	□	□	□	□	1 0 0 0 0 0	0.17
	□	□	□	□	□	□	1	1.0
	■	□	■	■	□	□	1 1 1	1.0

Figure 2.1: Method for determining activity frequency derived from activity fingerprint. Each row represents one compound, the assay data records for that compound is represented in the Assay Data column. In this column, each column of squares represents one assay. Each square represents a data point, if a square is missing, then the compound was not tested in that assay. An empty square is an inactive data point, filled squares denote active outcomes.

braries. For data overlap visualization, a pairwise Venn-diagram display was employed. If any set is an outlier, then it can be identified easily on the Venn diagrams. Making the circles of the Venn-diagram scale proportionally to the size of the dataset, made it possible to gauge how large or small the datasets were, visually.

The next step was to identify compounds that show unusually high activity. A standardized pipeline for was developed doing this task, as this needed to be done quickly in the early stage of a project. The workflow with python was as follows:

1. Transform all the assay data from the spreadsheet format in they were available (excel or csv) into Pandas Dataframe objects.
2. Merge all the Dataframes based on the Compound ID to get a list of all unique compounds.

- Using the Compound ID from the unique compound ID list as a key, query each Dataframe to find out if an entry for that exists. If it was not found, then the compound was not tested.
- Use “PUBCHEM_ACTIVITY_OUTCOME” field to determine whether the compound was active or inactive in that assay.
- Doing this for all assays under investigation would generate a fingerprint for each unique compound. 1 was appended to the fingerprint if it is active, 0 if it is inactive, and no action was taken if the entry is not found, meaning the compound was not tested (Figure 2.1).
- If the digits of the fingerprint are counted, it gives how many times the compound was tested. If the digits are added together, that is the total activity count. From this, activity fraction for any compound can be calculated by

$$Activity\ Fraction = \frac{\text{Number of times compound is active}}{\text{Number of times compound is tested}} \quad [2]$$

The program was written in such a way that it could accept a folder as input and all relevant statistics would be calculated automatically and presented in an easily accessible format. This provided a very quick and efficient way of finding frequently active compounds between multiple PubChem assays.

When a histogram of activity fraction for a dataset was plotted, for some specific fractions, there were a large number of compounds found, which led to a large peak at those fractions. Most compounds in a dataset were inactive, so there was a corresponding spike at zero. Also, as all the assay readings were independent, the probability that a compound would be tested n times is less than it being tested $(n-1)$ times. Therefore, there were a large number of

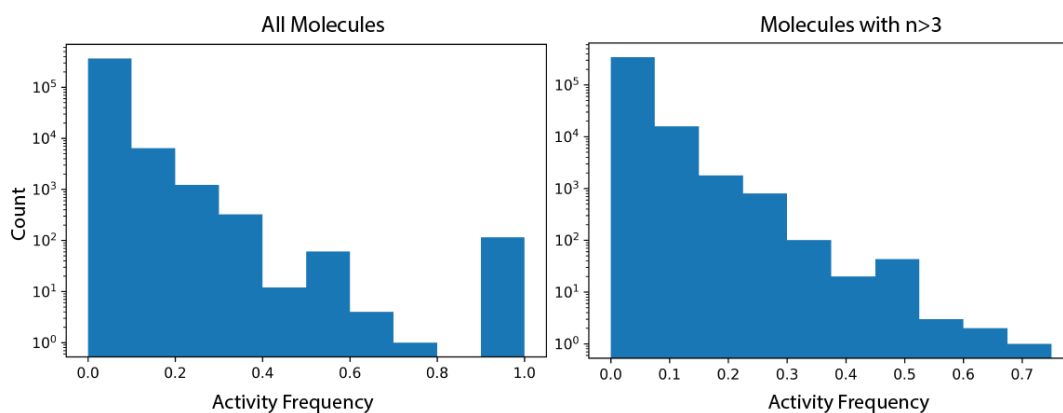


Figure 2.2 : Histogram plot of activity frequency with and without filtering molecules that are tested less than three times. Note that the spikes at 0.5 and 1.0 are diminished after filtering.

compounds that were tested once, and were active that one time. Based on such data, no conclusion could be drawn regarding its promiscuity. So all compounds that had been tested less than three times were excluded. We plotted the histogram after the exclusion, and the peaks, albeit present, diminished considerably.

Once the frequent hitters or false positives from a set were identified, we analyzed them and built the models.

2.3. Tools used in the study:

Throughout our projects, various different tools were employed depending on the task at hand. They will now be discussed briefly.

2.3.1. Programming Language: Python

As mentioned before, we used python for all of our programming needs. We chose Python because it has proven itself to be a workhorse in the field of data analysis. It is also the programming language of choice in machine learning at the moment. Development of packages such as Numpy, Pandas, matplotlib and Tensorflow have led to python being the programming language of choice for data scientists. We shall look at some of the main packages that we used extensively in our study.

1. NumPy⁷³ and SciPy⁷⁴:

NumPy is an open-source python package that adds support for large, multi-dimensional arrays and matrices, and functions to perform operations with such data structures. It is one of the oldest packages in Python and it has provided the backbone for many other packages in the Python data analysis ecosystem. Written in C and Python, NumPy is extremely memory efficient, and therefore, is capable of handling very large multidimensional data.

Scipy builds on the N-dimensional array of NumPy and provides modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. These two modules form the base of all the machine learning packages available in python today.

2. Pandas⁷⁵:

Pandas is an open-source python package that adds support for data analysis and manipulation. The Dataframe object in pandas provides an easy way to work with tabular data. Pandas provides wrapper functions that can read from databases and various file formats such as .csv or .xlsx, directly to a Dataframe object. The Dataframe object can be exported to various file formats and databases. This makes File I/O trivial, and allows for advanced filters while ingesting data. The Dataframe object has methods that mimic many operations found in database query languages such as SQL. This allows for manipulating very large sets of tabular data in python as if we are interacting with a database, while retaining all the power python provides us with.

Pandas is built on Numpy and is also very efficient in handling large data.

3. Matplotlib⁷⁶:

For graph plotting purposes, we used the matplotlib package from python extensively. Matplotlib supports many different kinds of plots, including the regular ones such as scatter, bar, line and histogram. It also supports Venn-diagrams, which we used for determining data overlap. The pyplot object from the matplotlib package, which handles most of the plotting is extremely versatile, allowing for quick customizations. It also integrates seamlessly with pandas, allowing us to call matplotlib directly on our ingested assay Dataframes. Matplotlib is also extremely fast and efficient. On a single core at 3.5Ghz it takes 4.5 seconds to plot about 0.5 million data points in a scatter plot.

4. RDKit⁷⁷:

RDKit is an open-source library for Cheminformatics and molecular modeling written in C++ and Python. It is the most popular open-source package for working with molecules and provides extensive functionality for working with molecules. Most of the basic molecular functionality is found in module rdkit.Chem. Features of RDKit includes:

- Molecule I/O
- Atom and bond manipulation
- 2D and 3D coordinate generation
- Drawing Molecules
- Substructure Searching
- Chemical Transformations
- Maximum Common Substructure
- Fingerprinting and Molecular Similarity
- Generating Similarity Maps Using Fingerprints
- Descriptor Calculation and visualization
- Chemical Reactions
- Chemical Features and Pharmacophores
- Molecular Fragments

Throughout our projects, we have used RDKit for calculating molecular properties and finger-

prints, visualizing a set of molecules as images and searching for PAINS substructures among other tasks. The RDKit library serves as the bridge between chemistry and informatics, and makes it possible to build cheminformatic machine learning pipelines in Python.

5. SciKit-Learn⁷⁸:

Scikit-learn is a popular machine learning framework for Python. It is famous for being beginner-friendly and exceptionally well documented. Scikit-Learn provides implementations for various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means among others. Most of the library is written in Python, using NumPy and SciPy as a base.

6. Jupyter-Notebook⁷⁹ and PyCharm CE:

Working with the various modules and working with big data requires a solid Integrated Development Environment (IDE). Our IDE of choice was PyCharm Community Edition along with the Jupyter notebook, depending on the situation. For running Machine Learning models and working with very large datasets, working with PyCharm was a better experience. For multi-threaded applications we preferred PyCharm, as interrupting the kernel in Jupyter often proved problematic. On the other hand, Jupyter Notebook is fantastic for exploring and visualizing data. The Jupyter Notebook is a web application that allows the user to create and share documents that contain live code, equations, visualizations and narrative text. It has many features of an IDE such as linting, and can also render markdown, proving an easy way of providing documentation.

In Appendix I of this thesis, a typical python workflow using Jupyter Notebook is presented.

2.3.2. Scaffold-Hunter⁶⁵

For visualization of scaffolds present in a molecule set, Scaffold-Hunter is a really useful program, for creating scaffold-trees. At the base of a scaffold tree are the building blocks of the molecule sets, and the branches represent different additions to the base scaffolds. The final node of the tree represents molecules in the set. This tree is particularly helpful in determining what type of scaffolds are present in a given set and their relative proportions.

Scaffold-Hunter is written in Java. It is suitable for smaller sets. The program is capable of handling a couple of thousand compounds, but larger datasets tend to affect its performance.

2.3.3. LigandScout⁸⁰

LigandScout is a commercial software for analyzing pharmacophores. It can generate structure-based, or ligand-based pharmacophores, and has a really nice, intuitive user interface. LigandScout can also perform molecular docking and extract pharmacophores from docked structures. Once a pharmacophore is identified, the library of molecules can be filtered using the pharmacophore. Because a pharmacophore is not based on the scaffold, but on the 3D

structure of the molecule and its interaction pattern, it can encode shape information better than scaffolds. In our projects, we have used LigandScout for pharmacophore analysis.

2.3.4. AutoDock

AutoDock is a freely available group of software for performing Molecular Docking⁸¹. The AutoDock Tools lets the user pre-process the protein and the ligands for docking. The latest docking program from AutoDock is AutoDock Vina⁸², which provides better performance and accuracy over AutoDock-4. In our project with Luciferase, we used python to build a docking pipeline for docking over 300,000 molecules. Autodock Vina was used as the docking program.

2.4. Machine Learning

2.4.1. OCHEM: Online Chemical Modelling Database

Machine Learning was used extensively in this research. In the preceding section, the more popular machine learning methods were contrasted. Subsequently, the respective implementations shall be discussed: For Machine Learning, the freely available platform OCHEM was utilized^{69, 83}. In this section, the various machine learning methods offered by OCHEM are examined.

OCHEM is a free online service that offers a vast array of tools for working with chemical data. As of late 2019, OCHEM contains 2,854,191 records for 637 properties (with at least 50 records) collected from 12957 sources. Anyone can contribute to the data pool, and once pub-

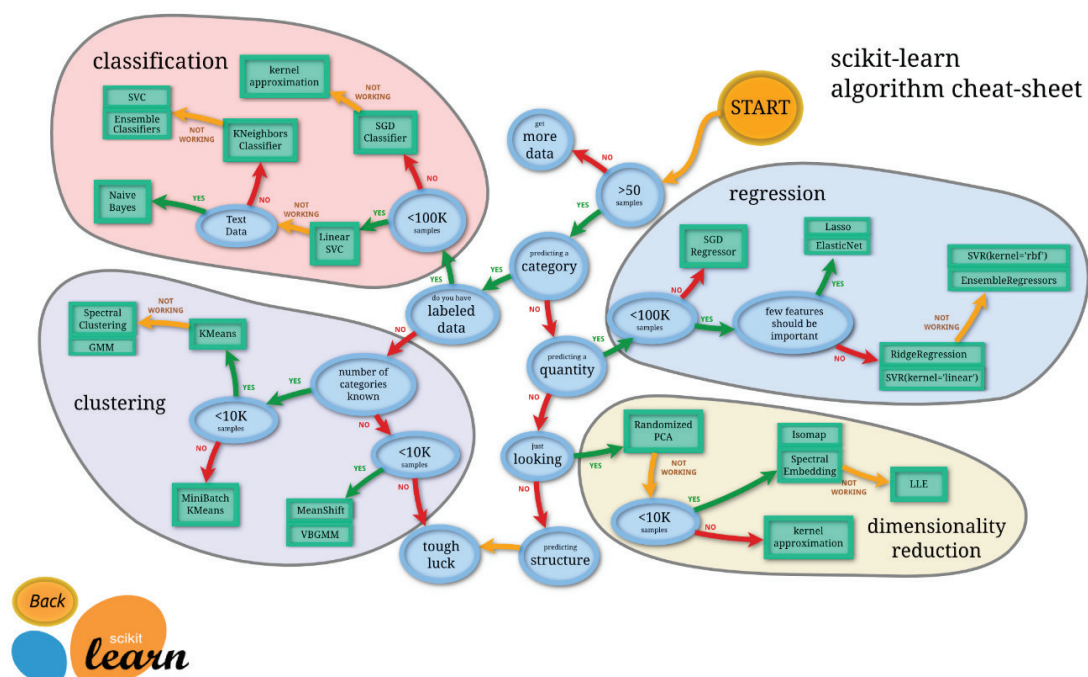


Figure 2.3 : Decision tree style flowchart showing different algorithms available in the Scikit-Learn library, and when best to choose them. Source: Scikit-Learn Website

lished, anyone can access them. Once data is uploaded, OCHEM provides a robust modeling workflow for building machine learning models.

OCHEM has support for many of the popular machine learning algorithms. It also can calculate and use most of the molecular descriptors developed thus far. OCHEM allows the user to create many models at once with different algorithms or descriptors. The OCHEM backend facilitates parallel processing of tasks and returns calculation results in an accessible manner. It also has support for GPU based Deep Learning, which has gained popularity in recent years.

OCHEM supports consensus modelling as well. With a consensus model, multiple models with different descriptors and algorithms developed from the same set can be combined, and outputs of each of the member models are averaged to obtain the final output.

2.4.2. Overview of Common ML algorithms

For different scenarios, there are different machine learning algorithms. The widely used python module scikit-learn offers over 25 such algorithms belonging to all the different categories described that are discussed below (Figure 2.3). Apart from the ones shown in the figure, there are numerous Neural Network architectures and deep learning algorithms. OCHEM provides implementations for over 15 machine-learning algorithms. A detailed discussion of all the algorithms is beyond the scope of this thesis, so focus will be directed toward the algorithms employed in this doctoral work.

1. Principal Component Analysis

Principal component analysis (PCA) is a multivariate statistical technique used to emphasize variation and reduce the dimension of data, losing the least amount of information whilst maximising interpretability of the results. It is often used to visualize and explore multidimensional data^{84, 85}. PCA is a dimensionality reduction technique, with which one attempts to project a higher-dimensional space into a lower-dimensional space with a minimal loss of information: From PCA, key groups of the data can be identified. Such groups are known as the principal components. Each axis in PCA is chosen in a way to maximize variance, so as to explain as much of the original data ordering as possible; the resultant axes are a product of Singular Value Decomposition of the higher-dimensional space. Principal components thus identified can help group the data and identify patterns. PCA has widespread use in object recognition, computer vision, data compression, and more.

2. Linear Regression

The general goal of supervised machine learning is to map a set of predictor variables to a response variable by identifying any relation between the inputs and the outputs. Linear regression can model the relationship between two or more predictor variables and a response variable by fitting a linear equation to observed data⁸⁶. Multiple linear regression with predictor variables can be written as

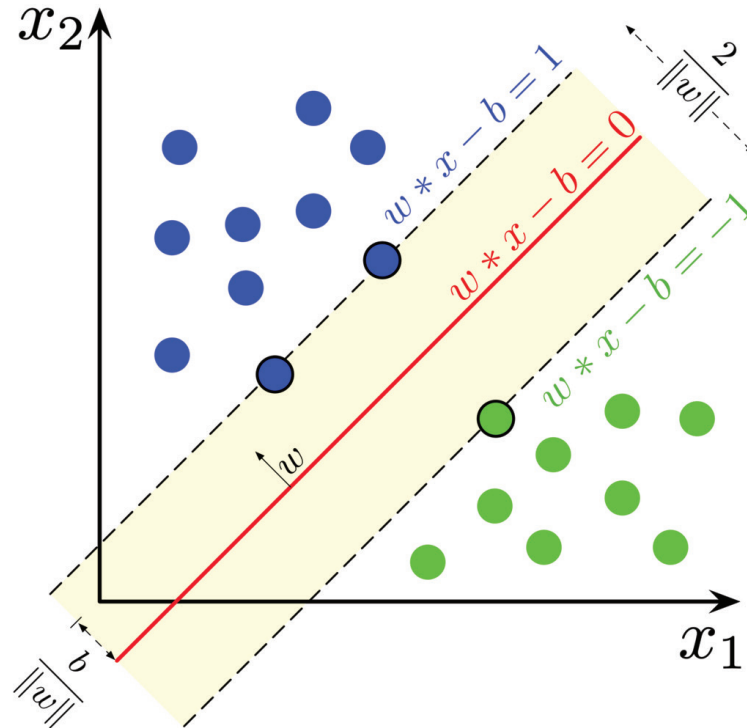


Figure 2.4 : Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Source: Wikipedia

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \epsilon \quad [3]$$

where the ϵ denotes the error term of the model. The goal of linear regression is to minimize the error. In practice, the least squares algorithm is often used to find a solution⁸⁷.

Fast Stagewise Multiple Linear Regression (FSMLR) is a variation of the linear regression algorithm. FSMLR⁸⁸ constructs linear regression models by using greedy descriptor selection. It is a specialized kind of regression boosting that uses a three-set approach. FSMLR uses three different internal subsets for learning: A training set, an internal tuning, and a validation set. The internal set is used for determining the optimal number of descriptors considered in the model. Then, an iterative descriptor selection process, and the corresponding model formulation, continues until the minimal prediction error for an internal test set can be achieved.

3. K-Nearest Neighbors

K-Nearest Neighbors or KNN is a non-parametric, instance-based algorithm that can be implemented in both a classification and a regression context. For classification, the training phase involves simply storing all the data in the training set with their corresponding labels⁸⁹⁻⁹¹. In the classification phase, the model classifies the new input based on k of its nearest neighbors from the training data, where k is a positive integer. Usually, the value of k is optimized by trials run on the training and validation sets. A general implementation of K-NN would measure the distance between the unknown sample to k of its nearest neighbors in the training data, and then classify the sample based on the majority class among those points.

4. Support Vector Machine

Support Vector Machine (SVM) is a non-probabilistic linear binary classifier, that aims to find an optimal hyperplane separating the input data points to classify them^{92,93}. The data is viewed as an N-dimensional vector, and the goal of the algorithm is to find an (N-1)-dimensional hyperplane that optimally separates the data. Many such hyperplanes may exist. An intuitive choice for the best hyperplane is the one that has the maximum distance or margin from the

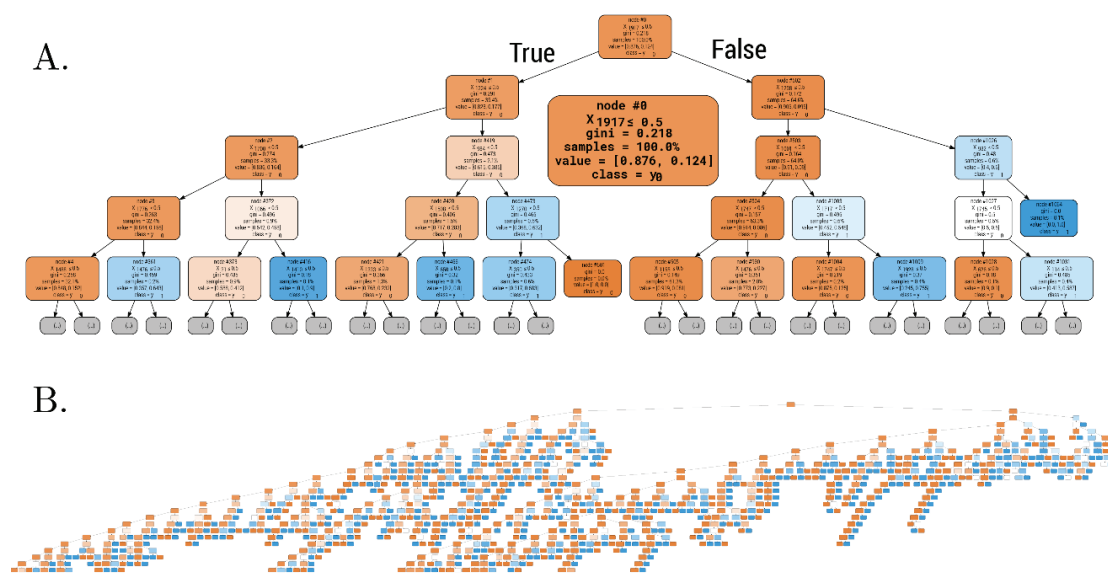


Figure 2.5: A. Graphical representation of a decision tree generated by fitting circular Morgan fingerprint of molecules. For clarity, only four levels are shown. The root node is shown in the middle enlarged. B. The entire decision tree. Colors represent numbers of samples in a node, orange hue denotes a higher number of samples, blue hue represents a fewer number of samples

data.

Support vector machines are effective in high dimensional spaces, which is useful in cheminformatics⁹⁴⁻⁹⁶, as the descriptor space is often highly multidimensional. SVM is memory efficient as it uses a subset of the training data samples known as support vectors in the decision function. The kernel function provides versatility in the SVM methodology, as different kernel functions can be used in the decision function to suit different needs⁹⁷⁻⁹⁹.

Most machine learning packages support SVM. Scikit-learn has a built-in class for SVM. The most popular package for SVM is LibSVM^{100,101}. It was developed at the National Taiwan University and written in C++. LibSVM implements the Sequential Minimal Optimization (SMO) algorithm for kernelized Support Vector Machines (SVMs) and supports classification and regression.

LSSVM¹⁰² is a version of SVM where the minimization problem is solved using a set of linear equations as opposed to convex quadratic programming. Solutions to such linear systems can

be obtained incrementally, and can be parallelized. Therefore, it is possible to take advantage of the massively parallelized architecture of a Graphical Processing Unit to make the model training potentially orders of magnitude faster.

5. Decision Tree

A decision tree is a flowchart of test events or decisions, organized in a tree-like structure^{103, 104}. Each node of the tree denotes a test, or branching condition. Each branch originating from the node represents an outcome of that test. As one traverses the tree, more and more conditional logic is applied to the input, until it (eventually) reaches a terminal node: The terminal nodes, known as the leaf nodes, contain a prediction value or class label.

There are three factors involved in generating a learning decision tree from a feature set: Selection of features, selection of conditions, and the termination condition. The feature that best classifies the training data is used to split data into two branches, and the process is repeated, until the termination condition is met. After a tree is generated, unnecessary branches need to be trimmed down for the tree to be more efficient, this is known as pruning. One popular algorithm for generating such trees is known as CART, or Classification and Regression Trees^{105, 106}, which uses the Gini index as the metric for classification. There is also the Iterative Dichotomiser 3 or ID3 algorithm¹⁰⁷, which uses information gain, or Entropy, as a metric function. The successor of ID3, the C4.5 algorithm¹⁰⁸ is also popular¹⁰⁹⁻¹¹² for building decision trees.

6. Random forest¹¹³

Random forest is a very popular ensemble learning algorithm¹¹⁴⁻¹¹⁷. The forest is made out of a collection of decision trees. To classify a new sample, the input vector is passed down each tree. Then the responses from each tree are collected, and the majority vote determines the final class. Therefore, this method relies on a group consensus, and so it is called an ensemble learning algorithm.

Random Forest usually has excellent performance, a low memory overhead, and scales very well with large datasets. It can also be parallelized to an extent since all the decision trees in the forest can be processed in parallel.

7. XGBoost: Scalable and Flexible Gradient Boosting

Gradient Boosting^{118, 119} refers to another ensemble-based machine learning technique. Boosting is an ensemble-meta algorithm that attempts to improve prediction gradually, by training a sequence of weak learners. The weak learners are grown sequentially, with each iteration improving upon the previous iteration. In gradient boosting, in each iteration, gradient descent is used to minimize the loss function, and the newly optimized learner is combined with the learner from the previous iteration. In contrast to Random Forest, which uses fully grown decision trees to reduce variance, and uses Bagging (see below), gradient boosting is based on weak learners, and uses sequential training of the weak learners. Random forest attempts to reduce the error in the model by reducing variance, whereas gradient boosting reduces error mainly by

reducing bias.

XGBoost¹²⁰ is a package that is developed under the gradient boosting framework to be fast, scalable and portable. First introduced in 2016, it quickly became a very popular choice for building models to solve a wide range of applications¹²¹⁻¹²³. XGBoost implements a decision tree ensemble with additive training. It provides a way to control model complexity through regularization terms that most other tree-based learning packages lack. XGBoost stores the data in in-memory units, called blocks. This reduces sorting time, which is typically the most time-consuming part of tree learning.

8. Artificial Neural Network¹²⁴

In all the projects described in this thesis, a variation of the Artificial Neural Network algorithm called the Associative Neural Network contributed the best performing models. In this section the basics of the Artificial Neural Network are examined in greater detail, as they have been employed heavily in this work.

Artificial Neural Networks are machine learning systems inspired by biological neural networks. The emergent complexity of an animal brain made out of relatively simple neurons is emulated in such systems. A network of relatively simple artificial neurons can learn complex patterns by considering examples, just like in primitive animal brains.

1. Single Artificial Neuron

The functional unit in a neural network is the neuron, also called a node or unit. It receives inputs from some other nodes in the network and computes a singular output. Each incoming input has an associated weight (w), which determines the contribution of that input relative to other inputs. The node applies a function to the weighted sum of its inputs (Figure 2.7), and then forwards the result as an output. The function is known as the activation function of the

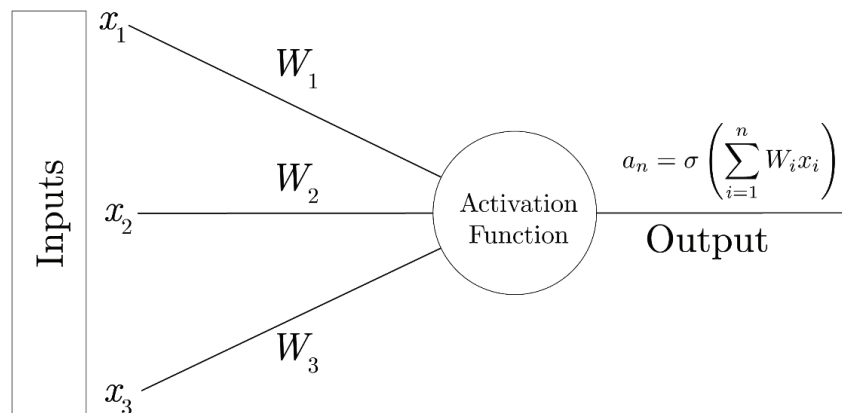


Figure 2.6 : Figurative representation of a single artificial neuron.

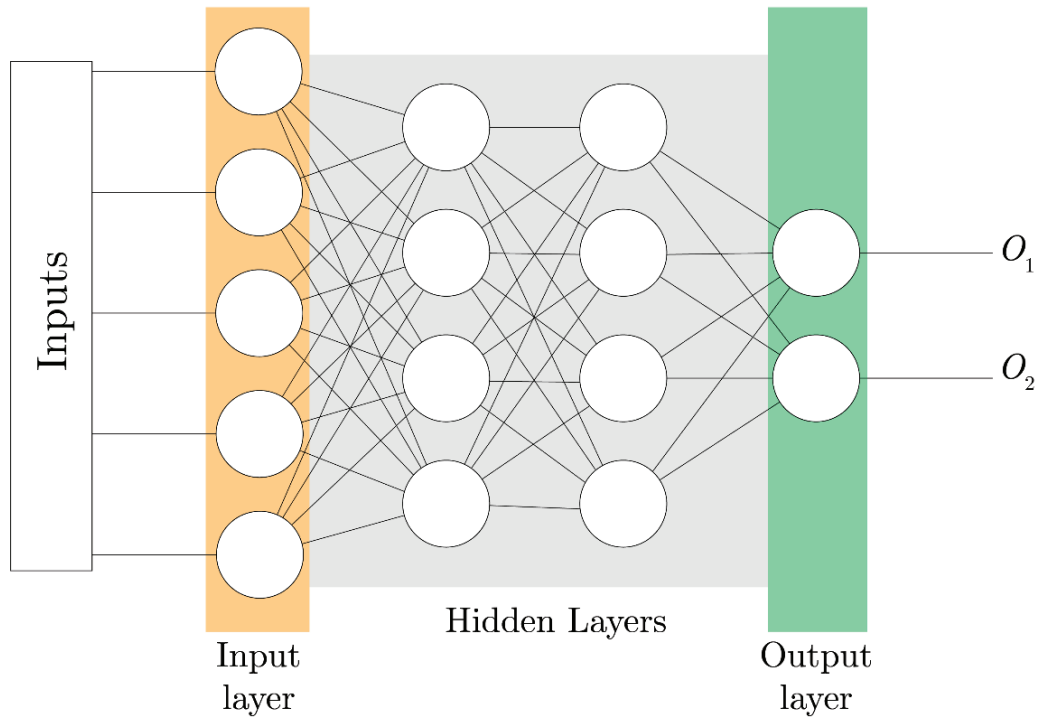


Figure 2.7: Representation of a neural network, or multilayer perceptron showing inputs, outputs and the different layers.

neuron. There are a few activation functions that are commonly used.

Sigmoid:
$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad [4]$$

tanh:
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad [5]$$

ReLU:
$$\text{Relu}(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ x, & \text{otherwise} \end{cases} \quad [6]$$

The neuron performs a weighted sum of all of its inputs. Each neuron also has a bias value, which provides the training algorithm a way to modify the output value of a neuron in addition to modifying the weight parameters. A single neuron with a sigmoid activation function is also known as a perceptron¹²⁵.

2. Multi-Layer Perceptron

A Multi-layer perceptron consists of artificial neurons organized into multiple layers^{126, 127}. The first layer in the network is known as the input layer and the last layer as the output layer. The layers in-between are called hidden layers because they are internal to the network. Given a set of features $X = (x_1, x_2, \dots)$ and a target y , a Multi-Layer Perceptron can learn the relationship between the features and the target, for either classification or regression. The input layer has the same number of neurons as there are features, and the last layer has the same number of

neurons as targets. All neurons in one layer are connected to all or some of the neurons in the next layer. So, outputs from (all) the neurons of the previous layer become the inputs of a neuron in the next layer, multiplied by a weight. These weights, w_i , and bias values, b_i , associated with the neurons are the trainable parameters of a neural network.

3. Training a Neural Network

In order to be able to predict data, neural networks need to be trained with a training set data. The full details of the training algorithm can be found elsewhere^{124, 128-130}, and therefore we will provide a high-level overview. Recall that neural networks contain weights and biases as parameters. The goal of the training is to produce output values that match the target values as closely as possible. To quantify this, one can define a cost function as

$$C(w, b) = \frac{1}{2} \times \|o - t\|^2 \quad [7]$$

where o are output and t are target values. This cost function or error function is the squared Euclidean distance between the observed and target vectors. Such a cost function is also known as the Mean Squared Error (MSE). The goal for training, therefore, is to minimize this scoring function.

1. **Initialization:** The first step is to create the network itself, with all the neurons and appropriate associated activation functions. This is done based according to the architecture of network, which defines the number of neurones in each layer and their connections. Typically, all the weights and bias values are generated randomly, or a normalized initialization approach such as Xavier Initialization¹³¹ can be used.
2. **Feed-Forward step:** Network training starts with the propagation of signals from the input layer to the output using the formula

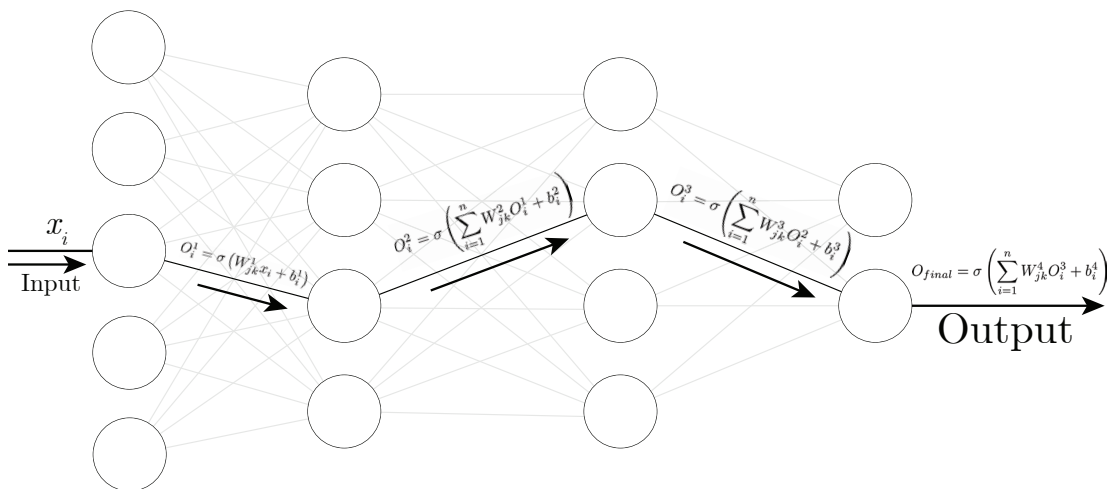


Figure 2.8 : Visual representation of the feed-forward step.

O_final = sigma(w_k^4 * sum_{j=1}^n W_{jk}^4 * O_j^3 + b_i^4)

$$a_n = \sigma \left(\sum_{i=1}^n W_i x_i \right) \quad [8]$$

where a_n is the output of the neuron, n is the number of neurons in the previous layer, W represents the weights, and x denotes input (can be input vector or outputs of the preceding layer of neurons). The output of a neuron is the weighted sum of all of its inputs, subjected to an activation function σ . The signal from the input neurons propagates through all layers of the network until it reaches the output.

3. **Back-Propagation step¹³²:** After the feed-forward step, the error between calculated and target values is calculated. The difference between observed and calculated values is used to adjust the weights and biases of the neural network through gradient descent (See Figure 2.9).

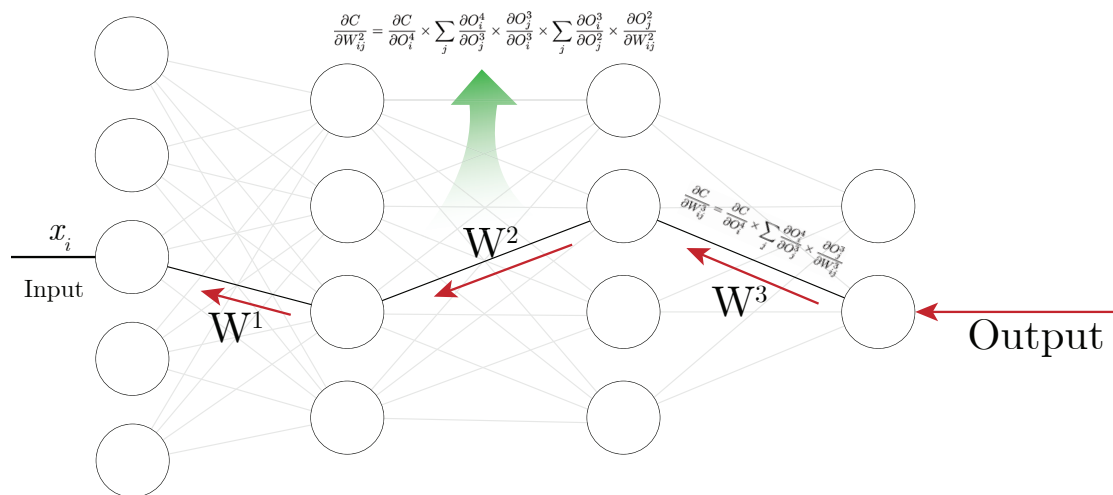


Figure 2.9 : Visual Representation of Back-propagation process. The error from the output nodes is fed backward through the network, to find out its gradient against each of the weight and bias parameters. The equations shown are simplified, and equation for W_1 is omitted for clarity.

The feed-forward and back-propagation steps are done multiple times until network output starts to match the target values. Since a typical ANN has many adjustable parameters, it can fit various data. The important step in neural network training is to prevent so-called overfitting, when neural networks begin to learn specifics about training data points, or begin to map the noise in the training data.. There are different techniques used to avoid overfitting, with one of the most efficient known as early-stopping¹²⁹.

2.4.1. Variations of the ANN Algorithm

1. ASsociative Neural Networks (ASNN)

ASNN^{133,134} is a combination of Neural Network Ensemble and K-Nearest Neighbour techniques. The method is inspired by the thalamo-cortical organization of brain¹³⁵. The organization of an ASNN allows the network to incorporate new data cases in short-term memory implemented as

an ensemble of neural network weights and provides high generalizability without the need to retrain the neural network weights. ASNNs explicitly corrects neural network ensemble biases leading to improved prediction ability over traditional neural networks and k-nearest neighbour techniques. ASNNs have proven to be very useful in the field of cheminformatics and drug discovery^{111, 136-143}, and can also efficiently model highly imbalanced datasets.

2. DNN: Deep Neural Network (GPU)

Deep Neural Networks (DNNs) are neural networks with more than one hidden layer. Each of the hidden layers can have different activation functions and therefore, Deep Neural Networks can model complex, non-linear patterns¹⁴⁴⁻¹⁴⁷. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives. The extra layers enable the composition of features from lower layers, potentially modelling complex data with fewer units than a similarly performing shallow network. With the GPU and CUDA framework, building efficient neural networks that are very complex has become possible¹⁴⁸. OCHEM has an implementation of DNN¹⁴⁹, which was used to build models described in this thesis. OCHEM also has implementations for building Convolutional Neural Fingerprint (CNF)¹⁵⁰⁻¹⁵² models using GPUs, CHEMCHAINER models¹⁵³, and several methods from Deepchem^{154, 155}.

2.4.2. Available Descriptors in OCHEM:

As the OCHEM platform has been used extensively in this doctoral project, it is appropriate to give a review of its available chemical descriptors (or ‘features’) that can be used as inputs to a given machine learning algorithm. OCHEM provides a variety of descriptors, ranging from 0-dimensional up to 3D. Properties calculated by other models can also be used as descriptors. Support for new descriptor packages are being added to OCHEM as they are developed.

1. Adriana.Code156

Adriana.Code comprises a unique combination of topological (2D), spatial (3D) and global molecular descriptors calculated on a sound geometric and physicochemical basis. Adriana offers simple molecular property descriptors such as molecular weight and molecular dipole moment as well as increasingly sophisticated geometric descriptors such as Molecular Radius of Gyration. ADRIANA.Code can perform empirical 3D optimization of the chemical structures or utilize a pre-optimized representation.

2. ALogPS

ALogPS calculates two descriptors provided by the ALOGPS¹³⁷ program, which determine the water/octanol partition coefficient ($\log P_{calc}$), and water solubility coefficient ($\log S_{calc}$)¹⁵⁷. These calculated values can then be used as descriptors for building new models.

3. CDK (3D)

CDK or the Chemistry Development Kit is an open source cheminformatics project¹⁵⁸. CDK

includes a descriptors engine that is capable of performing 2D and 3D molecular descriptor calculations. It supports 204 descriptors divided into 6 blocks: topological, electronic, geometrical, constitutional, hybrid and protein descriptors. CDK also calculates substructure keys including MACCS, PubChem, and E-state keys, as well as molecular fingerprints of 1024 bits based on the Daylight theory. CDK has a standalone program with a GUI and can be integrated into various pipelines, such as Knime. Descriptors in OCHEM are calculated with the recently released 2.0 version of CDK¹⁵⁹.

4. CDDD

CDDD stands for Continuous and Data-Driven Descriptors¹⁶⁰, and comprise descriptors derived from a molecular representation using a pre-trained deep learning model. The model that generates the descriptor uses the ability of deep neural networks to learn a feature representation from low-level encodings of molecules as SMILES strings.

5. ChemAxon Descriptors (3D)

Chemaxon Descriptors are a set of descriptors developed and implemented by the ChemAxon company¹⁶¹. The available descriptors are subdivided into seven categories, namely Elemental Analysis, Charge, Geometry, Partitioning, Protonation, Isomers, and Others. Descriptors that return a Boolean or Numerical value were implemented into OCHEM.

6. Dragon¹⁶² (3D)

Dragon is a well-known software package for the calculation of molecular descriptors, developed by the Milano Chemometrics and QSAR Research Group of Prof. R. Todeschini. It comprises perhaps one of the largest and most comprehensive molecular descriptor libraries available, with a total of 5,270 descriptors. The descriptors are divided into 30 discrete blocks, such as Topological, Constitutional, Drug-like indices, etc. Dragon is built into OCHEM, and it uses version 7 by default. It also supports version 6 for older models.

7. GSFrag¹⁶³

GSFRAG belongs to the category of 2D fragment descriptors. It calculates the occurrence numbers of certain special fragments from $k=2$ to 10 vertices in a molecular graph G , that can be used as molecular descriptors in quantitative structure-property/activity studies.

8. ISIDA descriptors

ISIDA descriptors are part of the ISIDA project, which stands for *In-Silico* Design and data Analysis¹⁶⁴. These fragment-like 2D descriptors are calculated from molecular graphs using three different methods, namely paths, trees, and neighbors. The descriptors are generated from the fragments by using different atom and bond labeling methods¹⁶⁵.

9. Mera and Mersy166 (3D)

Mera and Mersy are two related groups of descriptors. Mera provides a group of descriptors that deal with molecular area and surface. Mersy is abbreviated as Mera Symmetry, and the descriptors are calculated using 3D representations of molecules in the framework of the MERA algorithm.

10. MORDRED

MORDRED¹⁶⁷ is a python-based open source descriptor package that can calculate more than 1800 two- and three-dimensional descriptors. The Mordred package is easy to install in any environment, and can be deployed as a webservice. It also has preprocessing built-in to ensure correctness of the descriptors.

11. Spectrophores

Spectrophores^{168, 169} are 1D descriptors that encode property fields surrounding the molecules. The algorithm provides a fast method for calculating quantum-mechanical descriptors. The descriptors include atomic charges, Fukui functions, hardness and softness among other related descriptors. Spectrophores provide a chemical-class-independent descriptor that can be used to build models.

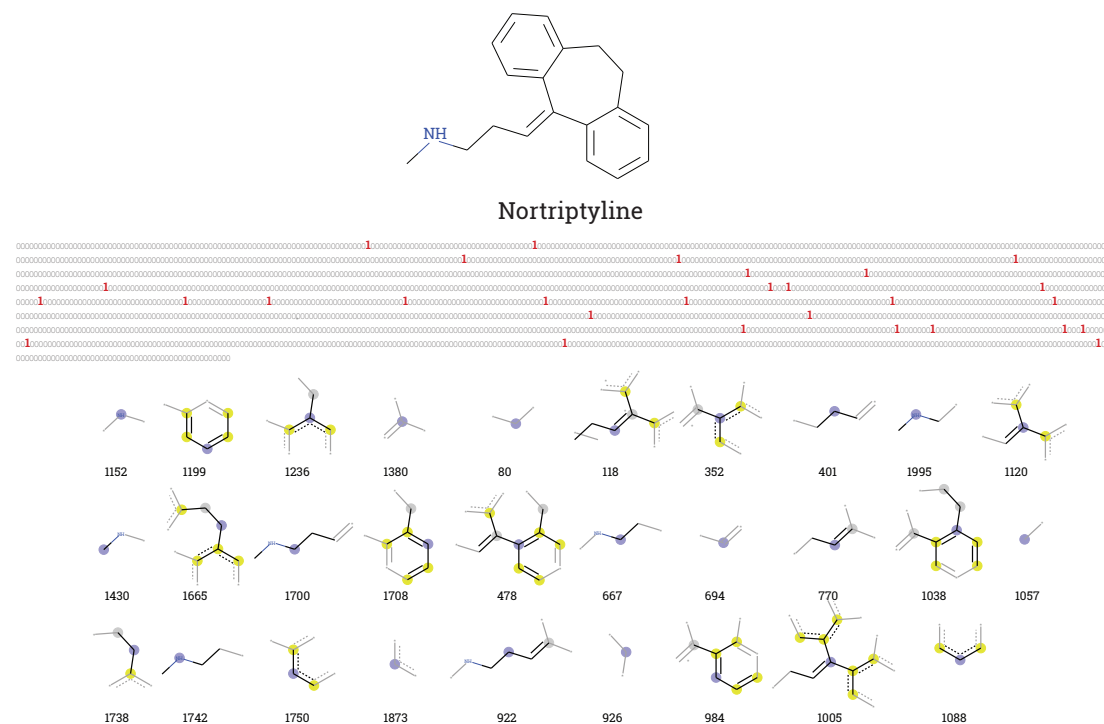


Figure 2.10: Circular Morgan Fingerprint (2048 bit) of a compound. The fragments responsible for setting various bits is shown below the fingerprint.

QNPR

QNPR or Quantitative Name Property Relationship are 1D descriptors that are directly based on the IUPAC names or SMILES representation of the molecules. The descriptors are calculated by splitting the respective string into all possible continuous substrings¹⁷⁰.

12. ToxAlert's 56 Extended Functional Group (EFG) 171 category

ToxAlert is a descriptor based on classification initially provided by the CheckMol software package¹⁷². The coverage was extended to include new groups, particularly heterocycles¹⁷¹. ToxAlert covers a total of 583 functional groups.

13. alvaDesc (3D)

alvaDesc is a new descriptor package from AlvaScience. It can calculate 5305 descriptors divided into 30 blocks. It includes ETA and Atom-type E-state indices together with functional groups and fragment counts. Additionally, alvaDesc implements an extensive number of 3-dimensional descriptors such as 3D-autocorrelation, Weighted Holistic Invariant Molecular descriptors (WHIM) and GETAWAY. alvaDesc provides the calculation of several model-based physicochemical properties such as molar refractivity, topological polar surface area (TPSA), molecular volume estimations, two LogP models (Moriguchi and Ghose-Chippen octanol-water partition coefficient). It calculates 27 drug-like indices including a score for the famous Lipinski's Rule of 5. alvaDesc carries out the calculation of MACCS166 fingerprint, Extended Connectivity Fingerprint and Path Fingerprint and allows the customisation and calculation of the most-used hashed molecular fingerprints. alvaDesc provides the fragments identified during fingerprint calculation as SMARTS strings

14. Inductive Descriptors (3D)

The Inductive Descriptors^{173, 174} are based on calculation of 'inductive' electronegativity, 'inductive' hardness-softness and 'inductive' partial charges. A model for calculating these properties were developed by the group of Prof. Cherkasov, and descriptors calculated using this method has been used to build a wide range of QSAR models, to calculate partial charge and electronegativity of atoms in proteins for docking ligands, amongst other tasks.

15. MAP4 Descriptors

MAP4¹⁷⁵ or MinHashed Atom Pair fingerprint of radius 2, is a fingerprint based on the topological distance between all atom pairs in a circular substructure within a given radius of the molecule. The original implementation sets the radius to be 2, but it can be customized for different purposes. If the radius is 2, the fingerprint is known as MAP4, which is used in this study.

16. SIRMS

SIRMS stands for Simplex Representation of Molecular Structure¹⁷⁶. A simplex is defined as a tetra-atomic fragment with fixed topology and stereo configuration. SIRMS descriptors are

the number of identical simplexes in a molecule. There are two implementations of SIRMS; OCHEM supports the open-source version that calculates 2D simplexes.

17. PyDescriptor (3D)

PyDescriptor¹⁷⁷ uses Python to calculate over 16000 molecular descriptors. This tool was developed by Vijay H. Masand and Vesna Rastija to be used as a PyMol plugin, however it has been implemented to work natively within OCHEM, as used in this work. PyDescriptor has Constitutional, Geometric, Circular fingerprint, Quantum Chemical and Topological descriptors. A complete list of all the descriptors can be found in the published report¹⁷⁷.

18. RDKit

The python distribution of the RDKit package was referenced in the beginning of this chapter⁷⁷. It can calculate molecular descriptors and hashed molecular fingerprints such as Circular Morgan Fingerprint. OCHEM has support for a selection of RDKit descriptors for model building.

19. JLogP

JLogP is an improved logP predictor trained using predicted data. OCHEM already has the ALogPS descriptors, which uses another model to calculate the same property. Recently, the JLogP model has been published¹⁷⁸, and support for that is available in OCHEM.

2.1. Model Training in OCHEM

Having looked at the various machine learning methods and descriptors OCHEM supports, the process of building a machine learning model on the platform must be defined. For the user, the workflow has two steps: Data Upload, and Model Training.

2.1.1. Data Upload

OCHEM also is a database of molecular property records. As of late 2019, OCHEM contains 2,854,192 records for 638 properties with at least 50 records, collected from 12958 sources. The first step when working with OCHEM is to import data to the OCHEM database.

OCHEM has a batch data upload feature that allows the user to upload all data at once from a spreadsheet. The format of the spreadsheet is specified on the upload page. Typically, Python was used to prepare and export such a spreadsheet for this work, primarily due to the powerful manipulation and export features offered by the Pandas package. Once uploaded, the spreadsheet was analyzed by OCHEM and the molecules, typically provided as SMILES strings in one column along with their properties, were processed and added to the OCHEM database. This step also looks for internal and external duplicates in the data, and removes any internal duplicates whilst warning the user about any external duplicates. The data is then represented as 'baskets' in OCHEM.

2.1.2. Building a model

Once the data is uploaded to OCHEM, the model-building process can be initiated. In OCHEM there are six steps to start model training. These steps are presented in Figure 2.11 and we will discuss these steps briefly below.

1. Select sets and learning methods:

The first step is to specify the data and method from which to build the model. We have to select a training set, and we can optionally add one or more validation sets. If validation sets are specified, OCHEM will automatically perform external validation after the model is trained.

Next, we specify the method or the algorithm for our model. One can make a selection from the various machine learning methods OCHEM supports, as previously discussed. Additionally, a validation method is selected for the model. Different methods for performing model validation are discussed in an upcoming section.

2. Data Pre-processing

Next, data pre-processing options are selected: Before any descriptors are calculated, the input molecules must be standardized, neutralized if any charges are present, and any counter-ions and salts should be removed. In OCHEM this is done using the ChemAxon software. By default, OCHEM performs these actions, but if for some reason we do not wish for them to be performed, we can deselect them.

3. Descriptor Selection

The subsequent step is to select descriptors for our model. One or more descriptors can be selected from the wide range available on OCHEM. Additionally, specific blocks of a descriptor can be chosen, and if applicable one can tune any parameters for that descriptor. For example, one could specify the fragment length for ISIDA fragment descriptors, or one could choose just the 2D descriptors from Dragon, for building a model that does not require calculating 3D structures information. The user can upload and use their own descriptors if they so choose. This allows for a flexible approach towards model building.

4. Descriptor Filtering

Descriptors are only useful in Machine Learning if they are different for different molecules, and thereby contain data specific to each molecule. If all the values for a descriptor are the same for all the molecules in a set, then that descriptor is of no use, and should be removed. OCHEM has a variety of methods for descriptor selection, some of which are set by default. OCHEM removes any descriptor with less than 2 unique values, and values that are very large. OCHEM sets a variance threshold of 0.01 for a descriptor, and any descriptor with a variance below this threshold is removed. These are set by default, but the user can customize the value and variance thresholds. The user can also use unsupervised forward selection for eliminating descriptors that have pair-wise Pearson's correlation coefficients greater than a given threshold.

Additionally, full manual selection of descriptors is also possible.

5. Configure Learning Method

A given machine learning method has many tuneable parameters from which to choose, often called hyperparameters. OCHEM exposes the most important hyperparameters for the chosen machine learning method, and allows the user to adjust them. For example, in SVM, this step allows the user to select from three different algorithms; kernel type, cost, and gamma and epsilon parameters. All of the parameters can be given as a range and step, allowing for gridsearch optimisation. Class weighting is also supported and can be enabled in this step.

6. Start calculation

Now the model is configured, the final step is to name the model and start calculation. OCHEM provides a default name for the model, which is a combination of the basket name, the model method, the descriptors used and the model ID. If it is needed, the model can be given a custom name. A priority can also be set for the task that determines how it will be treated in the queuing system employed by OCHEM. Once the user presses the red “Start calculation” button, the job is prepared and set for calculation.

2.2. Model evaluation parameters

Once the model is ready, the performance of the model must be quantified. Different statistical parameters are used for this purpose and they vary based on the type of task. In this thesis, all the models described are binary classification models, so model parameters will be examined with respect to classification metrics. Standard statistical parameters used to define model evaluation parameters are outlined below:

True Positive (TP): Data points that are truthfully positive, and predicted to be positive.

True Negative (TN): Data points that are truthfully negative, and predicted to be negative.

False Positive (FP): Data points that are truthfully negative, but predicted to be positive.

False Negative (FN): Data points that are truthfully positive, but predicted to be negative.

In statistical classification and machine learning, these four parameters are often represented in tabular form. Such a table is known as a confusion matrix (Figure 2.12). We will use confusion matrices to evaluate model performances throughout this thesis.

2.2.1. Sensitivity and specificity

Sensitivity and specificity are measures of the ability of the model to correctly detect “positives” and “negatives” respectively. Sensitivity is the percentage of actually positive compounds that are predicted as positive, whereas specificity is the percentage of actually negative compounds that are predicted as negative. Therefore, sensitivity is also known as the true positive rate (TPR) and specificity is also known as true negative rate (TNR). Sensitivity and specificity can

Online Chemical Modeling Environment

Online chemical database
with modeling environment

Welcome, Dear Mr.Ghosh! [My account](#)

Database ▾ Models ▾

model [?](#)
aiming and validation sets, the machine learning method and the validation protocol

Select the training and validation sets

Training set (required): [AlphaScreen-GST-FHs_public_ochem_3.0.csv \[details\]](#)
Add a validation set

The model will predict this property:
AlphaScreen-GST-FHs using unit: CLASS ▾

Skip model configuration and use the predefined settings

Choose the learning method: [?](#)

Suggested modeling methods:

- ASNN: Associative Neural Networks
- CHEMCHAINER: Chainer Chemistry models (GPU)
- CNF - Convolutional Neural Network Fingerprint (GPU)
- Consensus model (based on models developed for the same set)
- DEEPCHEM: several methods from DeepChem (GPU)
- DNN: Deep Neural Network (GPU)
- EAGCNG - Edge Attention based Multi-relational Graph Convolutional Networks
- FSMLR: Fast Stagemwise Multiple Linear Regression
- KNN: k - Nearest Neighbors
- Library model (A local bias correction model based on another ASNN model)
- LibSVM: grid-search parameter optimisation
- LSSVMG: Least Squares Support Vector Machine (GPU)
- MLR: Multiple Linear Regression
- PLS: Partial Least Squares
- RFR: Random Forest regression and classification
- WEKA-J48: Weka C4.5 decision trees, only classification - use with bagging
- WEKA-RF: Random Forest, only classification
- XGBoost: Scalable and Flexible Gradient Boosting

Methods under development:

Model validation
Validation method: N-Fold cross-validation ▾

Select the preferred data preprocessing options

Preprocessing of molecules (Chemaxon) [?](#)

- Standardization [?](#)
- Neutralize
- Remove salts
- Clean structure

1 → 2

Select the molecular descriptors

Recommended descriptor types

- E-state
- ALogPS (2)
- GSFfragment (1138)
- CDK 2.0 descriptors (256/3D)
- Dragon v. 7 (5270/3D)

[\[select all\]](#) [\[select none\]](#)

<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Constitutional descriptors (47) <input checked="" type="checkbox"/> Topological indices (75) <input checked="" type="checkbox"/> Connectivity indices (37) <input checked="" type="checkbox"/> 2D matrix-based descriptors (607) <input checked="" type="checkbox"/> Burden eigenvalues (96) <input checked="" type="checkbox"/> ETA indices (23) <input checked="" type="checkbox"/> Geometrical descriptors (3D, 38) <input checked="" type="checkbox"/> 3D autocorrelations (3D, 80) <input checked="" type="checkbox"/> 3D-MoRSE descriptors (3D, 224) <input checked="" type="checkbox"/> GETAWAY descriptors (3D, 273) <input checked="" type="checkbox"/> Functional group counts (3D, 154) <input checked="" type="checkbox"/> Atom-type E-state indices (172) <input checked="" type="checkbox"/> 2D Atom Pairs (1596) <input checked="" type="checkbox"/> Charge descriptors (3D, 15) <input checked="" type="checkbox"/> Drug-like indices (28) 	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Ring descriptors (32) <input checked="" type="checkbox"/> Walk and path counts (46) <input checked="" type="checkbox"/> Information indices (50) <input checked="" type="checkbox"/> 2D autocorrelations (213) <input checked="" type="checkbox"/> P_VSA-like descriptors (55) <input checked="" type="checkbox"/> Edge adjacency indices (324) <input checked="" type="checkbox"/> 3D matrix-based descriptors (3D, 99) <input checked="" type="checkbox"/> RDF descriptors (3D, 210) <input checked="" type="checkbox"/> WHIM descriptors (3D, 114) <input checked="" type="checkbox"/> Randic molecular profiles (3D, 41) <input checked="" type="checkbox"/> Atom-centred fragments (115) <input checked="" type="checkbox"/> CATS 2D (150) <input checked="" type="checkbox"/> 3D Atom Pairs (3D, 36) <input checked="" type="checkbox"/> Molecular properties (20) <input checked="" type="checkbox"/> CATS 3D (3D, 300)
--	--

- alvaDesc v.1.0.12 (5305/3D)
- ISIDA fragments
- 'Inductive' descriptors (54/3D)
- MERA descriptors (529/3D)
- MERSY descriptors (42/3D) [?](#)
- Chemaxon descriptors (499/3D)
- QNPR
- Spectrophores (144/3D)
- Structural alerts (ToxAlerts)

Predictions by OCHEM's featured models [?](#)

- Ames levenberg
- Toxicity against T. Pyriformis
- ALogPS 3.0
- CYP1A2 Estate+ALogPS
- CYP2C9 Estate+ALogPS
- CYP2C19 Estate+ALogPS
- CYP2D6 Estate+ALogPS
- CYP3A4 Estate+ALogPS
- Pyrolysis point prediction (best Estate)
- Melting Point prediction (best Estate)
- Water solubility model based on logP and Melting Point
- ALOGPS 2.1 logP
- ALOGPS 2.1 logS

Outputs of other OCHEM models

Obsolete/Additional descriptor types

- CDK 1.4.11 descriptors (256/3D)
- OESlate
- Dragon v. 5.4 (1644/3D)
- Dragon v. 5.5 (3224/3D)
- Dragon v. 6 (4885/3D)
- MOPAC 7.1 descriptors (25/3D)

3 ↓ 4

<Back [Next>](#)

Select a tool to optimize molecular structure

- No optimisation
- Optimise with Corina
- Optimise with OpenBabel
- Optimise with OBGEN (part of OpenBabel distribution)
- Optimise with BALLOON

Online chemical database
with modeling environment

Home Database Models

Welcome, Dear Mr.Ghosh! My account Logout

Privacy statement

Create a model
Select the training and validation sets, the machine learning method and the validation protocol

Select filters of descriptors

- Eliminate descriptors with less than 2 unique values
- Delete descriptors that have absolute values larger than 999999
- Delete descriptors that have variance smaller than 0.01
- Group descriptors, that have pair-wise correlations Pearson's correlation coefficient R larger than 0.95
- Use Unsupervised Forward Selection to delete variables using the above value of multiple correlation coefficient R
- After filtering, I want to select necessary descriptors myself (advanced)

<<Back Next>>

Configure ANN method

Training method: SuperSAB

Number of neurons in hidden layer: 3

Learning iterations (learning iterations): 1000

Ensemble: 64

Enable ASNN:

Additional Parameters (separated by comma): PARTITION=3,SELECTION

Experimental Parameters (conditions merging):

<<Back Next>>

Property weighting options

- Disable multi-property learning and develop individual model for each property
- Weighting options :
 - Set weights inversionally proportional to class/property frequencies
 - Normalize weights

Property/class name	Weight
AlphaScreen-GST-FHs	1.0
• No	1.0
• Yes	1.0

<<Back Next>>

Start calculation of the model

Now we are ready to start calculation. Please provide the name for your model:

Model_Name

- Save models

Task priority:

- High priority (please, use for fast tasks only)
- Normal priority
- Low priority (for long tasks)

<<Back Start calculation>> Discard

Figure 2.11 : Model Training Workflow in OCHEM

be calculated as:

$$\text{Sensitivity}(TPR) = \frac{TP}{TP + FN} \quad [9]$$

$$\text{Specificity}(TNR) = \frac{TN}{TN + FP} \quad [10]$$

A model with a TPR of 100% will report all observed positives as positives, but it may also report false positives. On the other hand, a model with a TNR of 100% will report all observed negatives as negatives, but it may also report false negatives. So, if a model indiscriminately classifies all compounds as positives, it will have a 100% true positive rate, and likewise a model that classifies all compounds as negatives will have a 100% TNR. Thus, model performance cannot be effectively judged by either sensitivity or specificity alone.

2.2.2. Accuracy

In classification tasks accuracy is determined as the ratio of correctly classified instances to the total number of classified instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad [11]$$

If the number of positive and negative data points in a set are comparable, such a set is called a balanced set. In a balanced set, accuracy is an acceptable parameter for judging model accuracy. However, if the dataset is highly unbalanced, classifying all the observations to the majority class will produce a high accuracy value. Biological assay datasets are often highly unbalanced, typically containing over 95% inactives and only 5% actives. Therefore, in such cases, using accuracy can lead to misleading results (e.g., for the above example classification of all data as inactives will provide a model with 95% accuracy, whilst not having any meaningful predictive power) and instead a more meaningful metric should be used.

2.2.3. Balanced Accuracy

Balanced accuracy is calculated as a weighted sum of accuracies within each class. It considers both sensitivity and specificity equally, and can be calculated as the average of sensitivity and specificity.

$$\text{Balanced Accuracy (BA) (NER)} = \frac{TPR + TNR}{2} = \frac{1}{2} \times \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \quad [12]$$

As it is an average of sensitivity and specificity, classifying all observations to the majority class will lead to a BA of 50, thus addressing the limitation of the standard accuracy metric.

2.2.4. Matthews Correlation Coefficient

The Matthews Correlation Coefficient¹⁷⁹ (MCC) can be thought of as a correlation coefficient between the observed and predicted binary classifications. It ranges between -1 and $+1$. A coefficient of $+1$ represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction. MCC is given by the following formula:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \quad [13]$$

MCC is a balanced metric, meaning it can be used regardless of class distribution in a dataset.

2.2.5. ROC-AUC

ROC-AUC¹⁸⁰ is the area under the Receiver Operating Characteristic curve (Figure 2.12). A Receiver Operating Characteristic curve or ROC is a graphical plot that describes the variance in the discrimination power of a binary classification model. The curve is created by plotting the models' sensitivity against its false positive rate at various threshold levels. Therefore, The ROC curve describes the sensitivity as a function of false positive rate for a given classification threshold. The area under this curve provides a measure for model performance. ROC-AUC can be used in unbalanced datasets. It ranges from 0 to 1 , with 1 being perfect prediction.

Model Evaluation Methods

After a model is built and trained, its prediction ability must be put to the test. There are a few approaches for doing this, and OCHEM has many implementations, of which the three most popular are considered.

2.2.6. Test set Evaluation

The simplest approach for testing model performance is to check its prediction against completely new molecules that the model has never seen before. This dataset is known as a test set, and based on the source of the data it can be of two types.

a. Internal Test Set: This dataset belongs to the dataset used for training. However, before training, a portion of the data is set aside as the test set. This test set is used to check model performance during model training. An internal test set often has similar characteristics to the training data, and therefore, the model can biasedly report a good performance when evaluated upon it. Another drawback is that by setting part of the data aside, the model has less training data, resulting in less efficient usage of all available data.

b. External Test Set: This is data gathered from a completely different source, that is used as a test set for a model. As it has no relationship with the training data, it is possible that the nature of data be completely different and lie outside the applicability domain of the model being tested, leading to unreliable prediction. On the other hand, being completely separate, such

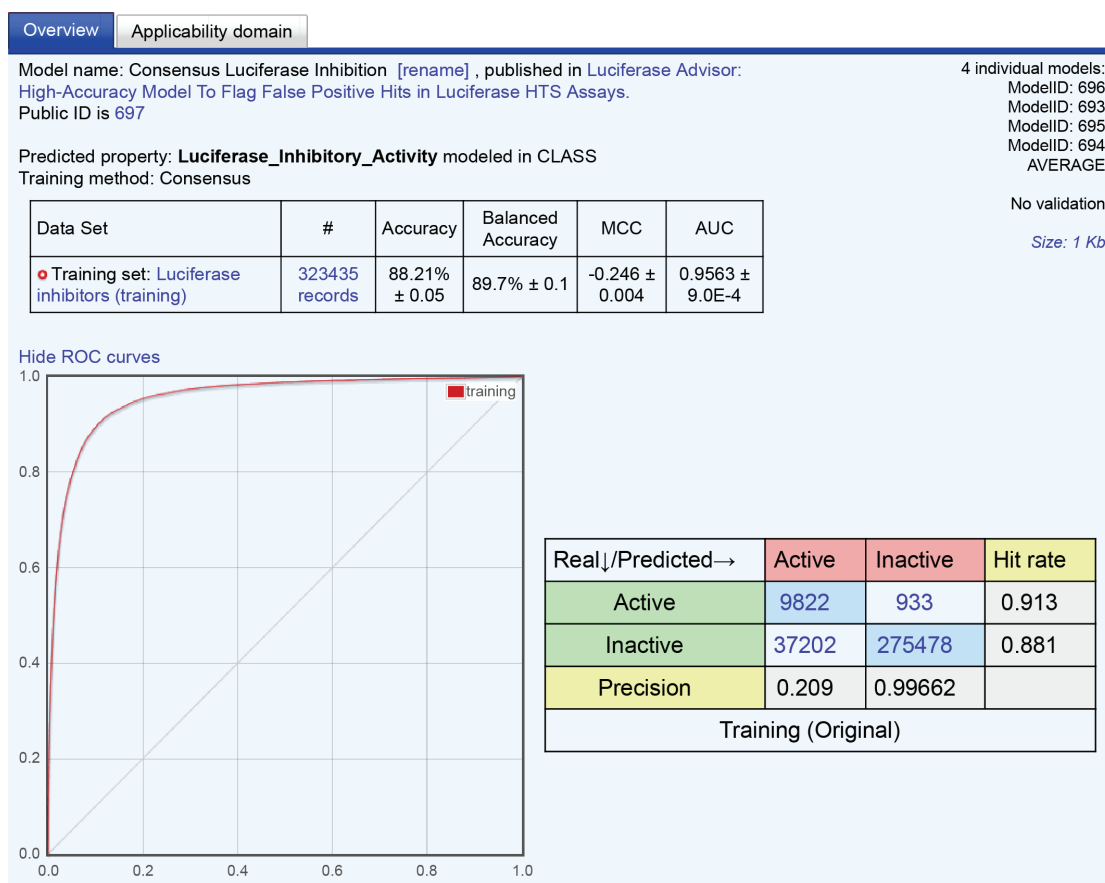


Figure 2.12: Model evaluation parameters, including the ROC curve and confusion matrix, as displayed in OCHEM.

data provides a real-world metric for the performance of the model, without any bias, and gives information about the generalizability of the model.

2.2.7. Cross validation

In the cross-validation (CV) strategy, the data is randomly divided into N folds or bins. The modelling procedure is then repeated N times. During each modelling procedure, a single fold is used as a validation set, and the rest ($N-1$) are combined to form a training set. Each of the N available folds are used once as a validation set and ($N-1$) times as a part of a training set. Performance measures are calculated based on the prediction values from validation folds. In cross-validation the model accuracy over the whole dataset is considered.

A special case of CV is the leave-one-out strategy, where one datapoint is used as a test set, and therefore, N is equal to the total number of datapoints. For very large datasets, this is infeasible, as it requires calculating N number of models, each with size ($N-1$). For the majority of QSAR models, N -fold cross validation is the preferred method. Five-fold cross validation is the default validation method on OCHEM.

2.2.8. Bagging Validation

Bootstrap aggregating¹⁸¹ (bagging) is a variation of machine-learning ensemble meta-algorithm that relies on building multiple classification or regression models and then averaging the results (for regression tasks) or voting on the result (for classification tasks) to obtain the final prediction. The training set for each individual model is obtained by resampling with replacement of the original training set. Given the uniform distribution of the selected training set instances, every resulting training set is likely to have 63.2% of unique instances of the original training set. This means the remaining 37% of instances can be used as a validation set for this run. The set of resampled dataset thus created, is known as a bootstrap sample (Figure 2.13).

If we run the procedure multiple times the chances are high that each molecule in the initial dataset will appear in the validation set at least once. If the molecule appears in multiple validation sets, the prediction results are averaged. OCHEM has implementation of bagging with 64 model ensembles, meaning this process is repeated 64 times. Bagging increases prediction accuracy and model stability, and reduce the risks of over-fitting for random individual estimators.

2.3. Applicability Domain of QSAR Models

As discussed in the preceding section, the performance of a machine learning model is estimated by evaluating the model against a test set, and measuring the performance. The problem with this approach is that chemical data is highly inhomogeneous: The performance of the model on compounds that are dissimilar from the training and validation sets is likely to be different from the estimated accuracy. Moreover, even within the validation set, the accuracy may be inhomogeneous and variable. There can be clusters of compounds that are predicted with an accuracy that is significantly higher (or lower) than the average accuracy. Thus, considering only the average prediction accuracy for an inhomogeneous set does not reflect the complete information on the performance of the model and, therefore, can be misleading. The general definition of Applicability Domain (AD) as formulated by Netzeva *et al.* is “*The applicability domain of a QSAR model is the response and chemical structure space in which the model makes predictions with a given reliability*”¹⁸². Therefore, if one could estimate the prediction accuracies for each and every compound, then a threshold could be defined for the applicability domain. Any compound with a prediction accuracy greater than the given threshold lies within the applicability domain of the model, else it is determined to be outside the applicability domain.

2.3.1. Distance to model

There are many approaches for estimating the prediction accuracy of a compound. In this study, the native OCHEM implementation was employed. All of these approaches rely on the abstract idea of Distance to Model¹⁸³. Distance to Model is any numerical measure of the prediction uncertainty of the model for a given compound. A Distance to Model measure assesses how “far” a given compound is from the model. The compounds that are “further from the model”, have larger values of Distance to Model, and are by definition expected to have lower prediction

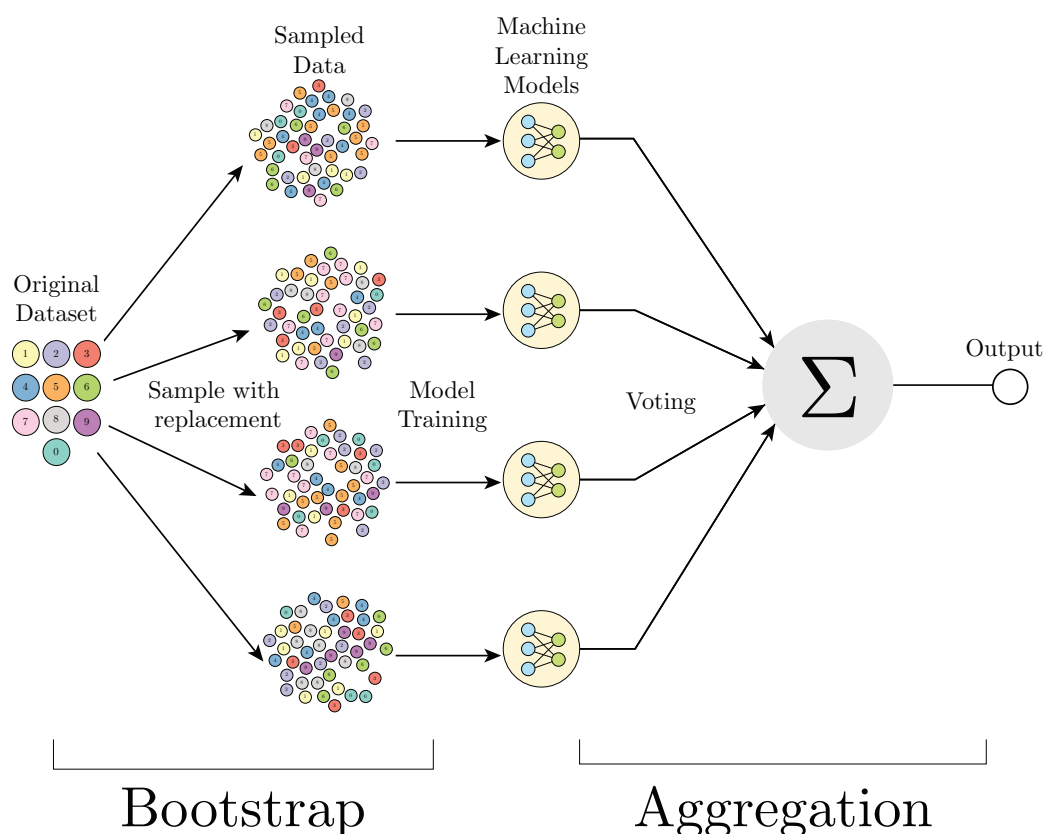


Figure 2.13: Graphical representation of Bagging validation.

accuracy than compounds that have smaller values of Distance to Model. Thus, a distance to model measure provides a way to estimate the prediction accuracy of the compound: This can then be used to assess the applicability domain of a given QSAR model.

It is important to note that Distance to Model estimates the *reliability* of predictions. While accuracy is an objective measure that has a rigid calculation procedure, reliability is subjective and can be estimated in numerous ways. Here, a brief overview is given of the Distance to Model approaches implemented in OCHEM. A comprehensive overview of the methods can be found in the thesis work where such methods were first introduced¹⁸⁴ for classification problems.

1. Standard deviation of the ensemble predictions (STD)

The standard deviation of the predictions obtained from an ensemble of models can be used as an estimator of model uncertainty. The general idea is that if different models yield significantly different predictions for a particular compound, then the prediction for this compound is more likely to be unreliable. The sample standard deviation can be used as an estimator of model uncertainty.

Assuming that \hat{y}_j is a set of predictions for a compound J given by a set of N trained models, the

corresponding distance to model (STD) is:

$$\sqrt{\frac{\sum (y_i(J) - \bar{y})^2}{N-1}} \quad [14]$$

2. Correlation of prediction vectors (CORREL)

This measure is based on the correlation of vectors of an ensemble's predictions for the target compound against other compounds from the training set. Similar to the STD, this measure is applicable only for ensembles of models. More precisely, the CORREL measure for the target compound J is calculated according to the following expression:

$$d_{correl}(J) = 1 - \max_{i=1..N} [corr(\overline{y(T_i)}, \overline{y(T_j)})] \quad [15]$$

3. Rounding effect (CLASS-LAG)

The CLASS-LAG is a measure of the prediction uncertainty in binary classification. Since the mean values of predictions are numeric, it must be rounded to the nearest label (-1 or +1) to identify the class of compound. The less rounding required, the more reliable the prediction is expected to be. This assumption is utilized by the CLASS-LAG DM. The absolute value of the difference between the mean prediction value and the nearest label can be used as a DM. In mathematical notation, this can be expressed as follows:

$$d_{CLASS-LAG}(J) = \min(|-1 - y(J)|, |1 - y(J)|) \quad [16]$$

4. Rounding effect and standard deviation combined (STD-PROB)

In STD-PROB, the two sources of uncertainties; uncertainty related to the rounding of predictions, and uncertainty from model ensemble, are combined. A normal distribution of prediction probability is assumed with mean $y(J)$ and a standard deviation corresponding to the STD value described before. e.g. The suggested distance to model STDPROB is calculated as follows:

$$d_{STD-PROB}(J) = \min \begin{cases} \int_0^{+\infty} N(x, y)(J), d_{STD}(J) dx \\ \int_{-\infty}^0 N(x, y)(J), d_{STD}(J) dx \end{cases} \quad [17]$$

2.3.2. Model Performance with Applicability Domain

With the distance to model metric, compounds with high and low estimated prediction accu-

racy can be distinguished. This gives a way to judge model performance for different subsets of predictions. There are several ways of calculating average estimated accuracies from the calculated Distance to Model value.

1. Sliding Window Average

A sliding window size N can be chosen, over which to average the accuracy on N adjacent compounds sorted by some particular distance to model (DM). The resulting value gives a sliding window average (SWA) estimate of prediction accuracy for a middle compound in the window. The window is then shifted by one compound and the averaging is repeated to get the accuracy estimate for the next compound. The sliding window averaging method is useful against noisy data, and provides smoothed dependencies. The plot of sliding window average accuracy vs Distance to Model should have an overall downward trend, because as Distance to Model increases, the average accuracy should decrease.

2. Cumulative Averaging

A different approach to assessing prediction accuracy is cumulative averaging. Here, the accuracy is averaged over all the compounds with DMs less than a particular threshold, that is varied to generate the DM vs cumulative averaging accuracy plot. In this approach, it is more convenient to choose an implicit Distance to Model, that corresponds to a certain percentage of the set of compounds. For example, in this percentage scale, a DM value of 10% would mean that 10% of the compounds from the set have DM values less than this DM value. This cumulative averaging is easily interpretable and very stable against noise. From this plot, it is very easy to pick the top or bottom $x\%$ predictions. The molecules that are predicted poorly can then be reviewed (checked for errors in original source, grouped and analysed to find common patterns and reasons for poor performance), and the model can thus be refined.

Motivation

Development of a New Chemical Entity (NCE) is a time and capital-intensive process, despite technological advancements and better understanding of the biological system. The throughput of new drugs being developed remain quite low, and the cost of developing one has increased over the past decades. To develop just one drug may take upwards of a decade, and an estimated 2.6 billion dollars, which often requires the involvement of governments and large pharmaceutical companies' investment.

The most expensive failure in the entire drug development pipeline is at the final stage — failure at the clinical trials. The earlier a potential toxic or otherwise unwanted compound is ruled out of the pipeline, the less resource it consumes. There are many ways of weeding out such compounds, which we have discussed in the introduction section of this thesis. Computational methods still remain the cheapest, accessible and scalable method for such filtering. However, the major drawback of computational method is their limited applicability domain along with the prevalence of false positives and false negatives.

Recently, machine learning has gained enormous popularity for its applicability to a wide range of non-trivial tasks such as image recognition and Natural Language Processing (NLP). Building good Machine Learning models requires good training data is sufficient quantity and recently, there has been a rapid growth of available biological assay data — both public and private. The central goal of this doctoral thesis is to develop high-performance machine learning models using such data. For our application, we intend to build models capable of identifying frequent hitters and false positive in various assay systems such as AlphaScreen or assays involving Luciferase.

The goal of such models is to flag hit compounds that are likely to be an assay artifact as opposed to a genuine hit. For target validation step, this means for subsequent secondary assays, a lower number of compounds may be sufficient to consider reducing expense and time that can be utilized elsewhere. But eliminating such compounds and 'failing early', it ensures that resources are not wasted on false leads, potentially reducing the cost of developing a novel drug. Though developing an accurate model is time consuming and requires expertise, once trained, the model can be applied with ease and in a scalable fashion. As new training data become available over time, the model may be retrained to increase the effectiveness further.

All models developed and described in this thesis are publicly available and would provide a valuable tool in early stage drug discovery.

Chapter 3

Modelling False Positive Hits in Luciferase HTS Assays¹⁴¹

3.1. Project Introduction

With advances in molecular biology and other areas such as engineering and computation, high-throughput assay formats have become routine and are widely used in early-stage drug discovery today¹⁸⁵. For hit detection a large fraction (~20%)²³ of these assays rely on bioluminescence, which has an excellent signal/noise ratio and a very low background. Such assays primarily rely on the luciferase enzyme, which is naturally found in various organisms across the animal kingdom, such as the firefly (*Photinus* sp.), larvae of certain beetles known as glow worms, and various marine organisms. Among these, the firefly luciferase (FLuc) obtained from fireflies (*Photinus pyralis*) is the most common and widely used variant. The natural substrate for Luciferase is luciferin. The enzyme catalyzes the production of oxyluciferin and light via a Luciferyl-adenylate intermediate, which is detected and measured in the assay.

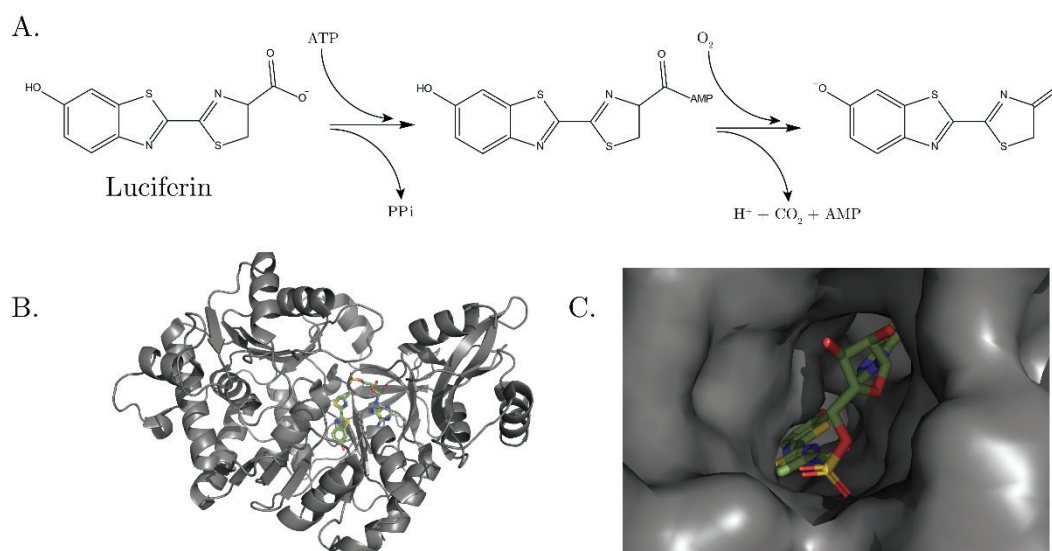


Figure 3.1: A. Chemical reaction catalyzed by luciferase that produces light. B. Crystal structure of the thermostable Japanese Firefly Luciferase complexed with High-energy intermediate analogue. C. Close-up view of the active site of the Japanese Firefly Luciferase complexed with High-energy intermediate analogue.

It has been known for a long time that ligand molecules tested in luciferase-based assays can inhibit the luciferase protein, and thus affect the assay outcome^{21, 186, 187}. For this reason, there has been significant interest towards understanding and evaluating luciferase inhibition, especially in the context of a high-throughput assay¹⁸⁵. In 2008, Auld *et al*¹⁸⁸ published the first comprehensive study in which they tested ~72000 compounds for luciferase inhibition. They also identified important scaffolds for FLuc inhibition¹⁸⁸. In 2012, the same group published a follow-up study where they tested a much larger set of compounds, and identified a few additional scaffolds²³. They also published a crystal structure of benzothiol, an inhibitor bound to FLuc, establishing the binding mode and identifying key interactions²³.

However, despite significant interest and large datasets being publicly available, there has been little to no reported effort towards building a computational model for luciferase inhibitors. Such models could potentially be used to identify and filter out these aberrant and false positive results from a high-throughput assay with good accuracy and relative ease. The goal of our study is to develop a model that can advise against possible luciferase inhibitors present in a HTS dataset. In this study, the publicly available data for building a model, using machine-learning methods, that can identify luciferase inhibitors is analyzed. Further, the influence of molecular shape and geometry in luciferase inhibition is examined.

3.2. Data

All data used in this study are publicly available in PubChem, as summarized in Table 3.1. The activity data were downloaded in a spreadsheet format and structures in SMILES format from PubChem following the Substance ID. The data were then uploaded to the OCHEM platform, which has established workflows for normalizing and managing the data. The data gathered were processed to look for overlap in the compounds tested, which should give an indication of the coverage and reproducibility of the results. Significant dataset overlap was observed (Figure 3.1).

Table 3.1: Summary of the data used in this study including PubChem Assay ID.

set	Concentration used for testing	Number of compounds tested	Number of compounds after excluding inconclusive	% of Actives	Pub-Chem AID [†]	Year
1	50 μ M	72359	70658	2.17	411	2008
1	11.5 μ M	70231	70231	0.72		
2	10 μ M	195634	195634	1.52	1006	2010
3	50 μ M	364105	326367	6.91	588342	2012
3	11.5 μ M	323224	323224	3.25		

[†] AID stands for Assay ID

Set2 is a complete subset of Set3, and Set1 has some unique compounds with respect to Set3. The union of all sets has a size of 375001 compounds. This size of data is good for building models and performing analysis.

In Set1 and 3 there were a few molecules with inconclusive properties. For these molecules, it was not possible to obtain a concentration-response curve, and therefore the activity was uncertain. We excluded these molecules from our analysis, and because of this, Set2 was no longer a complete subset of Set3.

A similar analysis was also performed on the active compounds from the three assays. Here, it was noticed that Set3 has a much larger active compound pool as compared to the others (Figure 3.1B). This is explained by the fact that there is a significant difference between the highest concentrations tested in the respective assays: Set2 was measured at a maximum concentration

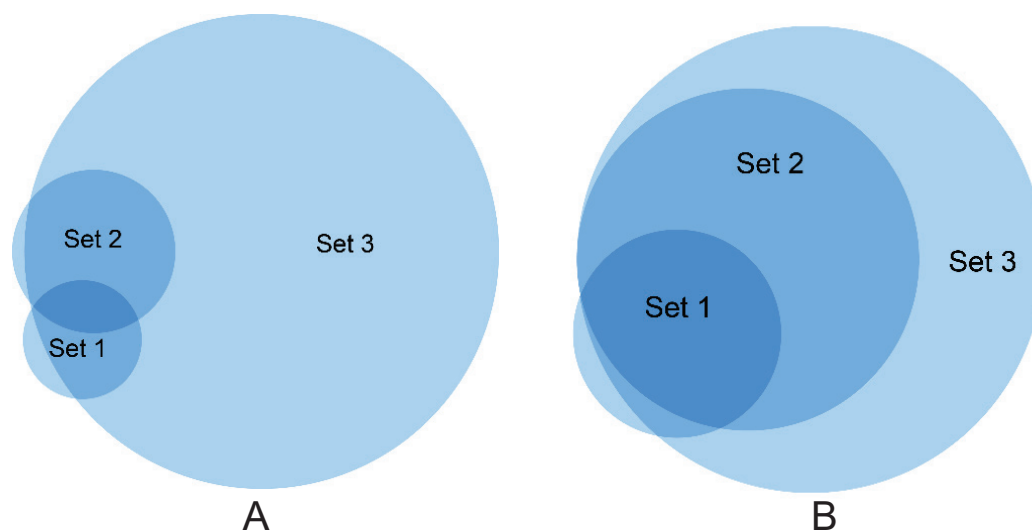


Figure 3.2: Venn diagram representation of the datasets used. The sizes of the circles reflect the relative sizes of the datasets. A. All molecules B. Active molecules

of 10 μM , whereas both Set1 and Set3 were tested at a maximum concentration of 50 μM . Due to the higher concentration, Set1 and Set3 contained larger percentages of active molecules compared to Set2. To compare data at the same concentration for all sets, the inhibition data at 11.5 μM for Set1 and Set3 was extracted and used. For a few molecules, there were no data points available at 11.5 μM , hence they were not considered.

It was found that the more recent assays had a significantly larger percentage of active molecules when compared at the same concentration (Table 3.1). This could be either due to difference in the chemical spaces, or that more recently performed assays are more sensitive due to improvements in assay technology. To assess if chemical space was a contributing factor, the common molecules in all three assays ($N = 61224$) was analyzed. The same increasing trend (0.7%, 1.0% and 2.4% for Set1, 2 and 3 respectively) was observed. Because the chemical space is fixed, this result points to an increase in assay sensitivity. Indeed, to identify potential luciferase inhibition through counterscreening, calibration of the counterscreen assay with known inhibitors is recommended to determine assay sensitivity²³. Because of this problem the different assays cannot be directly compared.

3.3. Methods

3.3.1. Docking studies

For molecular docking, Autodock Vina was used⁸². SMILES of the molecules were downloaded from PubChem, and, using CORINA¹⁸⁹, their optimized 3D structures were obtained. The molecules were prepared for docking using AutoDockTools⁸¹, and were then docked onto the luciferase enzyme with an optimal bounding box enclosing the binding pocket. The binding box was chosen to be large enough to cover the intended docking site, but not too large, in order to minimize calculation time. Default settings were used for the preparation and docking

processes.

The resulting docking poses were analyzed using PYMOL¹⁹⁰. A plane was defined, by choosing three points just outside the binding pocket. This plane denoted the beginning of the binding pocket and for each atom of a ligand, a position vector was calculated with respect to this plane. From this, we calculated which atoms were inside and outside the binding pocket. This information was then averaged over all the docking poses, resulting in the final score that determined how much of a ligand was inside the binding pocket.

3.3.2. Pharmacophore Analysis

Because the crystal structure of luciferase bound to an inhibitor was available, a 3D-structure-based pharmacophore approach to distinguishing between the active and inactive molecules was investigated. The pharmacophore development and screening were performed using LigandScout⁸⁰. The detailed procedure for developing the pharmacophores has been described in the results section.

3.3.3. Machine learning methods.

Using the freely accessible platform On-line Chemical and Modeling Environment (OCHEM)¹⁹¹, more than 150 models were built for all three datasets. Primarily Associative Neural Networks (ASNN)^{134, 192} and Support Vector Machine (LibSVM)¹⁰⁰ algorithms were used for training the models. Associative Neural Network (ASNN) is an ensemble-based method inspired by the function and structure of neural network correlations in brain. The method operates by simulating the short- and long-term memory of neural networks, and is particularly effective with imbalanced data sets (i.e., proportionally many more inactives than actives). These methods on average provided the highest predictive accuracy in comparison to other methods available on the OCHEM website. The methods were used with default parameters as specified on the OCHEM web site.

3.3.4. Molecular descriptors.

A variety of descriptors available within the OCHEM environment were used to train the models. We have discussed them in-depth in the methods section of this thesis (page 37 - 42).

3.3.5. Statistical coefficients.

For detailed description of the statistical coefficients used in this study, please refer to the Methods section of this thesis.

3.4. Results

3.4.1. Molecular Docking

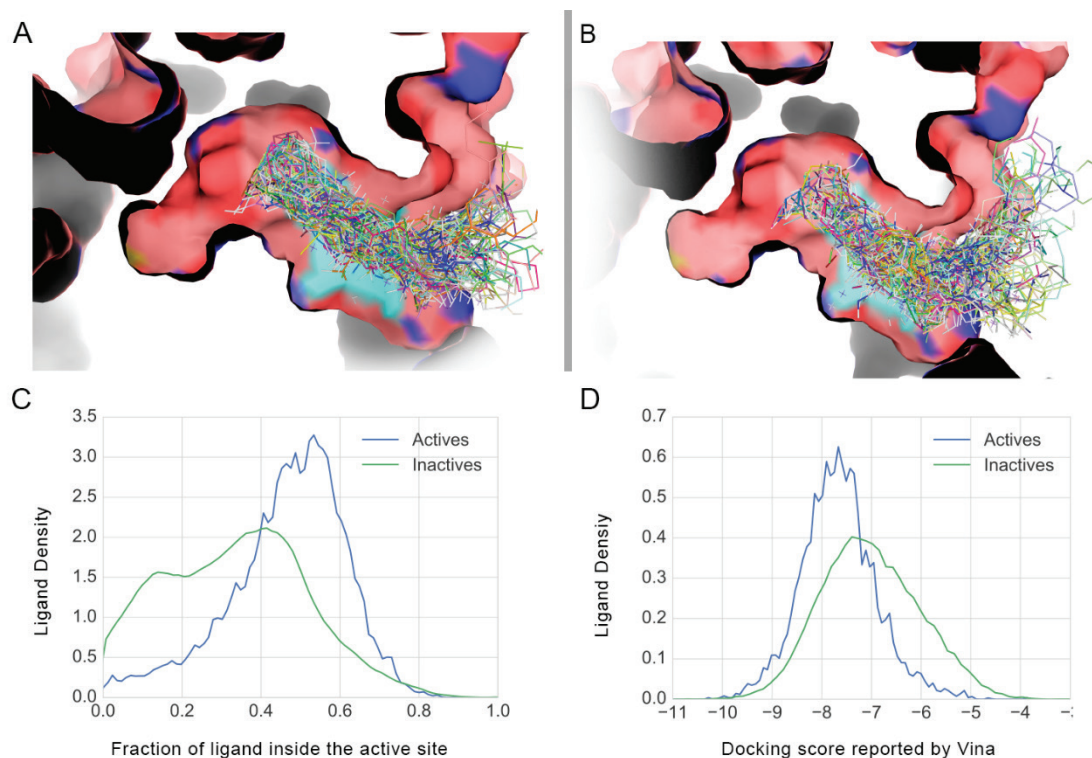


Figure 3.3 : **A.** Density plot of fraction of ligand present inside the active site, for the false positive predictions. Note that the majority of the population lies in between the regular active and inactive molecules. **B.** Density plot of docking score reported by Vina

In an effort to directly visualize the interaction of the ligands with Luciferase, high throughput molecular docking using Autodock Vina was performed. Interestingly, through visual inspection, it was found that there was a positional difference between the docked population **Figure 4.3:** Graphical representation of molecules docked onto Luciferase (top), and a histogram of the fraction of ligand present inside the active site and Vina docking score **A.** Luciferase Inhibitors **B.** Luciferase non-inhibitors. **C.** Density plot of ligands vs. fraction of ligand inside the active site. **D.** Density plot of ligands vs. docking score reported by Vina. Note that the Vina score is not able to distinguish between the inhibitors and the non-inhibitors as effectively.

of the inhibitory and non-inhibitory molecules (Figure 3.3). However, the docking score reported by Vina did not show significant differences between both sets. The optimal score to separate active and inactive compounds (-7.1) using Vina provided a BA of 65.8%. In order to quantify the difference in binding, the percentage of the ligand that was inside the binding pocket was calculated on an atom-by-atom basis, and then averaged over all the ligand poses (Figure 3.3). Doing this allowed the quantification of the positional difference that can be seen in Figure 3.2C, together with a measure of compatibility between the binding pocket and the ligand.

From the distribution, one can see that the inhibitory ligands are docked inside the active site significantly more than the non-inhibitory molecules. Applying a threshold of 0.4 obtained a balanced accuracy of 67.2% in classifying the two groups. Therefore, calculating the fraction of the ligand inside the active site, one can differentiate between the inhibitors and non-inhibitors with an even better accuracy than using the Vina docking score.

3.4.2. Scaffold Analysis

into identify the chemical nature of the active compounds, a scaffold tree analysis was performed using Scaffold Hunter^{65, 193}. This allowed the direct visualization of the structural hierarchy of the active compounds. It was immediately clear that there is a great deal of variation amongst the chemical motifs involved; they are not specific to a chemical subtype (Figure 3.3).

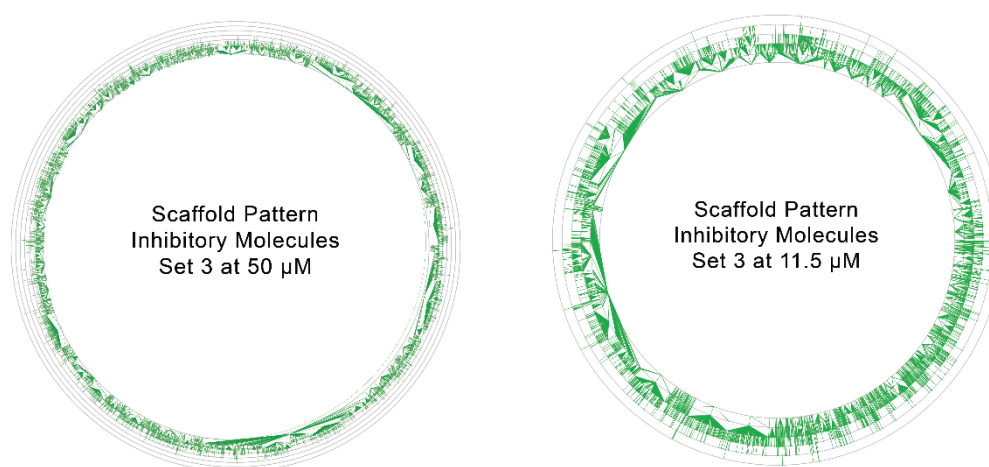


Figure 3.4 : Scaffold tree of Set3 in two different concentrations. The larger size and much higher variability in the chemical space can be clearly seen.

Comparing the scaffold structures of Set3 at 50 and 11.5 μM, it was observed that at the lower concentration, the scaffold hierarchy gets simplified considerably due to the reduced number of active molecules (reduction of about 50%, see Table 3.1). Further, prominent scaffolds emerge.

Upon closer examination, it became apparent that a clear majority of the scaffolds involved, although they belong to different chemical families, have a very flat structure with multiple aromatic rings. Using the SetCompare utility of OCHEM⁶⁸, this observation was quantified, and it was found that such scaffolds are enriched several times in the inhibitor population than in the non-inhibitors (Table 3.2): This implies that the presence of particular functional groups is less important than the overall 3D shape and structure of the molecule, when considered from the perspective of luciferase inhibition. This was also corroborated by reported literature²³, where the addition of a non-planar element, such as cyclohexane or a branched motif, to a pre-established motif drastically reduced inhibition. It should also be noted that all the scaffolds have a very limited coverage, therefore indicating a high variability in the chemical space.

Table 3.2 : Scaffold analysis using OCHEM

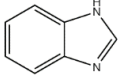
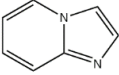
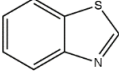
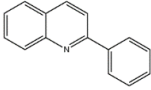
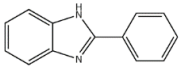
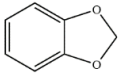
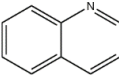
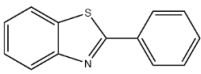
Scaffold Structure	Inhibitors	Non-inhibitors	Enrichment Factor
	6.5	1.8	3.6
	2.2	0.4	5.5
	4.8	1.1	4.3
	1.4	0.2	7.0
	0.8	0.1	8.0
	4.3	2.0	2.2
	3.5	1.6	2.2
	0.9	0.1	9.0

Figure 3.5: Scaffold tree of Set3 in two different concentrations. The larger size and much higher variability in the chemical space can be clearly seen.

In order to take the idea of prominent scaffolds one step further, a filter was constructed using SMARTS to screen active molecules from inactive ones based on the scaffold structure. All the SMARTS were uploaded to ToxAlerts⁵⁶ on the OCHEM platform, and can be accessed there online. As can be seen from Table 3.3, even with a general scaffold such as Benzo-imidazole, only *ca.* 21% of the actives were captured, along with 13% of the inactive molecules. The addition of further groups increases selectivity, but reduces coverage significantly. Due to this, the SMARTS query suffers from exclusivity between selectivity or specificity, and creating an effective filter with this approach proved very difficult due to the large chemical space and variability of the set. Although the scope of such a filter is limited, it provided key insights to the governing scaffold structures behind the inhibition process: This was useful in designing and refining the pharmacophore during our pharmacophore analysis.

Table 3.3: Filtering active compounds employing SMARTS.

Scaffolds encoded in SMARTS †	Actives	Inactives	Enrichment Factor
Benzo-imidazole scaffold	21.66	12.93	1.7

Benzyl imidazole scaffold	4.46	1.06	4.2
Biphenyl system with non-aromatic linker	8.85	6.21	1.4
2-(2-(1H-pyrrol-2-yl)ethyl)-1H-benzoimidazole scaffold	0.71	0.07	10.1
6-Phenyl naphthyl scaffold	2.87	0.92	3.1
Biphenyl system with non-ring linker	6.83	4.11	1.7
2-Phenyl benzo-imidazole scaffold	5.97	0.78	7.7
2-(2-(naphthalen-2-yl)ethyl)-1H-pyrrole scaffold	0.25	0.13	1.9

†For representation purposes, scaffolds that the SMARTS query represents have been used. All the SMARTS queries can be found in the TOXALERTS section of the OCHEM platform

3.4.3. Pharmacophore analysis

From the scaffold analysis, it is seen that the inhibitors are not scaffold specific, but depend on the overall 3D structure of the molecule. Therefore, a 3D-structure-based pharmacophore approach to distinguish between the active molecules and the inactives was investigated. Starting with a crystal structure of luciferase bound to a benzothiol inhibitor (PDB ID: 4e5d), and using Ligandscout⁸⁰, identified the key interactions between the ligand and the enzyme. This provided the basis of our pharmacophore, which lacks selectivity, but is moderately specific (Table 3.4). The initial pharmacophore is defined as a combination of three hydrophobic groups and two hydrogen bond acceptors, as can be seen in Table 3.4. The addition of aromatic rings to the pharmacophore increased the selectivity, and further made optional both the hydrogen bond donor to water interactions, and the hydrophobic interactions of the pharmacophore. This significantly increased coverage, but had a negative impact on specificity (Table 3.4). Examining the various scaffolds identified in earlier analysis (Table 3.2), it was found that there are several members of active compounds where two aromatic systems are bound to a linker group.

To cover this possibility during searching, one feature was allowed to be omitted. This made the pharmacophore much more flexible, as it can accommodate a biphenyl, benzyl or benzo-imidazole, and many other scaffolds, as long as the aromatic groups satisfy the geometry criteria. This is the crucial difference between the pharmacophore and the SMARTS query. For example, in the case of the SMARTS filter that was designed to capture biphenyl systems with a non-aromatic linker, the shape information is irrelevant. If, due to the nature of linker, the structure of the ligand becomes non-planar, the SMARTS would still pick it up. On the other hand, in a pharmacophore query, specifying the motifs involved is not allowed; as long as there are

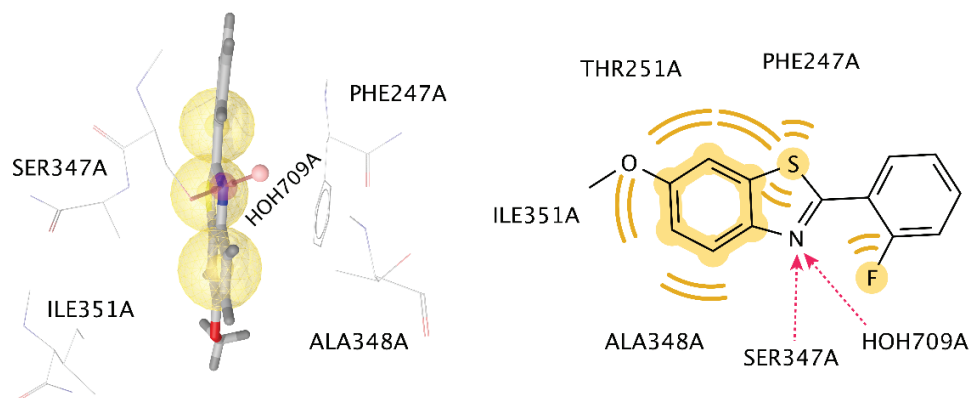


Figure 3.6 : 3D and 2D representation of the interactions of luciferase and benzothiol, its inhibitor (PDB ID 4e5d). The yellow spheres represent hydrophobic interactions, and red arrows show hydrogen bond donor interactions.

two aromatic groups present at the specified 3D position and orientation, it will be picked up. Due to this reason, 74.2% balanced accuracy was achievable with our designed pharmacophore on our current dataset. This resulting accuracy is higher than any approach based on SMARTS analysis and molecular docking explored thus far.

Table 3.4 : Filtering luciferase inhibitors using pharmacophores.

Pharmacophore Representation [†]	Actives (%)	Inactives (%)
	8.2	3.5
	33.5	15.6
	74.2	24.7

[†]Hydrophobic interactions have been shown in yellow, aromatic groups in purple and hydrogen bonds in red. An outlined shape indicates that the feature was marked as optional.

3.4.4. Machine Learning Models

Models were built with various different descriptors that were discussed in the methods section.

Across all three datasets, it was found that Dragon descriptors, along with CDK and Adriana, provided the highest performance. Dragon6 comprised a total of 5270 descriptors. Many of them capture the shape attributes of the molecules well. The same is true for Chemaxon, CDK and Adriana sets, which also have similar types of descriptors in the package. Thus 3D-based descriptors provided the highest accuracy for prediction of inhibitors of luciferase, which indicates the importance of including 3D structural information when modelling luciferase inhibition.

On the other hand, descriptors based on functional groups, such as Structural Alerts⁵⁶, performed poorly throughout. The best results were calculated with the ISIDA descriptors, which provide a comprehensive coverage of different molecular types with automatically generated descriptors. The 2D E-state indices resulted in the second-best models, with a performance that was not statistically different from the performance of models based on ISIDA descriptors.

Table 3.5: Associative Neural Network analysis

Descriptor	Balanced Accuracy † (%)		
	Set1	Set2	Set3
Dragon6 (3D)	83.7 ± 0.8*	83.6 ± 0.3*	88.1 ± 0.1*
CDK (3D)	83.5 ± 0.9*	84.3 ± 0.3*	88.0 ± 0.1*
ISIDA fragments	81.3 ± 0.8	82.7 ± 0.4*	87.7 ± 0.1*
Adriana (3D)	85.1 ± 0.8*	83.4 ± 0.3*	86.7 ± 0.2*
ALogPS, OEstate	81.3 ± 0.9	81.5 ± 0.3	86.6 ± 0.2
GSFrag	79.5 ± 0.9	80.7 ± 0.4	85.8 ± 0.2
QNPR	79.3 ± 0.9	80.2 ± 0.4	85.4 ± 0.2
Chemaxon Descriptors (3D)	81.2 ± 0.8	81.8 ± 0.3	85.3 ± 0.2
SIRMS	78.1 ± 0.9	81.1 ± 0.4	85.3 ± 0.2
Mera, Mersy (3D)	82.1 ± 0.8	81.8 ± 0.4	84.3 ± 0.2
Inductive Descriptors (3D)	78.1 ± 0.9	78.8 ± 0.4	80.7 ± 0.2
Structural Alerts	73.0 ± 1.0	72.7 ± 0.4	79.1 ± 0.2
Spectrophores (3D)	78.1 ± 0.9	77.4 ± 0.4	78.4 ± 0.2
Consensus Model	86.2 ± 0.7	86.4 ± 0.3	89.3 ± 0.1

†Balanced accuracy for all three datasets obtained using various descriptors and Associative Neural Network algorithm sorted by accuracy of models for set3.

*Models that are marked with an asterisk were used to create the consensus model.

1. Consensus Models

Consensus models were built for each dataset. This was performed by averaging the results of the four best-performing models, selected according to the balanced accuracy. As shown in table 3.5, the consensus models had an accuracy *ca.* 1-3% better than the individual models: All further analysis was performed using these consensus models.

2. Analysis across datasets:

To observe the effects of the increasing volume of data in the training sets of the models, as well as to determine the performance of the models against new compounds, the other two sets were used as test sets against each trained model:

Since Set1 is the smallest, and also had the least sensitivity amongst the three datasets, models from this set would not be able to effectively predict molecules from Set2 and Set3. As one can see from Table 3.6, Set1 models show lower accuracy against Set2 or Set3, in comparison to itself. In the case of Set2, the sensitivity is higher and training set size is larger than that of Set1, and therefore the model can effectively predict molecules from Set1. However, against Set3 the same model does not perform well, and this can be explained by the same argument as in the case of Set1. The model built from Set3 provided the best results, as the training set was the largest and also the sensitivity the highest, providing the largest number of active molecules in the training set. This makes Set3 the main dataset from which to build our final model.

Table 3.6 : Cross correlation of models between the datasets used in the study.

		Test Set †		
		Set1	Set2	Set3
Training Set	Set1	86.2 ± 0.7 (70,231)	81.2% ± 0.3 (195,546)	81.0% ± 0.2 (323,224)
	Set2	89.8% ± 0.7 (70,231)	86.4 ± 0.3 (195,546)	85.5% ± 0.2 (323,224)
	Set3	90.8 ± 0.5 (70,231)	87.7 ± 0.2 (195,546)	89.3 ± 0.1 (323,224)

†Number inside the parenthesis denotes the number of tested molecules in the respective set.

3. Analysis of incorrect predictions

In order to gain a better understanding of the inaccuracy of the models, the compounds that were predicted incorrectly were analyzed. First, the molecules that were predicted incorrectly in at least two consensus models were selected. For the FN (actives that are predicted as inactives), were 130 molecules, and for the FP (inactives that are predicted as actives), there were 13594 molecules. To understand the nature of the false positives, they were docked against Lu-

ciferase, and examined as per the analysis described in the docking section. This revealed that FP molecules have the propensity to dock inside the active site of luciferase more than regular inactives (Figure 3.3), but less than that of regular actives. This means that these molecules have some structural features that are capable of fitting inside the active site of luciferase, but the interactions are not favorable. This is well corroborated by the docking score reported by Vina, where the false positives have more favorable binding energy compared to the inactives, but less favorable compared to the actives. The structural features are being recognized by the machine learning algorithms, and because the machine learning methods do not consider the interactions, the molecules are being marked as inhibitors, when in fact due to unfavorable interactions they do not inhibit luciferase.

Since aggregation is known to play a role in inhibition¹⁹⁴, it was decided to investigate whether the activity of some compounds could be due to aggregation. As a property, aggregation is dependent on many variables, and therefore it is very difficult to predict: There has been significant effort in developing this area, and an aggregation advisor (<http://advisor.bkslab.org>)¹⁹⁵ has been established to address this problem. This on-line server checks new molecules against a database of known aggregators; the database contains compounds that are known to aggregate at concentrations of 10 μM or lower. Because at elevated concentration aggregation is promoted further, this test will identify such molecules in our datasets that were screened at 10 and 50 μM .

It was found that 3.2% of the active compounds are known to aggregate, as compared to 2.1% among the inactive molecules. It is also worth mentioning that in Set1 and Set3 assays, 0.01% Tween-20 was used as a detergent, presumably to prevent aggregation. In the case of Set2, compounds were dissolved in DMSO. Therefore, one might expect that in Set2, more aggregators would be present in the active pool. However, due to the small number of aggregator molecules, no appreciable difference in percentages of aggregation for active/inactive in Set1, Set3 vs. Set2 was observed. The use of detergent could decrease the percentage of aggregators amid active molecules. Still, the fraction of aggregators amid active molecules is 50% large than amid non-active ones. Thus, aggregation plays a significant role in making the molecules change class across experiments, and may have played some role in inhibiting luciferase.

4. Effect of concentration

Table 3.7 : Effect of concentration reporting balanced accuracy in consensus models for Set1 and Set3

Set	50 μM	11.5 μM
Set1	85.3 \pm 0.4	86.2 \pm 0.8
Set3	87.2 \pm 0.1	89.3 \pm 0.1

As mentioned previously, there is a concentration difference in the datasets taken for this study, and the models built are dependent on this concentration due to the activity of molecules changing based on concentration. It was noted that at higher concentrations, models became less accurate. To better understand this, the number of molecules (N=2666) were counted that were incorrectly predicted as inactives by the model developed using 50 μM data. It was found that

81% of these molecules became inactive upon lowering concentration. Contrary to that only 54% (N=22303) of correctly predicted active molecules (corresponding to on average a 50% decrease of actives when lowering concentration from 50 to 11.5 μ M) became inactive. Therefore, at higher concentration, such molecules introduce noise into the data, leading to inaccuracy. The models reported here were built using activity data at 10 μ M or 11.5 μ M: This must be taken into consideration when applying the model.

5. Merging datasets to create the final model

To create the final model, Set3 was chosen to be our primary set, as reasoned above.

Only the unique active molecules from Set1 and 2 were then added to it, reasoning that since these molecules are active in an assay with lower sensitivity, they have a higher probability to be active and not false positives. It was decided not to merge the inactives from three datasets together, as doing so would lead to having inactive molecules that come from experiments with lower sensitivity, which may bring false negatives. This gives a merged dataset with N = 323443, and with 3.3% of active molecules. Using the same procedure as previously discussed, a consensus model was obtained, with a balanced accuracy of 89.7%. It can be accessed at <http://ochem.eu/article/104546>.

3.4.5. Sensitivity of existing filters

As the inhibition of luciferase, and the nature of its inhibitors were explored in this study, it was wondered where these identified inhibitors lie in the context of existing frequent hitter and Pan Assay Interference Substance (PAINS³⁷) filters: These filters are implemented on OCHEM as part of the ToxAlerts platform⁵⁶, and they were ran against our dataset. It was found that PAINS filters flagged approximately twice as many active compounds as inactive compounds; the AlphaScreen filters to detect promiscuous compounds also provided an approximate threefold enrichment of flagged actives over inactives. However, the promiscuity filter that was designed to identify compounds likely to hit multiple assays¹⁹⁶, provided a much smaller enrichment. The highest enrichment was calculated for the AlphaScreen filter, however, this filter had the lowest coverage. The most prominent alert among the AlphaScreen filter that picked up luciferase inhibitors was the Amino alert (aminal on a pyridine-based system). This alert picked up several compounds with a planar structure, and provided an enrichment factor of 6.2. It should also be noted that the number of alerts involved in this case is very small, which gets reflected in the poor coverage of this filter. The difference in the number of alerts in each filter contributes to the specificity/selectivity trade-off.

Also of note, most of the compounds were flagged as being reactive, unstable or toxic. This is expected, as the responsible filter is known to pick up drug-like molecules: It is worth mentioning here that the presence of such alerts by itself does not make a molecule toxic in the context of medicinal application, due to dosage and clearance from the body.

Table 3.8: Luciferase inhibitors tested against a variety of other filters.

Compound Filters †	Actives (%)	Inactives (%)	Enrichment
Pan Assay Interference Substance (PAINS) (480)	9.8	4.9	2.0
Promiscuity (178)	4.7	3.8	1.2
AlphaScreen FH filters (25)	1.7	0.6	2.8
Reactive, Unstable, Toxic (340)	66.9	62.3	1.1

†The numbers in parentheses represent the number of alerts in each respective filter.

3.5. Discussion

The developed chemoinformatic model is suitable for providing an early warning against potential inhibitors of luciferase that may interfere with HTS experiments. Since the model does not have 100% accuracy, some compounds can be predicted as luciferase inhibitors when in reality they are not. On the other hand, even if the molecule is indeed a luciferase inhibitor, that does not mean that it cannot be a potential lead. Hence, it is strongly advised not to discard the flagged molecules as false leads but rather to consider them further, to better interpret experimental results.

Thus, the model described here should be used to identify *potential* interference in luciferase-based assay systems. The identified molecules should be re-tested using other assay protocols that do not rely on luciferase. The merit of this study is that one can find potential interference in very large datasets, and only the flagged molecules then need be tested by orthogonal assays. This reduces cost, time and effort in the counterscreening effort.

3.6. Project Conclusions

In this study, various methods of filtering and detecting luciferase inhibitors in a luciferase-based HTS assay were explored. Computational models using machine-learning methods on publicly available data from PubChem were designed. Further, molecular docking was employed to understand how inhibitors bind to luciferase, and a scaffold analysis was performed to gain a better understanding of the chemical nature of such inhibitors. The machine learning models that were developed outperformed other methods of filtering luciferase inhibitors, such as SMARTS or pharmacophore-based filters. A prediction accuracy of 89.7% was obtained, which makes the final model a good tool for filtering potential luciferase inhibitors. Still, the predictions of the model should be considered as advice, and flagged compounds may be re-tested in orthogonal assays. All models and data reported here are publicly accessible at <http://ochem.eu/article/104546>.

Chapter 4

Machine Learning model to filter Frequent Hitters for AlphaScreen assays

4.1. Introduction

AlphaScreen is a very versatile assay technology which is commonly used in drug discovery projects¹⁹⁷. It is particularly suitable for HTS, due to the high signal-to-background ratio, dynamic range and sensitivity, together with the homogenous assay format and reagent stability. The “Alpha” in AlphaScreen stands for Amplified Luminescent Proximity Homogeneous Assay. As the name indicates, the assay relies on the intended biological interaction to bring two beads together; a donor and an acceptor bead. The donor bead contains a photosensitizer, phthalocyanine, which excites ambient oxygen to its singlet state upon irradiation with 680nm light. This excitation is typically done using a laser. If an acceptor bead is present within about 200nm (typical diameter of a single bead particle), then the singlet oxygen interacts with chemical dyes in the acceptor beads resulting in a cascade that ends with emission of light at 520-620nm. Excitation of a single donor bead can produce up to 60000 singlet oxygens per second, which leads to massive amplification of response, if a donor-acceptor interaction is present¹⁹⁸. If no such interactions are present, the singlet oxygens diffuse in the medium and go undetected, producing exceptionally low background noise.

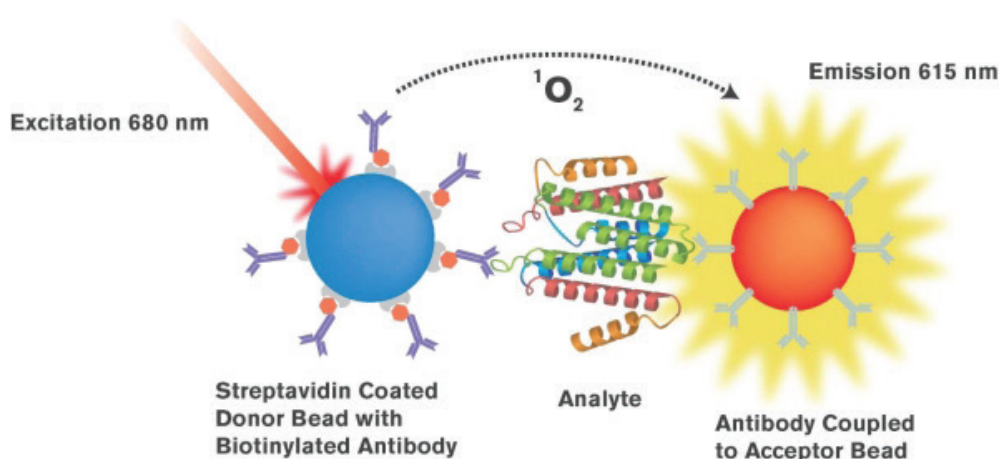


Figure 4.1: Principle of AlphaScreen. Figure obtained from article by Elgen et al¹⁹⁸.

However, there are multiple ways ligand molecules can interfere with various components of AlphaScreen assay technology. Based on the mechanism of action, there are three general categories: These are Singlet oxygen quenchers, color quenchers, or inner filters and light scatterers. Efforts have been made to identify such bad actors, and filters have been published to isolate them in a high throughput screening setting^{19, 37, 199}.

In a previous study¹⁹, researchers from the Tetko group reported two classes of interfering compounds, one that interfered with the interaction of the protein His-tag moiety to nickel chelate (Ni^{2+} -NTA) beads of the AlphaScreen detection system, and another generic class of compounds, that interfered with the assays via unknown mechanisms. In a follow-up study¹⁹⁹ a class of compounds interfering with the interaction of glutathione S-transferase (GST) to glutathione (GSH) was examined, which interfere with AlphaScreen assays involving beads containing Glutathione. Scaffolds were identified that were over-represented among the iden-

tified frequent hitters. Such scaffolds were then encoded using SMARTS²⁰⁰ strings, and the ToxAlerts⁵⁶ platform was used to build a working filter. In this study, machine learning methods to build models we employed using the OCHEM platform¹⁹¹ from the same data, in order to compare the efficacy of machine learning and scaffold-based approaches.

4.2. Data

The *in house* data that were used to create the SMARTS based filter from the previous studies^{19, 199} was already available. This was used as the training set for building the models. A robust test set against which both methods could be tested was needed. For this, publicly available data in PubChem²⁰¹ was utilized. AlphaScreen confirmatory high throughput screenings was sought, and from these were selected 15 HTS campaigns with the highest number of actives (see Table S1 in Appendix I). HTS campaigns with very small number of actives are statistically less useful for identifying frequent hitters.

From previous studies, two *in house* datasets were available corresponding to the two frequent hitters (FH) types identified. However, the identified assays from PubChem used various combinations of donor and acceptor beads (Table S1, Appendix I). In order to be directly comparable, compounds had to be identified that were interfering with either the Ni-NTA beads, or GSH coated beads. However, without counter-screen information, separating compounds based on Mode Of Action (MOA) was not possible. Therefore, the various types of FH identified in the previous study were merged into one category in order to compare them against the generic FH from the PubChem sets. Although an external test set was not available, models were built for each of the different classes, in an effort to provide models that could suggest mechanism of action of FHs. Thus, in total, there were four different datasets (Table 4.1).

Table 4.9: Descriptions of the datasets used in this study. In the PubChem-Combined dataset, the selection threshold was varied, resulting in a variable number of FH.

Data Set abbreviation	Data Source	Total Compounds	Active Compounds	FH	FH specificity
OCHEM-Ni-NTA	<i>in house</i>	24988		77	Streptavidin-Ni-NTA/His
OCHEM-GST	<i>in house</i>	24988		53	GST/GSH
OCHEM-Combined	<i>in house</i>	24988		190	-
PubChem-Combined	PubChem	489951	55560	Variable*	-

*The number of FHs was determined by the used enrichment threshold.

For identifying FH, statistical analysis of compound activities was employed. First, compounds that were tested less than five times were omitted. The activity fractions associated with low-test count were much higher for such compounds and thus such data points would introduce noise

in the analysis. For example, if a compound was tested once and found to be active, then its activity fraction was 1. Such compounds should not be considered a FH since it could be just an active. After filtering, the activity fraction $F_{obs} = n/k_i$ for each compound was calculated, where a compound i was tested in n assays and was active k times. Next, for each assay the activity labels were shuffled, keeping the total number of active compounds unchanged. This means that per compound, the activity was randomized. After this was calculated the activity fraction again. Let this be called F_{calc} . To obtain a statistically significant result, the randomization was repeated and F_{calc} was computed 10,000 times, along with the 95% upper confidence interval $Fav_{(95\%)}$. Then for each activity fraction, an enrichment factor E_n was defined, as the ratio of F_{obs} to $Fav_{(95\%)}$. This enrichment factor was the degree of overrepresentation for that activity fraction, when compared to the one obtained randomly. For example, if for activity fraction 0.7 we calculated $E_n = 10$, then compounds that had an activity fraction of 0.7 were observed in the active group 10 times more than would be expected by chance. Considering that assays were not related, such an overrepresentation was related to the non-specific activity of compounds, i.e., these compounds were frequent hitters (FH). Therefore, this metric served as an indicator of FH propensity (Table S2, Appendix I). Of course, only compounds with enrichment factor > 1 could be considered as FHs.

For identifying FH, a threshold was selected for the enrichment. The larger the threshold the higher the probability of selected compounds to be FH. If the models and methods of prediction were accurate, the use of the larger threshold could increase their prediction score, i.e., the models should be able to predict compounds that are more likely to be frequent hitters with higher accuracy. This provided an additional measure for comparing the accuracy of the machine learning and the scaffold-based methods. Because of this reason the FH count for the PubChem-Combined set was mentioned to be variable in Table 1.

4.3. Methods

4.3.1. Machine learning methods

Using the freely accessible platform On-line Chemical and Modeling Environment (OCHEM)¹⁹¹, a comprehensive modeling was performed for all three datasets. Different machine learning algorithms available in OCHEM were applied, such as Associative Neural Network (ASNN)¹⁹², Deep Neural Network (DNN)¹⁴⁹, Least Square Support Vector Machine (LSSVM)¹⁰² for training the models. The newly proposed Transformer-CNN²⁰² method that uses the SMILES representation of molecules was also included. Detailed descriptions of the methods were provided in the Methods section of this thesis (Chapter 2).

4.3.2. Molecular descriptors

A variety of descriptors available within the OCHEM environment were used to develop the models. They were described in the Methods section of this thesis (page 37 - 42).

4.3.3. Statistical coefficients

Statistical coefficients used in this study were described in detail in the Methods section of this thesis (Page 44).

4.4. Results

4.4.1. Frequent Hitter Analysis

In order to compare performances of previously developed ToxAlert filters^{19, 199} and machine-learning model, independent test sets were used, different from those utilized to develop the methods. As described in the Data section these sets were selected from PubChem^{19, 201} and identified FHs based on statistical analysis. If one considers compounds with an enrichment value > 10 to be a frequent hitter, then 7633 FHs were identified out of $\sim 350\text{K}$ compounds, corresponding to $\sim 2.1\%$. In comparison, in the *in house* training set there were 190 FH compounds out of 24988, which is $\sim 0.7\%$. The lower number of FHs in the *in house* library could be due to explicit exclusion of potential FHs by using filters on reactive and unstable compounds available in the ToxAlert platform when selecting compounds for the library. While the PAINS filters were not employed in the library design, such a selection still reduced the percentage of FHs.

4.4.2. Machine-learning Models:

For determining comparative performance of machine-learning models against scaffold-based FHs, comprehensive modeling with the combined *in-house* data was performed using a variety of descriptors in OCHEM (Table 4.2).

Table 4.10 : Comprehensive modeling with the OCHEM-Combined dataset. The models presented in the table, with an exception of Transformer-CNN, were created with the Associative Neural Network. The ROC-AUC and BA scores calculated with stratified bagging are shown.

Descriptors	AUC	BA
ALogPS, OEstate	0.86	78
alvaDesc	0.88	79
CDDD#	0.88	78
CDK2	0.86	79
ChemaxonDescriptors	0.86	77
Dragon6 (2D Only)#	0.87	79
Dragon6 (3D)	0.88	80
Fragmentor*#	0.89	81
GSFrag	0.84	76
InductiveDescriptors	0.8	73
JPlogP	0.85	76
MAP4	0.84	73
Mera, Mersy	0.78	69

MORDRED (3D)	0.88	79
PyDescriptor (PyDescriptor)*	0.89	80
QNPR	0.86	78
RDKit (3D)*	0.91	81
SIRMS *#	0.89	80
Spectrophores	0.71	64
StructuralAlerts	0.85	78
Transformer-CNN *#	0.88	81
Consensus	0.92	84
Consensus 2D	0.91	84

*Models denoted were considered for the general consensus model. #Models were used in calculating the 2D only consensus model.

ASNN, and deep learning algorithms available in OCHEM, were used. Out of all methods, ASNN showed best overall performance, as summarized in Table 4.2. The five best performing models were chosen for building the consensus model, which was applied to the test set.

A variety of descriptor packages available in OCHEM were used. Some of the descriptors required the 3D structure of the molecule, for which the Corina software package¹⁸⁹ was used as a part of the modeling pipeline in OCHEM. It was decided to make one consensus model from the best performing models with descriptors that did not require 3D structures, and another consensus model with the best performing models regardless of the descriptor package used. The consensus model using 2D+3D descriptors outperformed the 2D-only consensus model, but only by a small margin with an AUC score of 0.9 for the 2D-only model versus 0.91 for the general consensus model. A similar trend was also observed while modeling with other datasets in the study (Table 4.3 and 4.4). This indicates that 2D structure-based models can be used with almost equal effectiveness to models based on 3D structures. As 3D structure calculation is computationally expensive, 2D structure-based models maybe used as a faster and lighter solution.

A Transformer-CNN²⁰² model was also developed with this data. This method requires no descriptors to be calculated, and therefore, is agnostic of any bias that may arise from using one descriptor over the other. The model is also very lightweight. It required only 8MB of disk space, whereas some of the other models that depend on large descriptor sets such as Dragon needed 300MB.

4.4.3. Comparison between Scaffold based Filter and Machine-learning Models:

To compare the performance of the developed machine learning filter, the general consensus model developed from the combined training set was applied to the test set collected from PubChem data. As discussed in the data section, for classifying FHs, a threshold value for the enrichment was used, that we calculated. As we increased this threshold, performances for machine learning consensus model were compared against the PAINS filter developed by Baell

*et al*³⁷, as well as the combined filter created from previous AlphaScreen studies^{19, 199} using *in house* data.

With a very low enrichment value, the entire library was marked as frequent hitters, and therefore, it resulted in very poor balanced accuracy from all the methods. As the threshold value increased, there was a marked increase in the BA for both PAINS and the machine learning methods, with PAINS outperforming machine learning. However, the balanced accuracy for the AlphaScreen FH based on *in house* data barely increased (Figure 4.2). At an enrichment threshold of >25, the consensus machine learning model calculated a BA of 64%, and the PAINS filter calculated a BA of 65%. It should also be noted that increasing the threshold also reduced the number of identified FHs significantly. At an enrichment threshold of 25, the number of FHs was down to 4k from 7.6k compounds calculated for threshold of 10.

The reason for the poor performance of the scaffold-based FHs was the limited nature of the *in house* dataset. A scaffold-based method could be very effective, but the scaffolds identified must cover a wide chemical space. This was evident from the fact that PAINS, which is also a scaffold-based filter, had significantly better performance. The PAINS scaffolds were identified from six AlphaScreen based assays with a total compound pool of 93212. They covered 2062 FH (2.2%) that were active in four or more out of the six assays. Compared to that, as explained below the *in house* data had a reduced fraction (0.9%) of FHs due to the library design. Therefore, the scaffold-based filter derived from the *in-house* data performed poorly compared to the PAINS filters. However, interestingly, the machine-learning model developed from such a limited dataset was still able to perform almost comparably to the PAINS filter, demonstrating the effectiveness of this approach.

As the machine learning and the PAINS filter both performed well and were derived from independent datasets, they were combined and a compound was considered a FH when it was predicted by any of the approaches. The combined filter improved balanced accuracy by ~5 % over the entire range (ML+ PAINS as shown in Figure 4.2) and showed 68.3% balanced accuracy at an enrichment threshold of >25.

As the performance of the machine-learning model is always limited by the training dataset, it was desired to explore whether better results could be calculated by developing a model with a larger dataset. Therefore, models were developed using PubChem-Combined, and applied to OCHEM-Combined. The rationale was that the PubChem-Combined set was much larger – therefore, a model trained with such a set should be able to pick up FHs from the comparatively limited OCHEM-Combined set.

Comprehensive modeling was performed using OCHEM (Table 4.3), and it was decided to use 5-fold cross-validation instead of bagging, since performing bootstrap aggregation on such a large dataset resulted in very large models. For this dataset, the Transformer-CNN method contributed the model with the highest AUC-ROC=0.94 and BA=87%. A consensus model was built with the Transformer-CNN model and the four other best performing models that were built using ASNNs. When applied to OCHEM-Combined, the consensus model identified 139 out of 190 frequent hitters, resulting in a sensitivity score of 73% and a BA of 79%. Compared

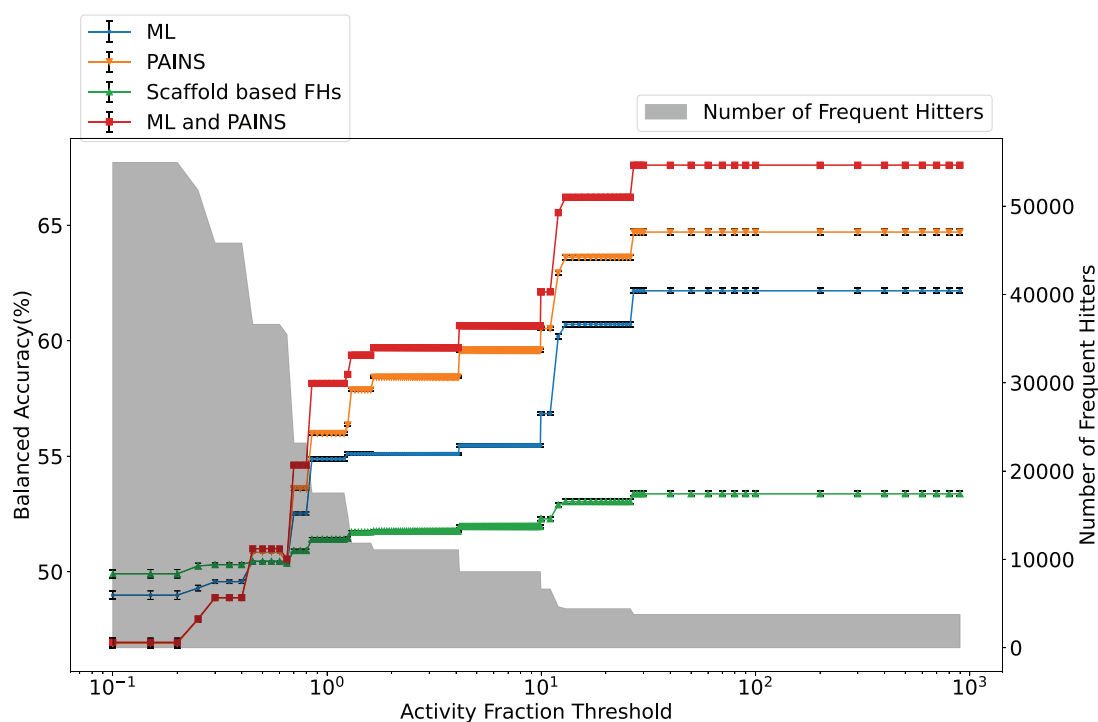


Figure 4.2 : Comparative performance of filters based on the chemical scaffolds groups (Scaffold based FHs from previous studies^{19, 199}) and machine learning model (ML) developed from the same data across a range of selectivity thresholds. The PAINS filter and a combined filter (ML+PAINS) is also shown. Number of compounds identified as FHs in PubChem are shown as bar diagram for the different enrichment thresholds.

to this, the PAINS filter applied to OCHEM-Combined only identified 65 out of 190 frequent hitters, with a sensitivity score of only 34% and BA of 65%. Since both PAINS and consensus model were developed with different sets, one cannot compare their performances directly. Still, this clearly demonstrates that machine learning models, when trained with an appropriate dataset, may provide very good accuracy.

Table 4.11 : Comprehensive modeling using PubChem-Combined dataset. The models presented in the table, with an exception of Transformer-CNN, were created with the Associative Neural Network. The ROC-AUC and BA scores calculated using 5-fold cross-validation are shown. Optimal threshold was used to calculate the BA.

Descriptors	AUC	BA
ALogPS, OEstate#	0.88	83
alvaDesc (3D)	0.87	83
CDDD *#	0.89	83
CDK2	0.87	82
ChemaxonDescriptors	0.84	80
Dragon6 (2D Only)	0.86	82
Dragon6 (All)	0.87	82
Fragmentor *#	0.9	85

GSFrag	0.84	80
InductiveDescriptors	0.74	71
JPlogP	0.86	82
MAP4	0.77	74
Mera, Mersy	0.77	74
MORDRED*	0.89	84
PyDescriptor	0.86	82
QNPR	0.86	81
RDKit (3D)	0.88	84
SIRMS*#	0.89	84
Spectrophores	0.66	63
StructuralAlerts	0.86	81
Transformer CNN*#	0.94	87
Consensus	0.94	86
Consensus 2D	0.93	86

*Models denoted were considered for the general consensus model. #Models were used in calculating the 2D only consensus model.

4.4.4. Machine learning models to identify mechanism of action of FHs

As machine learning models were determined to be effective, also it was decided to develop models that could suggest a mechanism of action (MoA) for FHs (Table 4.3). For example, the frequent hitters in the OCHEM-Ni-NTA and OCHEM-GST datasets (Table 4.1) interfered with Histidine binding to Ni-NTA and interaction of glutathione S-transferase (GST) to glutathione (GSH), respectively. Since the machine learning method provided better performance compared to scaffold-based filters for the combined set, the models developed for each of the sets could be more efficient in identifying the MoA of new FHs. Therefore, all datasets were joined and a multitask model was developed to simultaneously predict whether a compound is a frequent hitter, and its MoA. The training set had 371604 molecules spanning three different target properties, i.e., one for if the molecule is a frequent hitter, and two for the two modes of action. Employed were both a multitask learning approach²⁰³, and multiple single task learning approaches using stratified cross validation. For this study the combined single task

Table 4.12 : Multiple Single task learning for predicting frequent hitter and possible Modes of Action. Different datasets were used to create the three submodels, as appropriate. For example, OCHEM-GST dataset was used to create the model for predicting compounds interfering with GST. The models presented in the table, with an exception of Transformer-CNN, were created with Associative Neural Network. The ROC-AUC and BA scores calculated using 5-fold stratified cross-validation are shown. Optimal threshold was used to calculate the BA.

	AUC			Balanced Accuracy (%)		
	OCHEM-GST	OCHEM-Ni-NTA	PubChem-Combined	OCHEM-GST	OCHEM-Ni-NTA	PubChem-Combined
ALogPS, OEstate	0.79	0.81	0.88	77	78	83

alvaDesc (3D)*	0.82	0.89	0.87	78	84	82
CDDD*#	0.84	0.83	0.89	80	80	83
CDK2*	0.83	0.86	0.87	81	80	82
ChemaxonDe- scriptors	0.79	0.84	0.84	74	78	79
Dragon6 (2D Only) [#]	0.83	0.86	0.86	76	83	82
Dragon6 (3D)	0.81	0.88	0.87	75	83	83
Fragmentor [#]	0.77	0.87	0.89	73	80	85
GSFrag	0.75	0.82	0.84	74	76	80
InductiveDe- scriptors	0.75	0.74	0.74	73	67	71
JPlogP	0.79	0.75	0.86	74	74	82
Mera, Mersy	0.79	0.74	0.77	75	72	74
QNPR	0.78	0.83	0.86	77	77	81
RDKit(3D)*	0.8	0.93	0.88	79	89	84
SIRMS [#]	0.76	0.89	0.89	72	80	84
StructuralAlerts	0.74	0.85	0.86	71	78	81
Transformer CNN*#	0.79	0.87	0.93	80	87	87
Consensus	0.86	0.93	0.93	81	87	86
Consensus-2D	0.84	0.91	0.91	79	85	86

*Models denoted were considered for the general consensus model. #Models were used in calculating the 2D only consensus model.

learning performed better. Among the three properties predicted, PubChem-Combined showed the best performance, followed by OCHEM-Ni-NTA, and OCHEM-GST showed worse performance consistently. This is due to the amount and nature of data involved. PubChem-Combined is a much larger dataset than OCHEM-GST or OCHEM-Ni-NTA, and this produces a better model. OCHEM-GST has fewer compounds marked as active (53 compared to 77). All other compounds between the two sets are the same, so the lesser number of actives results in worse performance. Other models were built from the PubChem-Combined set in this study (Table 4.3). The performances of the two sets of models were very similar (Consensus 0.93 vs 0.94), which is expected due to the multiple single task learning approach used. The transformer neural network showed the best performance amongst individual models, with average AUC of 0.87 and BA of 84.7% (Table 4.4). A consensus model with only 2D descriptors improved the AUC and BA score to 0.90. Including all types of descriptors in the consensus produced the final model, with an AUC of 0.91 and BA of 84.7%. The model is publicly available on the OCHEM web site (<https://ochem.eu/article/125278>).

In order to find out what improvements the model achieved, apart from the statistical scores, we applied this final model to OCHEM-Combined. It was able to identify 140 out of 190 frequent hitters. This is very similar to the performance shown by models built from the PubChem-Combined dataset (Table 4.3), and this is expected. As explained previously, the PAINS filter identified only 65 of these 190 compounds, so the model identified an additional 75 compounds over the PAINS filters. The full list of these compounds is presented in Appendix II of

this thesis, and a few scaffolds of interest are presented in Figure 4.2. In general, the collection is quite heterogeneous, and no prominent scaffold could be identified. This is expected, as a well-known scaffold-based filter such as PAINS was not able to identify these compounds. There are a few toxoflavines, as well as toxoflavin mimics. Also of note, were the presence of multiple condensed polyaromatic moieties, Cyanodithiines, aminothazoles and Picolylamines (Figure 4.3). Many of these scaffolds have been identified manually before, and are already present in the scaffold-based filter. However, as has been discussed in detail, scaffold-based filters perform very poorly against a broad test set. They are able to pick up these particular scaffolds, but suffer greatly in overall performance. On the other hand, the machine learning model was able to identify these scaffolds without prior exposure to them, and provided much better overall performance (79% BA compared to 65% of PAINS, when the final model was applied to OChem-Combined). Therefore, in the final model, the presence of a frequent hitter is identifiable with better accuracy, and further the detection of scaffolds and compounds is possible that a traditional method such as PAINS may overlook. In addition to this, it is also possible to comment on a potential mode of action. This makes our model an excellent tool for filtering frequent hitter compounds in AlphaScreen assays.

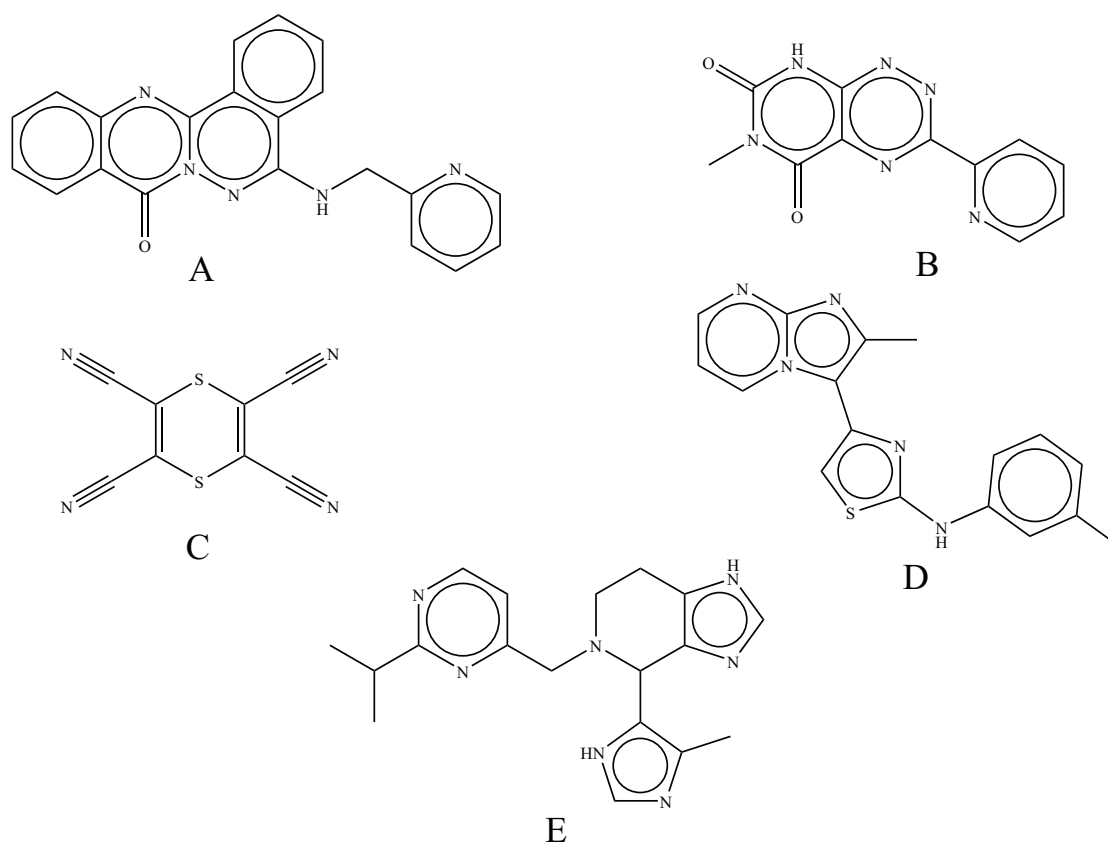


Figure 4.3: Examples of confirmed AlphaScreen frequent hitter compounds identified by the ML filter but not by PAINS. **A.** Scaffold resembling toxoflavin **B.** Toxoflavin **C.** Cyanodithiine **D.** aminothiazole and **E.** Picolyamine.

Conclusion:

Scaffold-based methods and ML methods were compared for identifying AlphaScreen frequent hitters. The result showed that ML models outperformed scaffold-based methods by a large margin, when both approaches originated from the same data. It was demonstrated that a ML model trained on a large dataset collected from PubChem outperformed PAINS³⁷ filters for prediction of both public and *in house* data. However, combining PAINS with a ML model improved the prediction outcome. Further, ML models were developed from the individual *in-house* datasets, for which the mechanism of action (MoA) was known. These models should be effective in identifying compounds that interfere through particular MoAs and should have higher accuracy compared to existing scaffold-based filters developed in previous studies^{19, 199}. Finally, was contributed a ML model based on the PubChem dataset. As this set was much larger and more diverse than the training set used to develop PAINS, it provided better coverage of the chemical space, and was able to identify frequent hitters in the *in-house* AlphaScreen assays with higher accuracy than the PAINS filter. Also demonstrated was that the new ML model is able to identify scaffolds that are not identified by PAINS, without introducing false positives, and thus provides better balanced accuracy scores.

In summary, the scaffold-based methods were limited to the identified scaffolds only, and therefore had lower accuracy compared to ML when both types of models were used to screen large heterogeneous datasets. Identifying scaffolds from large data sets, such as the PubChem-Combined would be a very daunting task, and would take a very long time. ML on the other hand benefited from very large datasets. Re-training the models with new datasets is trivial, and adding new molecules could further improve the accuracy of the model. If a scaffold-based filter is too specific, it is ineffective in large sets, if it is too general and has many scaffolds, then it may produce too many false positives. Machine learning can identify much more sophisticated and complex patterns in available data, and therefore provide better prediction accuracy and generalizability.

The models developed in this study are freely available on the OCHEM platform (<https://ochem.eu/article/125278>).

Chapter 5

Modelling False Positives in GPCR assays ²⁰⁴

5.1. Introduction

G-Protein Coupled Receptors (GPCR) are the largest family of cell surface receptors²⁰⁵. These plasma membrane-bound receptors have evolved to recognize a variety of extracellular physical and chemical signals and, upon recognition, act as the proximal stimulus in cell signalling pathways. With over ~800 members²⁰⁶, GPCRs are involved in almost every physiological function, from sensation to growth to hormone responses. Due to their widespread physiological relevance, and the presence of druggable sites, GPCRs are one of the major targets of therapeutic drugs. A 2017 study notes that 475 drugs act at 108 unique GPCRs. Approximately 321 agents are currently in clinical trials, of which ~20% target 66 potentially novel GPCR targets. GPCRs also account for ~27% of the global market share of therapeutic drugs, with aggregated sales for 2011–2015 of ~US\$890 billion.

As promising drug targets, assays involving a member of the GPCR family are commonly employed in High Throughput Screening (HTS) campaigns. There is a plethora of different techniques and a wide range of commercial kits available, many of which are suitable for High Throughput Screening (HTS)²⁰⁷. In such HTS, identifying false positives is a challenge. False positives may be compounds that interfere with the assay detection technology in some way, such as inhibiting luciferase in luciferase-based system¹⁴¹, or quenching fluorescence where it is the final readout¹⁹. There may also be compounds that are not specific to the target protein, but are promiscuous, either to a narrow or broad class of proteins¹⁸.

5.2. Data

5.2.1. Data description

An initial goal was the exploration of available data, to find suitable assays that can then be used for further analysis. On PUBCHEM were identified 92 assays with more than 500 compounds for GPCR agonists and antagonists. The two were separated and it was decided to focus on the agonists. This was to narrow down the scope of the study. From the list of available agonist screenings were selected the 20 assays with the highest number of active compounds, since the aim is to find False Positives (FPs). Assays that have little to no positives are less relevant. For further selection of particular assays, the focus was on GPCR subtypes as described below.

5.2.2. Data Collection

The GPCR family is commonly classified into five different families based on their structural and sequence similarity. The families are then further classified into a family tree^{208, 209}. Of these five major families, the Rhodopsin class is the largest. For selecting assays, the target proteins were mapped onto this family tree (Fig. 1), and assays were selected with sets of representative proteins distant from each other in the family tree. This ensured that compounds that are frequently active, are not preferential agonists of a subtype of GPCR, but are more likely a result of an assay artifact.

Using these criteria a set of 12 assays was chosen, and compounds that are frequently active in these assays were sought (see Methods section), i.e. actives across all of the various different

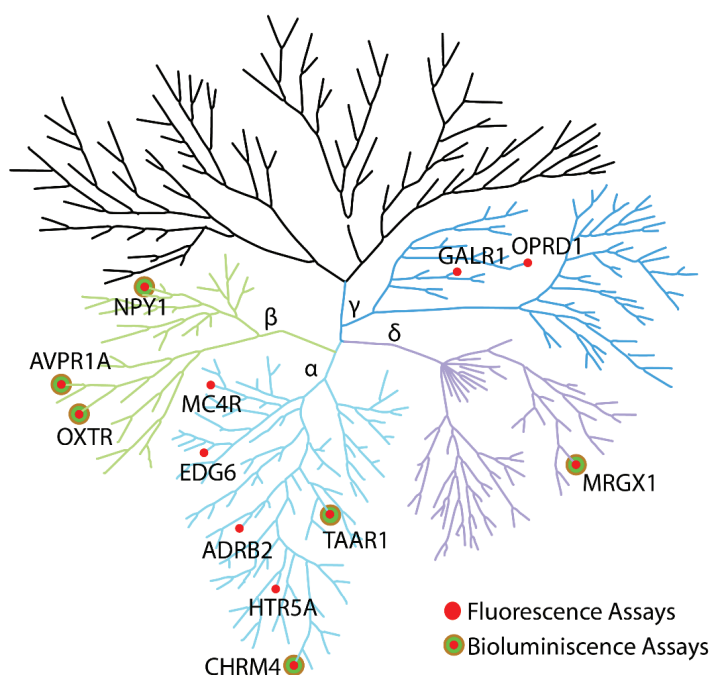


Figure 5.1: GPCR family tree represented as a tree and dots mapping the protein targets in identified assays. The colored part of the tree represents the Rhodopsin class of the GPCR family and various subfamilies of the Rhodopsin class are marked with different colors.

subtypes and assay technologies and thus frequent hitters of the Rhodopsin class of GPCR. However, only 59 out of 373,131 compounds matched our definition of being frequently active. Upon closer examination it was found that these compounds were tested only thrice, and therefore are more likely to be an artifact of selection criteria rather than a GPCR frequent hitter or assay artifact.

To further refine the search, focused was pivoted to the different detection technologies used in the assays. It was found that half of the assays (six) used fluorescence while the other six assays used bioluminescence. Only 71 compounds were frequently active in the bioluminescence group. In the fluorescence group, although the number of datapoints and active compounds was very similar, 502 compounds were frequently active (Table 5.1). This indicates that fluorescence technology contributes many more artifacts and these 502 compounds were selected for further analysis.

All data were harvested from PUBCHEM²¹⁰, manually or by using the PUBCHEM REST API with Python. All data were obtained and stored locally in the CSV format to be analyzed later with various python scripts.

Table 5.13 : Statistics of compounds for the datasets used in the study.

	Inactive	Active	Frequently Active
All assays	352685	20446	59
Fluorescence Assays	363459	9605	502
Bioluminescence Assays	358770	10841	71

5.2.3. Frequent Hitter Flagging

Frequent hitters were defined as compounds that were active according to our criteria in more than half of the assays in which they were tested. Additionally, each compound had to be tested at least in three different assays. Compounds satisfying both criteria were identified using a Python script and flagged as frequent hitters.

5.3. Methods

5.3.1. Data Gathering:

All data were harvested from PUBCHEM, sometimes manually, other times by using the PUBCHEM REST API with Python. All data were obtained and stored locally in the CSV format to be analyzed later with various python scripts.

5.3.2. Activity Cross-check:

To see if these flagged compounds were frequently active in other assays, the list of assays in which each compound was tested was retrieved from PUBCHEM. It was then checked if the compound was active or not in these assays. By repeating this process for all the compounds, a list of assays was compiled, in which it was known how many flagged compounds are active in each. Then, PUBCHEM was queried for information regarding the assay technology used, and the assays were annotated. Plotting this, gives us Figure 5.2. The entire process, from getting the list of assays to producing the figure, was done using Python. For annotating the assays, text mining with Python was followed by manual curation.

5.4. Machine Learning

XGBoost, DNN and ASNN algorithms were employed from the online modelling environment OCHEM. Details of the algorithms are described in the Methods section of this thesis (page 33-37).

5.5. Results and Discussion

5.5.1. Compound Activity Profile:

The flagged compounds in the bioluminescence group were checked and it found that although some of the compounds were tested, none of them showed any activity. This improved confi-

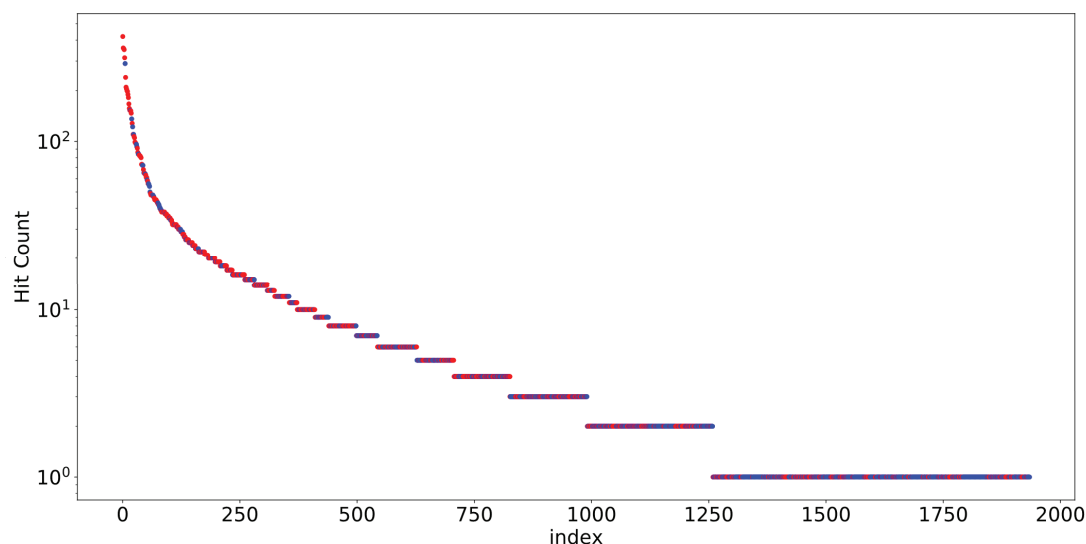


Figure 5.2 : Plot showing activity profile of the identified frequent hitter compounds and their correspondence with the assay technology used. X axis represent assay index, and each point represents an individual assay. Assays with more identified frequent hitters shows correspondence with fluorescence being used as assay technology.

dence that the compounds were indeed fluorescence assay artifacts. In order to test this further, PUBCHEM was inspected for assays where these flagged compounds were tested. A plethora of different unrelated assays were found, and where compounds were active in these assays was noted. A majority of such assays where the compounds were active, fluorescence was being used as the detection technology. This led to the conclusion that the flagged compounds were indeed artifacts of the assay detection technology present in GPCR screening.

5.5.2. Scaffold Identification:

To recognise prominent scaffolds in the identified compounds, hierarchical clustering was performed using Circular Morgan Fingerprints. About 10 major clusters were identified with more than 20 compounds including one particular cluster with 68 compounds.

From the hierarchical clustering, a few prominent motifs were identified that were present in the largest clusters. The identified scaffolds were encoded into SMARTS and checked for enrichment in the false positive class, as compared to the active and inactive classes (Table 5.1). This led to a few scaffolds of interest (Table 5.2). In particular the 2-thioxoimidazolidin-4-one containing scaffold showed a 37.5-fold enrichment among the actives, though the coverage was very poor (0.05%). The urea motif on the other hand showed a 6.25-fold enrichment compared

to regular urea motif (1.44) in the inactive class.

The identified false positives were also examined for PAINS scaffolds. In total, 114 compounds were identified that matched the PAINS scaffolds identified by Baell and Holloway. The scaffolds identified in this study performed better in terms of coverage and selectivity as compared to PAINS for identifying the false positives, but overall accuracy and coverage was still poor. Therefore, it was decided to use machine learning to build a model for identifying these compounds.

Table 5.14: Table showing hits found by the identified scaffolds among the active, inactive, and frequently active compounds.

SMARTS Query	Actives	Freq. Hit-ters	Inactives	Enrichment
<chem>O=C1[NX3]C(NC1=Cc:2:c:c:c(:c:c:2)[NX3])=S</chem>	10	9	5	37.5
<chem>CC1(C(N(C(N1)=O)C)=O)c:2:c:c:[cX3]:c:c:2</chem>	55	17	725	2.13
<chem>C(=[O,o,S,s,N,n])NC(=[O,S])N[N,C,n,c]</chem>	428	103	9858	0.99
<chem>NC(N)=S</chem>	473	88	12072	0.69
<chem>[NX3]C(CC([NX3])=O)=O</chem>	78	7	2282	0.29
<chem>NC(N)=O</chem>	448	30	18047	0.16
<chem>O=C(CC[NX3])Nc:1:c:c:c:c:1</chem>	81	6	4998	0.18
<chem>N(~[SX4](=O)=O)*</chem>	894	49	55471	0.08

5.5.3. Machine Learning

The analyzed methods were used in combination with different descriptors sets. LSSVM provided on average the highest accuracy amid the chosen algorithms (Table 5.2). LSSVM models were selected with the highest accuracy based on their ROC-AUC score for building a consensus model. The consensus model had ROC-AUC score of 0.93 with a balanced accuracy of 86%.

Table 5.15: The performance of models built using the GPCR dataset. The ROC-AUC scores are calculated using 5-fold stratified cross-validation. Models marked with asterisk were used to build the consensus model.

Descriptors/methods	DNN	ASNN	XG-BOOST	LSSVM
AlogPS, OEstate (2D)	0.84	0.84	0.87	0.89*
CDK2 (3D)	0.79	0.85	0.86	0.87*
ChemaxonDescriptors (3D)	0.82	0.82	0.84	0.88*
Dragon6 (2D blocks)	0.83	0.87	0.88	0.91

Dragon6 (3D, all blocks)	0.87	0.85	0.89	0.91*
Fragmentor (2D)	0.85	0.83	0.88	0.89*
GSFrag (2D)	0.81	0.8	0.86	0.85
InductiveDescriptors (3D)	0.79	0.78	0.79	0.83
JPlogP (2D)	0.82	0.79	0.85	0.84
Mera, Mersy (3D)	0.69	0.76	0.8	0.81
PyDescriptor (3D)	0.89	0.86	0.85	0.89*
QNPR (2D)	0.81	0.82	0.87	0.86
RDKit (2D, all blocks)	0.88	0.88	0.87	0.91*
RDKit (3D, all blocks)	0.88	0.88	0.87	0.91
SIRMS (2D)	0.86	0.83	0.86	0.87*
Spectrophores (3D)	0.63	0.69	0.72	0.68
StructuralAlerts (2D)	0.79	0.79	0.77	0.78
alvaDesc (2D blocks)	0.86	0.85	0.87	0.91*
alvaDesc (3D, all blocks)	0.88	0.86	0.88	0.91

For model testing, an independent dataset was constructed by looking up GPCR agonist assays in PUBCHEM that were not used for the training set. Five relevant assays were identified with 4323 active compounds. Frequent hitter analysis identified 157 compounds from these 5 assays. The consensus model predicted the molecules from this set with a balanced accuracy of 76% and an AUC score of 0.85. The consensus model which was based only subset of 2D descriptors provided a similar accuracy of 75% and AUC score of 0.85 thus indicating the importance of mainly 2D information for this analysis.

5.6. Conclusion

In this study, GPCR assays from PUBCHEM were analyzed with the aim of identifying frequent hitters. It was found that fluorescence-based assays are more susceptible to false positives than bioluminescence. Compounds that were frequent hitters in fluorescence-based assays did not appear as frequent hitters in bioluminescence assays. A predictive machine-learning model to identify such compounds for GPCR assays was developed. The provided analysis can help to interpret HTS screening using GPCR assays.

Comparison with other tools

The performance of models generated with existing frequent hitter filters was compared. Most such filters were targeted towards selecting promiscuous compounds, rather than trying to identify compounds that interfere with a given particular assay type. HitDexter⁷⁰ is one such publicly available filter, developed by Kirchmair *et al*, against which the developed model was compared.

A luciferase counterscreen assay from PubChem (AID 1379) was selected with 201160 tested compounds. After removing the inconclusive results and common compounds with the training set, 95 actives (compounds that inhibited luciferase) remained and 36362 inactive molecules. The developed model performed against this set with a balanced accuracy of 72%, AUC 0.8. It identified 74 out of 95 luciferase inhibitors, with a sensitivity score of 0.78. As this is an independent set with no overlapping molecules, some of the molecules were outside the applicability domain of the model. Reevaluating the test set with those molecules excluded improved the balanced accuracy score to 73%.

Some difficulty was encountered in applying such a large set through HitDexter, so initially just the 95 inhibitors were considered. Among them, HitDexter identified only 18 compounds to be highly promiscuous with a high confidence (1.0), and 33 compounds with moderate confidence (>0.5). This calculates to a sensitivity score of 0.34, as compared to 0.78 for our model. As our model is trained on luciferase counterscreen data, it is much more effective in finding luciferase frequent hitters, as opposed to a frequent hitter filter such as HitDexter. Thus it is demonstrated increased efficacy in identifying offending compounds in a particular type of assay system, compared to a generic frequent hitter filter.

Machine learning and scaffold-based methods were compared, and in the process further comparison was drawn against PAINS. Trained on a diverse dataset, machine learning models were able to identify frequent hitters with significantly better balanced accuracy than PAINS. also It was also noted that the Machine Learning model was able to identify additional scaffolds.

The model built for finding GPCR frequent hitter had an external test set. So, it was decided to use a subset of that for our comparison. All the actives were selected and an equal number of inactives was sampled from the test set. In this case, both HitDexter and our model performed with a Balanced Accuracy of 77%. A slightly better sensitivity was observed, contrasted to HitDexter which showed better specificity score (Table 6.16). This model was built to identify frequent hitters, similarly to HitDexter. Thus, they performed almost equally well.

Table 6.16 : Confusion matrices for HitDexter and our model tested against the GPCR test set.

HitDexter			GPCR FH Model		
Predicted→	No	Yes	Predicted→	No	Yes
Real↓			Real↓		
No	132	26	No	110	48
Yes	44	113	Yes	26	131

Discussion

The primary objective of this project was to develop filters to flag false positives in High Throughput Screening. In the course of the past three years in this project, three different assay systems for finding frequent hitters were examined, each with a different scope. Each assay differed from another in its protocol and, of course, each of the assays had different frequent hitters. Therefore, models were developed to identify the artefact compounds for each given type of assay, and further for the luciferase assay given a mechanistic interpretation of the interference. In this work, compounds were identified through statistical analysis and also high-performance Machine Learning models were developed for analyzed assays.

For identifying frequently active compounds, different metrics can be used. In projects 1 and 3 presented in this thesis, frequent hitters were determined by calculating activity frequency and selecting a static threshold. This threshold was intuitive and could work for studies where all compounds were tested in the similar number of assays, but a more rigorous statistical analysis was to use a dynamic threshold, which depended on the number of assays in which compound was tested. In the study involving AlphaScreen, a Binomial distribution was used to determine a suitable activity frequency threshold. Also, compounds that had not been tested at least three times were excluded. Using additional criteria, such as clustering compounds based on scaffolds, and then considering frequently active scaffolds, may provide additional confidence in selecting frequent hitters.

In this thesis, molecules were consistently represented using SMILES strings. However, SMILES from different sources may not be canonical, and may follow different standards for canonicalization. Before processing, the molecules therefore should be standardized, and then and filtered for invalid molecules. Otherwise, a difference in canonicalization may result in a difference in structure, which may translate to the calculated descriptors being different, which in turn may lead to variation in the model or prediction. In the projects, the Standardizer from ChemAxon, built into OCHEM, was used to standardize molecular sets. OCHEM also checked for internal and external duplicates, and provided the option to remove them.

In the third chapter, a machine-learning model was developed to identify false positives in Luciferase assays that arise due to Luciferase inhibition. Though Luciferase inhibition could contribute a large number of frequent hitters, false positives could also arise due to other reasons, such as chemical reactivity and promiscuity. The developed model did not cover such cases, so a separate promiscuity filter should be used in conjunction with the Luciferase Advisor model. Three Luciferase counterscreen assays were selected from PubChem as datasets for modelling, all of which had been developed for the wild-type Luciferase from firefly (*Photinus pyralis*). Consequently, the developed model was most relevant for screening experiments using the wild type Luciferase. There are several variants of Luciferase enzymes from different organisms such as the click beetles from the superfamily Elateroidea, marine organisms like Sea Pansy (*Renilla reniformis*), Photobacetria such as *Vibrio fischeri*, and Dinoflagellates. There are also engineered variants of the enzyme, such as UltraGlo™ from Promega Corp. The different structures of Luciferase enzyme could have different structure activity relationship (SAR), and

therefore could have different compounds as their inhibitors. While developing the models, other variants of Luciferase were examined, but insufficient counterscreen data was found in the public domain to build a Machine Learning model. The developed model may still identify false positive hits against such other variants of Luciferase, as closely related targets often have similar ligands. However, the resultant model will have the highest accuracy when applied to the assays using wild-type Luciferase.

As discussed in the introduction section, inhibition of enzymes can be mediated by molecules which form aggregates. For this reason, it was tried to be understood if any of the compounds reported as Luciferase inhibitors were known as aggregators as well. However, due to the lack of experimental data and good prediction models, a definite conclusion could not be reached. A study where the reported luciferase inhibitors are experimentally tested for aggregation would prove valuable for elucidating the mode of action for such inhibitors.

In this study a reliable model for flagging luciferase inhibitors was constructed, which would interfere with assays involving Luciferase. However, this model can be further refined by taking frequent hitters into consideration. Therefore, a follow-up study which develops further on frequent hitters in luciferase assays may be useful. This would allow one to flag other potentially unwanted compounds in luciferase assays, and also understand the contribution of inhibition versus compound promiscuity in the context of unwanted compounds in Luciferase assays.

In the fourth chapter, the comparison between machine learning models and scaffold-based filters for AlphaScreen frequent hitter was examined. The scaffold-based filters were based on a highly imbalanced, relatively small *in-house* datasets, and for comparison, our model was trained on the same data. For test set a comparatively large set was prepared from PubChem. The machine learning model performed significantly better than the scaffold-based filters, but PAINS was better than the machine learning model. This led to the hypothesis that the model must be limited by the training data, and to test this, another model was built using PubChem data. This model was able to predict our *in-house* set with 79% balanced accuracy, far outperforming PAINS (65%). Finally, a multitask model was developed to predict if a molecule is a frequent hitter, and its possible mode of action.

The main constraint in this study was test data. Assays contain many variables, and if the analysis is unable to account for them, then artifacts would be introduced. In this study it was initially planned to develop individual models per mode of action and test them against appropriate test sets. However, it was not possible to find such test sets on PubChem, and so datasets were merged to create a generic AlphaScreen frequent hitter class. Experiments need to be carried out to disambiguate the various modes of interference a compound may show, but if such test sets can be obtained for each mode of action, our final model can be tested against it to gauge its effectiveness in predicting the modes of action.

Finally, in the fifth chapter the study of frequent hitters in GPCR assays was performed. GPCRs are a very large family of proteins, and preferred scaffolds are known to be active against GPCR subtypes²¹¹. We attempted to exclude such preferential scaffolds by selecting assays with GPCRs from different families in the phylogenetic tree. However, it is possible to have com-

pounds active against multiple families of GPCR, and they will be labelled as frequent hitters. Arguably, as they lack specificity, they may not be interesting as drug leads, but classifying such compounds as promiscuous would be incorrect as well. A scaffold-based filter to identify such compounds before the model building may help mitigate this issue further.

In the GPCR study, 6 assays involving fluorescence-based detection methods were considered, and 6 assays with bioluminescence as a detection method. It was found that significantly more (~ 500 as compared to 72) frequently active compounds came from the fluorescence-based assays. This highlights the point that frequent hitters are assay system dependent, and it is therefore worthwhile taking the assay technology into consideration while looking for frequent hitters.

Machine learning was used extensively in all the projects in this thesis. As OCHEM supports various different machine learning algorithms and descriptors, this was utilized as best possible. Here are reported the best performing models, however, building models with other popular algorithms available in OCHEM such as LibSVM, XGBoost and DNN was also explored. For our datasets, the Associative Neural Network algorithms performed consistently very well. One reason for this could be because of the highly imbalanced nature of the training set. As discussed in the Methods section, ASNN is effective in modelling such highly imbalanced datasets, and that may be the reason why it has either produced the best models or produced models that are very close to the best performing models in our datasets.

The highly imbalanced nature of the data can also lead to issues while judging model performance. As seen in the methods section, for such datasets, balanced accuracy must be used, as opposed to regular accuracy. Therefore, model performance was reported either in balanced accuracy, or as a ROC-AUC score.

A mixture of 2D and 3D descriptors was employed for building the models. For generating 2D conformation from the 2D structure information available in PubChem, Corina was used, which is the default method in OCHEM. Interestingly, minor differences were often observed in model performance between 2D and 3D descriptors (Table 3.5, 4.3 and 5.3), the algorithm and descriptor used often has a more significant impact. This indicates that 3D structural information may not be crucial in building machine learning models. Calculating 3D structures of molecules is computationally very expensive, and is prone to errors. Conformations generated computationally can be infeasible in the real world, and the compound may adopt an entirely different conformation, which may lead to different descriptors, and therefore compound error in the model. Thus, when building machine learning models in cheminformatics, an entirely 2D workflow should also be considered.

Conclusion

In this thesis, computational filters were developed for identifying frequent hitters in high throughput assays. Three different types of assays were considered, namely Luciferase based assays, AlphaScreen, and GPCR assays, and high-performance models were developed for each. It was demonstrated that the Luciferase Inhibitor detection model developed as a part of the study outperforms other popular methods available for detecting false positives in a luciferase-based assay. Comparative performances of scaffold-based and machine learning models were explored in identifying frequent hitters of AlphaScreen, and machine learning models were developed that are trained to identify false positives in GPCR assays. Compounds that behave poorly in assays have been a consistent problem for biologists and the problem has only been compounded after the introduction of robot-driven High Throughput Assays. The studies described in this thesis add to the growing collection of computational filters being developed to identify and flag such compounds in the High Throughput Screening context. Recently, there has been an explosion of interest in Machine Learning and Artificial Intelligence across all fields of science. Through our development of the machine learning models described herein, contributions have also been made to the machine learning community, demonstrating how machine learning can be leveraged to identify false positives and frequent hitters. Such targeted approach of finding and modelling frequent hitters and false positives in specific assay type or technology is more useful and applicable to the assay systems under consideration. A rational approach was followed for developing these models, and the models have been made freely available through the OCHEM platform. The models, and procedure followed for developing them, will be useful to the scientists performing such assays, and to the scientific community at large.

Acknowledgments

I would like to personally thank Dr. Igor Tetko, who has been the guide and mentor for me throughout the project. Without his constant effort and help and patience, the project and my Doctoral studies would never have succeeded. I would like to thank Dr. Uwe Koch for mentoring me during my time at the Lead Discovery Center, as well as helping me move forward beyond my PhD. For their invaluable guidance through my doctoral studies, I would like to thank Dr. Kamyar Hadian. My sincere thanks to Prof. Dr. Michael Sattler for his guidance and supervision during my doctoral studies. Despite his busy schedule, he always found time to give his valuable inputs on my studies. I would like to thank Dr. Hongming Chen and Dr. Ola Engvist for helping me during my secondment at AstraZeneca. A very big thanks to all the people in Dr. Tetko group: Dr. Ekaterina Ratkova, Dr. Pavel Karpov, Zhonghua Xia for being together with me during my stay at the lab and beyond and being such great company. A special thanks to Michael Withnall, fellow doctorate student, for helping from day one, and being a good friend. He has also helped proofread the manuscript of this thesis, which is no small task. I would also like to thank the people here in the Lead Discovery Center, Dr. Bert Klebl, Dr. Peter Nussbaumer, and Dr. Michael Hamacher for welcoming me into LDC. A special thanks to officemates in LDC during my PhD, Matthäus Brandt, Blaz Andlovic and Pragya Jatoo, and all other awesome people I met during my stay here at LDC.

The project leading to this thesis has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 676434, "Big Data in Chemistry". The thesis reflects only my view and neither the European Commission nor the Research Executive Agency (REA) are responsible for any use that may be made of the information it contains. I thank ChemAxon (<http://www.chemaxon.com>) for Academic license of software tools (Standartizer, ChemAxon plugins) as well as AlvaScience (<http://alvascience.com>), Molecular Networks GmbH (<http://mn-am.com>) and Chemosophia (<http://chemosophia.com>) for providing descriptors and Corina 2D to 3D conversion program used in this study.

References

1. Gaynes, R., The Discovery of Penicillin—New Insights After More Than 75 Years of Clinical Use. *Emerging Infect. Dis.* **2017**, *23* (5), 849-853.
2. Lobanovska, M.; Pilla, G., Penicillin's Discovery and Antibiotic Resistance: Lessons for the Future? *The Yale journal of biology and medicine* **2017**, *90* (1), 135-145.
3. Recent Advances in Microscopy. *Nature* **1934**, *133* (3356), 286-287.
4. Boutros, M.; Heigwer, F.; Laufer, C., Microscopy-Based High-Content Screening. *Cell* **2015**, *163* (6), 1314-25.
5. Richards, O. W., Some Recent Advances in Microscopy. *Transactions of the American Microscopical Society* **1949**, *68* (4), 292-303.
6. Watson, J. D.; Crick, F. H. C., Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature* **1953**, *171* (4356), 737-738.
7. D W Green, V. M. I., Max Ferdinand Perutz, The structure of haemoglobin - IV. Sign determination by the isomorphous replacement method. *Proceedings of the Royal Society A* **1954**, 225.
8. Kirk, R., The first of its kind. *Nature* **2014**, *511* (7509), 13-13.
9. Khanna, I., Drug discovery in pharmaceutical industry: productivity challenges and trends. *Drug Discovery Today* **2012**, *17* (19), 1088-1102.
10. Ben-Menahem, S. M.; von Krogh, G.; Erden, Z.; Schneider, A., Coordinating Knowledge Creation in Multidisciplinary Teams: Evidence from Early-Stage Drug Discovery. *Academy of Management Journal* **2015**, *59* (4), 1308-1338.
11. Tetko, I. V.; Engkvist, O.; Koch, U.; Reymond, J. L.; Chen, H., BIGCHEM: Challenges and Opportunities for Big Data Analysis in Chemistry. *Mol Inform* **2016**, *35* (11-12), 615-621.
12. Tetko, I. V.; Engkvist, O.; Chen, H., Does 'Big Data' exist in medicinal chemistry, and if so, how can it be harnessed? *Future Med Chem* **2016**, *8* (15), 1801-1806.
13. Baell, J. B., Redox-active nuisance screening compounds and their classification. *Drug Discovery Today* **2011**, *16*, 840.
14. Mayr, L. M.; Bojanic, D., Novel trends in high-throughput screening. *Curr. Opin. Pharmacol.* **2009**, *9* (5), 580-8.
15. Bickle, M., The beautiful cell: high-content screening in drug discovery. *Anal. Bioanal. Chem.* **2010**, *398* (1), 219-26.
16. Hughes, J. P.; Rees, S.; Kalindjian, S. B.; Philpott, K. L., Principles of early

- drug discovery. *Br J Pharmacol* **2011**, *162* (6), 1239-49.
17. Sink, R.; Gobec, S.; Pecar, S.; Zega, A., False Positives in the Early Stages of Drug Discovery. *Curr. Med. Chem.* **2010**, *17* (34), 4231-4255.
 18. Roche, O.; Schneider, P.; Zuegge, J.; Guba, W.; Kansy, M.; Alanine, A.; Bleicher, K.; Danel, F.; Gutknecht, E. M.; Rogers-Evans, M.; Neidhart, W.; Stalder, H.; Dillon, M.; Sjogren, E.; Fotouhi, N.; Gillespie, P.; Goodnow, R.; Harris, W.; Jones, P.; Taniguchi, M.; Tsujii, S.; von der Saal, W.; Zimmermann, G.; Schneider, G., Development of a virtual screening method for identification of "frequent hitters" in compound libraries. *J. Med. Chem.* **2002**, *45* (1), 137-42.
 19. Schorpp, K.; Rothenaigner, I.; Salmina, E.; Reinshagen, J.; Low, T.; Brenke, J. K.; Gopalakrishnan, J.; Tetko, I. V.; Gul, S.; Hadian, K., Identification of Small-Molecule Frequent Hitters from AlphaScreen High-Throughput Screens. *J. Biomol. Screen.* **2014**, *19* (5), 715-26.
 20. Magolda, R., Faculty of 1000 evaluation for Fluorescence spectroscopic profiling of compound libraries. In *F1000 - Post-publication peer review of the biomedical literature*, F1000 (Faculty of 1000 Ltd): 2008.
 21. Leitão, J. M. M.; Esteves da Silva, J. C. G., Firefly luciferase inhibition. *J. Photochem. Photobiol. B: Biol.* **2010**, *101* (1), 1-8.
 22. Thorne, N.; Auld, D. S.; Inglese, J., Apparent activity in high-throughput screening: origins of compound-dependent assay interference. *Curr. Opin. Chem. Biol.* **2010**, *14* (3), 315-24.
 23. Thorne, N.; Shen, M.; Lea, W. A.; Simeonov, A.; Lovell, S.; Auld, D. S.; Inglese, J., Firefly luciferase in chemical biology: a compendium of inhibitors, mechanistic evaluation of chemotypes, and suggested use as a reporter. *Chem. Biol.* **2012**, *19* (8), 1060-72.
 24. Rishton, G. M., Reactive compounds and in vitro false positives in HTS. *Drug Discovery Today* **1997**, *2* (9), 382-384.
 25. Potashman, M. H.; Duggan, M. E., Covalent Modifiers: An Orthogonal Approach to Drug Design. *J. Med. Chem.* **2009**, *52* (5), 1231-1246.
 26. Dragovich, P. S.; al, e.; et al., ChemInform Abstract: Structure-Based Design, Synthesis, and Biological Evaluation of Irreversible Human Rhinovirus 3C Protease Inhibitors. Part 7. Structure-Activity Studies of Bicyclic 2-Pyridone-Containing Peptidomimetics. *ChemInform* **2010**, *33* (24), no-no.
 27. Matthews, D. A.; Dragovich, P. S.; Webber, S. E.; Fuhrman, S. A.; Patick, A. K.; Zalman, L. S.; Hendrickson, T. F.; Love, R. A.; Prins, T. J.; Marakovits, J. T.; Zhou, R.; Tikhe, J.; Ford, C. E.; Meador, J. W.; Ferre, R. A.; Brown, E. L.; Binford, S. L.; Brothers, M. A.; DeLisle, D. M.; Worland, S. T., Structure-assisted design of mechanism-based irreversible inhibitors of human

- rhinovirus 3C protease with potent antiviral activity against multiple rhinovirus serotypes. *Proceedings of the National Academy of Sciences* **1999**, 96 (20), 11000-11007.
28. Blanchard, J. E.; Elowe, N. H.; Huitema, C.; Fortin, P. D.; Cechetto, J. D.; Eltis, L. D.; Brown, E. D., High-Throughput Screening Identifies Inhibitors of the SARS Coronavirus Main Proteinase. *Chem. Biol.* **2004**, 11 (10), 1445-1453.
29. Epps, D. E.; Taylor, B. M., A Competitive Fluorescence Assay to Measure the Reactivity of Compounds. *Anal. Biochem.* **2001**, 295 (1), 101-106.
30. McGovern, S. L.; Caselli, E.; Grigorieff, N.; Shoichet, B. K., A Common Mechanism Underlying Promiscuous Inhibitors from Virtual and High-Throughput Screening. *J. Med. Chem.* **2002**, 45 (8), 1712-1722.
31. Elcock, A., Faculty of 1000 evaluation for Kinase inhibitors: not just for kinases anymore. In *F1000 - Post-publication peer review of the biomedical literature*, F1000 (Faculty of 1000 Ltd): 2003.
32. Malvezzi, A.; de Rezende, L.; Izidoro, M. A.; Cezari, M. H. S.; Juliano, L.; Amaral, A. T. d., Uncovering false positives on a virtual screening search for cruzain inhibitors. *Bioorganic & Medicinal Chemistry Letters* **2008**, 18 (1), 350-354.
33. Reddie, K. G.; Roberts, D. R.; Dore, T. M., Inhibition of Kinesin Motor Proteins by Adociasulfate-2. *J. Med. Chem.* **2006**, 49 (16), 4857-4860.
34. Feng, B. Y.; Toyama, B. H.; Wille, H.; Colby, D. W.; Collins, S. R.; May, B. C. H.; Prusiner, S. B.; Weissman, J.; Shoichet, B. K., Small-molecule aggregates inhibit amyloid polymerization. *Nat. Chem. Biol.* **2008**, 4 (3), 197-199.
35. Frenkel, Y. V.; Clark, A. D.; Das, K.; Wang, Y.-H.; Lewi, P. J.; Janssen, P. A. J.; Arnold, E., Concentration and pH Dependent Aggregation of Hydrophobic Drug Molecules and Relevance to Oral Bioavailability. *J. Med. Chem.* **2005**, 48 (6), 1974-1983.
36. McGovern, S. L.; Helfand, B. T.; Feng, B.; Shoichet, B. K., A Specific Mechanism of Nonspecific Inhibition. *J. Med. Chem.* **2003**, 46 (20), 4265-4272.
37. Baell, J. B.; Holloway, G. A., New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *J. Med. Chem.* **2010**, 53 (7), 2719-40.
38. Baell, J. B., Feeling Nature's PAINS: Natural Products, Natural Product Drugs, and Pan Assay Interference Compounds (PAINS). *J. Nat. Prod.* **2016**, 79 (3), 616-28.
39. Dahlin, J. L.; Nissink, J. W.; Strasser, J. M.; Francis, S.; Higgins, L.; Zhou,

- H.; Zhang, Z.; Walters, M. A., PAINS in the Assay: Chemical Mechanisms of Assay Interference and Promiscuous Enzymatic Inhibition Observed during a Sulfhydryl-Scavenging HTS. *J. Med. Chem.* **2015**, *58* (5), 2091-113.
40. Baell, J. B.; Ferrins, L.; Falk, H.; Nikolakopoulos, G., PAINS: Relevance to tool compound discovery and fragment-based screening. *Aust. J. Chem.* **2013**, *66*.
41. Baell, J.; Walters, M. A., Chemistry: Chemical con artists foil drug discovery. *Nature* **2014**, *513*, 481.
42. Lackovic, K.; Lessene, G.; Falk, H.; Leuchowius, K. J.; Baell, J.; Street, I., A perspective on 10-years HTS experience at the Walter and Eliza Hall Institute of Medical Research- eighteen million assays and counting. *Comb. Chem. High Throughput Screening* **2014**, *17*, 241.
43. Saubern, S.; Guha, R.; Baell, J. B., KNIME Workflow to Assess PAINS Filters in SMARTS Format. Comparison of RDKit and Indigo Cheminformatics Libraries. *Mol Inform* **2011**, *30* (10), 847-50.
44. Baell, J. B.; Nissink, J. W. M., Seven Year Itch: Pan-Assay Interference Compounds (PAINS) in 2017—Utility and Limitations. *ACS Chemical Biology* **2018**, *13* (1), 36-44.
45. Ryan, A. J.; Gray, N. M.; Lowe, P. N.; Chung, C.-w., Effect of Detergent on “Promiscuous” Inhibitors. *J. Med. Chem.* **2003**, *46* (16), 3448-3451.
46. Feng, B. Y.; Shoichet, B. K., A detergent-based assay for the detection of promiscuous inhibitors. *Nat Protoc* **2006**, *1* (2), 550-3.
47. Coan, K. E. D.; Shoichet, B. K., Stability and equilibria of promiscuous aggregates in high protein milieus. *Molecular BioSystems* **2007**, *3* (3), 208.
48. Lor, L. A.; Schneck, J.; McNulty, D. E.; Diaz, E.; Brandt, M.; Thrall, S. H.; Schwartz, B., A Simple Assay for Detection of Small-Molecule Redox Activity. *J. Biomol. Screen.* **2007**, *12* (6), 881-890.
49. Wyatt, P. J., Light scattering and the absolute characterization of macromolecules. *Anal. Chim. Acta* **1993**, *272* (1), 1-40.
50. Miraglia, F.; Ricci, A.; Rota, L.; Colla, E., Subcellular localization of alpha-synuclein aggregates and their interaction with membranes. *Neural Regeneration Research* **2018**, *13* (7), 1136-1144.
51. Sung, J. J.; Pardeshi, N. N.; Mulder, A. M.; Mulligan, S. K.; Quispe, J.; On, K.; Carragher, B.; Potter, C. S.; Carpenter, J. F.; Schneemann, A., Transmission electron microscopy as an orthogonal method to characterize protein aggregates. *J. Pharm. Sci.* **2015**, *104* (2), 750-759.
52. Alves, V.; Muratov, E.; Capuzzi, S.; Politi, R.; Low, Y.; Braga, R.; Zakharov,

- A. V.; Sedykh, A.; Mokshyna, E.; Farag, S.; Andrade, C.; Kuz'min, V.; Fourches, D.; Tropsha, A., Alarms about structural alerts. *Green Chem.* **2016**, *18* (16), 4348-4360.
53. Mannhold, R.; Ostermann, C., Prediction of Log P with Substructure-Based Methods. In *Molecular Drug Properties*, Mannhold, R., Ed. 2007; pp 357-379.
54. Bruns, R. F.; Watson, I. A., Rules for identifying potentially reactive or promiscuous compounds. *J. Med. Chem.* **2012**, *55* (22), 9763-72.
55. Salmina, E. S.; Haider, N.; Tetko, I. V., Extended Functional Groups (EFG): An Efficient Set for Chemical Characterization and Structure-Activity Relationship Studies of Chemical Compounds. *Molecules* **2016**, *21* (1), 1.
56. Sushko, I.; Salmina, E.; Potemkin, V. A.; Poda, G.; Tetko, I. V., ToxAlerts: A Web Server of Structural Alerts for Toxic Chemicals and Compounds with Potential Adverse Reactions. *J. Chem. Inf. Model.* **2012**, *52* (8), 2310-6.
57. Ajay, A.; Walters, W. P.; Murcko, M. A., Can we learn to distinguish between "drug-like" and "non drug-like" molecules? *J. Med. Chem.* **1998**, *41* (18), 3314-24.
58. Walters, W. P.; Murcko, M. A., Prediction of 'drug-likeness'. *Adv. Drug Del. Rev.* **2002**, *54* (3), 255-271.
59. Hann, M.; Hudson, B.; Lewell, X.; Lively, R.; Miller, L.; Ramsden, N., Strategic Pooling of Compounds for High-Throughput Screening. *J. Chem. Inf. Comput. Sci.* **1999**, *39* (5), 897-902.
60. Percival, M. D.; Ouellet, M.; Campagnolo, C.; Claveau, D.; Li, C., Inhibition of Cathepsin K by Nitric Oxide Donors: Evidence for the Formation of Mixed Disulfides and a Sulfenic Acid. *Biochemistry* **1999**, *38* (41), 13574-13583.
61. Devine, S. M.; Mulcair, M. D.; Debono, C. O.; Leung, E. W.; Nissink, J. W.; Lim, S. S.; Chandrashekar, I. R.; Vazirani, M.; Mohanty, B.; Simpson, J. S.; Baell, J. B.; Scammells, P. J.; Norton, R. S.; Scanlon, M. J., Promiscuous 2-aminothiazoles (PrATs): a frequent hitting scaffold. *J. Med. Chem.* **2015**, *58* (3), 1205-14.
62. Walters, W. P.; Stahl, M. T.; Murcko, M. A., Virtual screening—an overview. *Drug Discovery Today* **1998**, *3* (4), 160-178.
63. Capuzzi, S. J.; Muratov, E. N.; Tropsha, A., Phantom PAINS: Problems with the Utility of Alerts for Pan-Assay Interference Compounds. *J. Chem. Inf. Model.* **2017**, *57*, 417.
64. Yang, J. J.; Ursu, O.; Lipinski, C. A.; Sklar, L. A.; Oprea, T. I.; Bologa, C. G., Badapple: promiscuity patterns from noisy evidence. *J. Cheminf.* **2016**, *8*,

- 29.
65. Schäfer, T.; Kriege, N.; Humbeck, L.; Klein, K.; Koch, O.; Mutzel, P., Scaffold Hunter: a comprehensive visual analytics framework for drug discovery. *J. Cheminformatics* **2017**, *9* (1 %@ 1758-2946), 28.
66. Varnek, A.; Fourches, D.; Horvath, D.; Klimchuk, O.; Gaudin, C.; Vayer, P.; Solov'ev, V.; Hoonakker, F.; Tetko, I. V.; Marcou, G., ISIDA- Platform for virtual screening based on fragment and pharmacophoric descriptors. *Curr Comput Aided Drug Des* **2008**, *4* (3), 191-198.
67. Taylor, R.; Cole, J. C.; Cosgrove, D. A.; Gardiner, E. J.; Gillet, V. J.; Korb, O., Development and validation of an improved algorithm for overlaying flexible molecules. *Journal of Computer-Aided Molecular Design* **2012**, *26* (4), 451-472.
68. Vorberg, S.; Tetko, I. V., Modeling the Biodegradability of Chemical Compounds Using the Online CHEmical Modeling Environment (OCHEM). *Mol Inform* **2014**, *33* (1), 73-85.
69. Sushko, I.; Novotarskyi, S.; Korner, R.; Pandey, A. K.; Rupp, M.; Teetz, W.; Brandmaier, S.; Abdelaziz, A.; Prokopenko, V. V.; Tanchuk, V. Y.; Todeschini, R.; Varnek, A.; Marcou, G.; Ertl, P.; Potemkin, V.; Grishina, M.; Gasteiger, J.; Schwab, C.; Baskin, I. I.; Palyulin, V. A.; Radchenko, E. V.; Welsh, W. J.; Kholodovych, V.; Chekmarev, D.; Cherkasov, A.; Aires-de-Sousa, J.; Zhang, Q. Y.; Bender, A.; Nigsch, F.; Patiny, L.; Williams, A.; Tkachenko, V.; Tetko, I. V., Online chemical modeling environment (OCHEM): web platform for data storage, model development and publishing of chemical information. *J. Comput. Aided Mol. Des.* **2011**, *25* (6), 533-54.
70. Stork, C.; Wagner, J.; Friedrich, N. O.; de Bruyn Kops, C.; Šícho, M.; Kirchmair, J., Hit Dexter: A Machine-Learning Model for the Prediction of Frequent Hitters. *ChemMedChem* **2017**, *13* (6), 564-571.
71. Koza, J. R.; Bennett, F. H.; Andre, D.; Keane, M. A., Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. In *Artificial Intelligence in Design '96*, Gero, J. S.; Sudweeks, F., Eds. Springer Netherlands: Dordrecht, 1996; pp 151-170.
72. PubChem PubChem BioAssay Database. <http://pubchem.ncbi.nlm.nih.gov> (accessed 4 August 2016).
73. Walt, S. v. d.; Colbert, S. C.; Varoquaux, G., The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering* **2011**, *13* (2), 22-30.
74. Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Jarrod Millman, K.; Mayorov,

- N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; Contributors, S., SciPy 1.0--Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints* **2019**, arXiv:1907.10121.
75. McKinney, W. In *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, 2010; van der Walt, S.; Millman, J., Eds. pp 51- 56-56.
76. Hunter, J. D., Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* **2007**, *9* (3), 90-95.
77. Landrum, G. A. RDKit, Open-Source Cheminformatics. <http://www.rdkit.org>.
78. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E., Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825-2830.
79. Perez, F.; Granger, B. E., IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering* **2007**, *9* (3), 21-29.
80. Wolber, G.; Langer, T., LigandScout: 3-D Pharmacophores Derived from Protein-Bound Ligands and Their Use as Virtual Screening Filters. *J. Chem. Inf. Model.* **2005**, *45* (1), 160-169.
81. Morris, G. M.; Huey, R.; Lindstrom, W.; Sanner, M. F.; Belew, R. K.; Goodsell, D. S.; Olson, A. J., AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *J. Comput. Chem.* **2009**, *30* (16), 2785-2791.
82. Trott, O.; Olson, A. J., AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multi-threading. *J. Comput. Chem.* **2010**, *31* (2), 455-461.
83. OCHEM On-line Chemical Database and Modelling Environment (OCHEM). <http://www.ochem.eu> (accessed 28 February 2017).
84. Abdi, H.; Williams, L. J., *Principal component analysis*. John Wiley & Sons, Inc.: 2010; Vol. 2, p 433-459.
85. Jolliffe, I., Principal Component Analysis. In *International Encyclopedia of Statistical Science*, Lovric, M., Ed. Springer Berlin Heidelberg: Berlin, Heidelberg, 2011; pp 1094-1096.
86. Yan, X.; Su, X. G., *Linear Regression Analysis: Theory and Computing*. World

- Scientific Publishing Co., Inc.: 2009.
87. Schneider, A.; Hommel, G.; Blettner, M., Linear regression analysis: part 14 of a series on evaluation of scientific publications. *Deutsches Arzteblatt international* **2010**, *107* (44), 776-782.
 88. Tibshirani, R. J., *A general framework for fast stagewise algorithms*. JMLR.org: 2015; Vol. 16, p 2543–2588.
 89. Mucherino, A.; Papajorgji, P. J.; Pardalos, P. M., k-Nearest Neighbor Classification. In *Data Mining in Agriculture*, Mucherino, A.; Papajorgji, P. J.; Pardalos, P. M., Eds. Springer New York: New York, NY, 2009; pp 83-106.
 90. Chavan, S.; Abdelaziz, A.; Wiklander, J. G.; Nicholls, I. A., A k-nearest neighbor classification of hERG K(+) channel blockers. *J. Comput. Aided Mol. Des.* **2016**, *30* (3), 229-36.
 91. Khan, M.; Ding, Q.; Perrizo, W., *k-nearest Neighbor Classification on Spatial Data Streams Using P-trees*. Springer-Verlag: 2002; p 517–518.
 92. Hearst, M. A., *Support Vector Machines*. IEEE Educational Activities Department: 1998; Vol. 13, p 18–28.
 93. Steinwart, I.; Christmann, A., *Support Vector Machines*. Springer Publishing Company, Incorporated: 2008.
 94. Cheng, T.; Li, Q.; Wang, Y.; Bryant, S. H., Binary classification of aqueous solubility using support vector machines with reduction and recombination feature selection. *J. Chem. Inf. Model.* **2011**, *51* (2), 229-36.
 95. Wu, Y.; Wang, G., Machine Learning Based Toxicity Prediction: From Chemical Structural Description to Transcriptome Analysis. *Int J Mol Sci* **2018**, *19* (8).
 96. Lind, P.; Maltseva, T., Support vector machines for the estimation of aqueous solubility. *J. Chem. Inf. Comput. Sci.* **2003**, *43* (6), 1855-1859.
 97. Liu, Z.; Xu, H., Kernel Parameter Selection for Support Vector Machine Classification. *Journal of Algorithms & Computational Technology* **2014**, *8* (2), 163-177.
 98. Müller, K. R.; Mika, S.; Rätsch, G.; Tsuda, K.; Schölkopf, B., An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Networks* **2001**, *12* (2), 181-201.
 99. Schölkopf, B.; Smola, A. J., *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT Press: Cambridge, Mass., 2002; p xviii, 626 p.
 100. Chang, C.-C.; Lin, C.-J., LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* **2011**, *2* (3), 27:1-

- 27:27.
101. Chang, C. C.; Lin, C. J. LIBSVM: a Library for Support Vector Machines, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>. (accessed November 10, 2015).
 102. Suykens, J. A. K.; Vandewalle, J., Least Squares Support Vector Machine Classifiers. *Neural Processing Letters* **1999**, *9* (3), 293-300.
 103. Podgorelec, V.; Kokol, P.; Stiglic, B.; Rozman, I., *Decision Trees: An Overview and Their Use in Medicine*. Plenum Press: 2002; Vol. 26, p 445–463.
 104. Quinlan, J. R., *Induction of Decision Trees*. Kluwer Academic Publishers: 1986; Vol. 1, p 81–106.
 105. Bel, L.; Allard, D.; Laurent, J. M.; Cheddadi, R.; Bar-Hen, A., *CART algorithm for spatial data: Application to environmental and ecological data*. Elsevier Science Publishers B. V.: 2009; Vol. 53, p 3082–3093.
 106. Boulesteix, A. L.; Tutz, G.; Strimmer, K., A CART-based approach to discover emerging patterns in microarray data. *Bioinformatics* **2003**, *19* (18), 2465-72.
 107. Xiaohu, W.; Lele, W.; Nianfeng, L., An Application of Decision Tree Based on ID3. *Physics Procedia* **2012**, *25*, 1017-1021.
 108. Quinlan, J. R., *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1993.
 109. Kretschmann, E.; Fleischmann, W.; Apweiler, R., Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on SWISS-PROT. *Bioinformatics* **2001**, *17* (10), 920-6.
 110. Novotarskyi, S.; Sushko, I.; Korner, R.; Pandey, A. K.; Tetko, I. V., A comparison of different QSAR approaches to modeling CYP450 1A2 inhibition. *J. Chem. Inf. Model.* **2011**, *51* (6), 1271-80.
 111. Tetko, I. V.; Novotarskyi, S.; Sushko, I.; Ivanov, V.; Petrenko, A. E.; Dieden, R.; Lebon, F.; Mathieu, B., Development of dimethyl sulfoxide solubility models using 163 000 molecules: using a domain applicability metric to select more reliable predictions. *J. Chem. Inf. Model.* **2013**, *53* (8), 1990-2000.
 112. Chen, G.; Peijnenburg, W. J. G. M.; Kovalishyn, V.; Vijver, M. G., Development of nanostructure-activity relationships assisting the nanomaterial hazard categorization for risk assessment and regulatory decision-making. *RSC Advances* **2016**, *6* (57), 52227-52235.
 113. Breiman, L., Random forests. *Machine Learning* **2001**, *45* (1), 5-32.
 114. Kovdienko, N. A.; Polishchuk, P. G.; Muratov, E. N.; Artemenko, A. G.;

- Kuz'min, V. E.; Gorb, L.; Hill, F.; Leszczynski, J., Application of random forest and multiple linear regression techniques to QSPR prediction of an aqueous solubility for military compounds. *Mol. Inform.* **2010**, *29* (5), 394-406.
115. Liaw, A.; Wiener, M., Classification and Regression by RandomForest. *R News* **2002**, *2*, 18-22.
116. Palmer, D. S.; O'Boyle, N. M.; Glen, R. C.; Mitchell, J. B. O., Random forest models to predict aqueous solubility. *J. Chem. Inf. Model.* **2007**, *47* (1), 150-158.
117. Sheridan, R. P., Using random forest to model the domain applicability of another random forest model. *J. Chem. Inf. Model.* **2013**, *53* (11), 2837-50.
118. Freund, Y.; Schapire, R. E., Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteen National Conference*, Saitta, L., Ed. Morgan Kaufmann: 1996; pp 148-156.
119. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.-Y. In *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*, Advances in Neural Information Processing Systems 30, 2017; pp 3146-3154.
120. Chen, T.; Guestrin, C., XGBoost: A Scalable Tree Boosting System. *ArXiv e-prints* **2016**, *1603*, arXiv:1603.02754.
121. Sheridan, R. P.; Wang, W. M.; Liaw, A.; Ma, J.; Gifford, E. M., Extreme Gradient Boosting as a Method for Quantitative Structure-Activity Relationships. *J. Chem. Inf. Model.* **2016**, *56* (12), 2353-2360.
122. Halder, A. K., Finding the structural requirements of diverse HIV-1 protease inhibitors using multiple QSAR modelling for lead identification. *SAR QSAR Environ. Res.* **2018**, *29* (11), 911-933.
123. Hu, T.; Song, T., Research on XGboost academic forecasting and analysis modelling. *Journal of Physics: Conference Series* **2019**, *1324*, 012091.
124. Hassoun, M. H., *Fundamentals of Artificial Neural Networks*. MIT Press: 1995.
125. Browne, A., Representation and Extrapolation in Multi-Layer Perceptrons. *Neural Comput.* **2001**, *in press*.
126. Gonzalez-Arjona, D.; Lopez-Perez, G.; Gonzalez, A. G., Non-linear QSAR modeling by using multilayer perceptron feedforward neural networks trained by back-propagation. *Talanta* **2002**, *56* (1), 79-90.
127. Mateos, A.; Dopazo, J.; Jansen, R.; Tu, Y.; Gerstein, M.; Stolovitzky, G., Systematic learning of gene functional classes from DNA array expression data by using multilayer perceptrons. *Genome Res.* **2002**, *12* (11), 1703-

-
- 15.
128. Livingstone, D. J.; Manallack, D. T.; Tetko, I. V., Data modelling with neural networks: advantages and limitations. *Journal of computer-aided molecular design* **1997**, *11* (2), 135-42.
129. Tetko, I. V.; Livingstone, D. J.; Luik, A. I., Neural network studies. 1. Comparison of overfitting and overtraining. *Journal of Chemical Information & Computer Sciences* **1995**, *35* (5), 826-833.
130. Kovalishyn, V. V.; Tetko, I. V.; Luik, A. I.; Kholodovych, V. V.; Villa, A. E. P.; Livingstone, D. J., Neural network studies. 3. Variable selection in the cascade-correlation learning architecture. *J. Chem. Inf. Comput. Sci.* **1998**, *38* (4), 651-659.
131. Glorot, X.; Bengio, Y., Understanding the difficulty of training deep feed-forward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Yee Whye, T.; Mike, T., Eds. PMLR: Proceedings of Machine Learning Research, 2010; Vol. 9, pp 249--256.
132. *Backpropagation: theory, architectures, and applications*. L. Erlbaum Associates Inc.: 1995.
133. Tetko, I. V., Associative neural network. *Methods Mol Biol* **2008**, *458*, 185-202.
134. Tetko, I. V., Neural network studies. 4. Introduction to associative neural networks. *J. Chem. Inf. Comput. Sci.* **2002**, *42* (3), 717-728.
135. Villa, A. E.; Tetko, I. V.; Dutoit, P.; De Ribaupierre, Y.; De Ribaupierre, F., Corticofugal modulation of functional connectivity within the auditory thalamus of rat, guinea pig and cat revealed by cooling deactivation. *J. Neurosci. Methods* **1999**, *86* (2), 161-178.
136. Tetko, I. V.; Bruneau, P., Application of ALOGPS to predict 1-octanol/water distribution coefficients, logP, and logD, of AstraZeneca in-house database. *J. Pharm. Sci.* **2004**, *93* (12), 3103-3110.
137. Tetko, I. V.; Tanchuk, V. Y., Application of associative neural networks for prediction of lipophilicity in ALOGPS 2.1 program. *Journal of Chemical Information & Computer Sciences* **2002**, *42* (5), 1136-1145.
138. Tetko, I. V., Application of Associative Neural Networks for Prediction of Physico-Chemical Properties. In *EuroQSAR2002 Designing Drugs and Crop Protectants: processes, problems and solutions*, Ford, M.; Livingstone, D.; Dearden, J.; Van de Waterbeemd, H., Eds. Blackwell Publishing: Bournemouth, UK, 2003; pp 199-203.
139. Tetko, I. V. Associative Neural Networks: an innovative method to pre-

- dict physico-chemical, biological and ADMETox properties. Université de Strasbourg, Strasbourg, France, 2011.
140. Varnek, A.; Kireeva, N.; Tetko, I. V.; Baskin, I. I.; Solov'ev, V. P., Exhaustive QSPR Studies of a Large Diverse Set of Ionic Liquids: How Accurately Can We Predict Melting Points? *J. Chem. Inf. Model.* **2007**, *47* (3), 1111-1122.
 141. Ghosh, D.; Koch, U.; Hadian, K.; Sattler, M.; Tetko, I. V., Luciferase Advisor: High-Accuracy Model To Flag False Positive Hits in Luciferase HTS Assays. *J. Chem. Inf. Model.* **2018**.
 142. Kovalishyn, V.; Abramenko, N.; Kopernyk, I.; Charochkina, L.; Metelytsia, L.; Tetko, I. V.; Peijnenburg, W.; Kustov, L., Modelling the toxicity of a large set of metal and metal oxide nanoparticles using the OCHEM platform. *Food Chem. Toxicol.* **2018**, *112*, 507-517.
 143. Nizami, B.; Tetko, I. V.; Koorbanally, N. A.; Honarparvar, B., QSAR models and scaffold-based analysis of non-nucleoside HIV RT inhibitors. *Chemo-metrics Intellig. Lab. Syst.* **2015**, *148*, 134-144.
 144. LeCun, Y.; Bengio, Y.; Hinton, G., Deep learning. *Nature* **2015**, *521* (7553), 436-44.
 145. Goh, G. B.; Hodas, N. O.; Vishnu, A., Deep learning for computational chemistry. *J. Comput. Chem.* **2017**.
 146. Gawehn, E.; Hiss, J. A.; Schneider, G., Deep Learning in Drug Discovery. *Mol. Inform.* **2016**, *35* (1), 3-14.
 147. Schmidhuber, J., Deep learning in neural networks: an overview. *Neural Netw* **2015**, *61*, 85-117.
 148. Ma, J.; Sheridan, R. P.; Liaw, A.; Dahl, G. E.; Svetnik, V., Deep neural nets as a method for quantitative structure-activity relationships. *J. Chem. Inf. Model.* **2015**, *55* (2), 263-74.
 149. Sosnin, S.; Karlov, D.; Tetko, I. V.; Fedorov, M. V., Comparative Study of Multitask Toxicity Modeling on a Broad Chemical Space. *J. Chem. Inf. Model.* **2019**, *59* (3), 1062-1072.
 150. Kim, Y. In *Convolutional Neural Networks for Sentence Classification*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014//; Association for Computational Linguistics: 2014; pp 1746-1751.
 151. Yamashita, R.; Nishio, M.; Do, R. K. G.; Togashi, K., Convolutional neural networks: an overview and application in radiology. *Insights into Imaging* **2018**, *9* (4), 611-629.
 152. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; ., J. C.; et al., *Recent advances in convolutional neural net-*

- works. Elsevier Science Inc.: 2018; Vol. 77, p 354–377.
153. Tokui, S.; Oono, K. In *Chainer : a Next-Generation Open Source Framework for Deep Learning*, 2015.
 154. Democratizing Deep-Learning for Drug Discovery, Quantum Chemistry, Materials Science and Biology. GitHub: 2016.
 155. Ramsundar, B.; Eastman, P.; Walters, P.; Pande, V.; Leswing, K.; Wu, Z., *Deep Learning for the Life Sciences*. O'Reilly Media: 2019.
 156. Gasteiger, J., Of molecules and humans. *J. Med. Chem.* **2006**, *49* (22), 6429-34.
 157. Tetko, I. V.; Tanchuk, V. Y.; Kasheva, T. N.; Villa, A. E. P., Estimation of Aqueous Solubility of Chemical Compounds Using E-State Indices. *J. Chem. Inf. Comput. Sci.* **2001**, *41* (6), 1488-1493.
 158. Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E., The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics. *J. Chem. Inf. Comput. Sci.* **2003**, *43* (2), 493-500.
 159. Willighagen, E. L.; Mayfield, J. W.; Alvarsson, J.; Berg, A.; Carlsson, L.; Jeliazkova, N.; Kuhn, S.; Pluskal, T.; Rojas-Chertó, M.; Spjuth, O.; Torrance, G.; Evelo, C. T.; Guha, R.; Steinbeck, C., The Chemistry Development Kit (CDK) v2.0: atom typing, depiction, molecular formulas, and substructure searching. *J. Cheminformatics* **2017**, *9* (1), 33.
 160. Winter, R.; Montanari, F.; Noé, F.; Clevert, D.-A., Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chem. Sci.* **2019**.
 161. Myrdal, P. B.; Manka, A. M.; Yalkowsky, S. H., AQUAFAC 3: aqueous functional group activity coefficients; application to the estimation of aqueous solubility. *Chemosphere* **1995**, *30* (9), 1619-1637.
 162. Todeschini, R.; Consonni, V., In *Handbook of Molecular Descriptors*, Wiley-VCH Verlag GmbH: 2000.
 163. Skvortsova, M. I.; Baskin, I. I.; Skvortsov, L. A.; Palyulin, V. A.; Zefirov, N. S.; Stankevich, I. V., Chemical graphs and their basis invariants. *Journal of Molecular Structure-Theochem* **1999**, *466*, 211-217.
 164. Ruggiu, F.; Marcou, G.; Varnek, A.; Horvath, D., ISIDA Property-Labelled Fragment Descriptors. *Mol Inform* **2010**, *29* (12), 855-68.
 165. Alexandre, V.; Denis, F.; Dragos, H.; Olga, K.; Cedric, G.; Philippe, V.; Vitaly, S. e.; Frank, H.; Igor, V. T.; Gilles, M., ISIDA - Platform for Virtual Screening Based on Fragment and Pharmacophoric Descriptors. *Curr Comput Aided Drug Des* **2008**, *4* (3), 191-198.

-
166. Potemkin, V. A.; Grishina, M. A., A new paradigm for pattern recognition of drugs. *J. Comput. Aided Mol. Des.* **2008**, *22* (6-7), 489-505.
167. Moriwaki, H.; Tian, Y.-S.; Kawashita, N.; Takagi, T., Mordred: a molecular descriptor calculator. *J. Cheminformatics* **2018**, *10* (1), 4.
168. Bultinck, P.; Langenaeker, W.; Lahorte, P.; De Proft, F.; Geerlings, P.; Van Alsenoy, C.; Tollenaere, J. P., The Electronegativity Equalization Method II: Applicability of Different Atomic Charge Schemes. *The Journal of Physical Chemistry A* **2002**, *106* (34), 7895-7901.
169. Bultinck, P.; Langenaeker, W.; Carbó-Dorca, R.; Tollenaere, J. P., Fast Calculation of Quantum Chemical Molecular Descriptors from the Electronegativity Equalization Method. *J. Chem. Inf. Comput. Sci.* **2003**, *43* (2), 422-428.
170. Tetko, I. V.; M. Lowe, D.; Williams, A. J., The development of models to predict melting and pyrolysis point data associated with several hundred thousand compounds mined from PATENTS. *J. Cheminformatics* **2016**, *8*, 2.
171. Salmina, E. S.; Haider, N.; Tetko, I. V., Extended Functional Groups (EFG): An Efficient Set for Chemical Characterization and Structure-Activity Relationship Studies of Chemical Compounds. *Molecules* **2015**, *21* (1), E1.
172. Haider, N., Functionality Pattern Matching as an Efficient Complementary Structure/Reaction Search Tool: an Open-Source Approach. *Molecules* **2010**, *15* (8), 5079-5092.
173. Cherkasov, A., 'Inductive' Descriptors: 10 Successful Years in QSAR. *Current Computer - Aided Drug Design* **2005**, *1* (1), 21-42.
174. Cherkasov, A., Inductive QSAR Descriptors. Distinguishing Compounds with Antibacterial Activity by Artificial Neural Networks. *International Journal of Molecular Sciences* **2005**, *6* (1), 63-86.
175. Capecchi, A.; Probst, D.; Reymond, J.-L., One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome. *J. Cheminformatics* **2020**, *12* (1), 43.
176. Ognichenko, L. N.; Kuz'min, V. E.; Artemenko, A. G., New Structural Descriptors of Molecules on the Basis of Symbiosis of the Informational Field Model and Simplex Representation of Molecular Structure. *QSAR & Combinatorial Science* **2009**, *28* (9), 939-945.
177. Masand, V. H.; Rastija, V., PyDescriptor: A new PyMOL plugin for calculating thousands of easily understandable molecular descriptors. *Chemometrics Intellig. Lab. Syst.* **2017**, *169*, 12-18.
178. Plante, J.; Werner, S., JPlogP: an improved logP predictor trained using

- predicted data. *J. Cheminformatics* **2018**, *10* (1), 61.
179. Matthews, B. W., Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure* **1975**, *405* (2), 442-451.
180. Hajian-Tilaki, K., Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation. *Caspian. J. Intern. Med.* **2013**, *4* (2), 627-635.
181. Breiman, L., Bagging Predictors. *Machine Learning* **1996**, *24* (2), 123-140.
182. Netzeva, T. I.; Worth, A.; Aldenberg, T.; Benigni, R.; Cronin, M. T.; Gramatica, P.; Jaworska, J. S.; Kahn, S.; Klopman, G.; Marchant, C. A.; Myatt, G.; Nikolova-Jeliazkova, N.; Patlewicz, G. Y.; Perkins, R.; Roberts, D.; Schultz, T.; Stanton, D. W.; van de Sandt, J. J.; Tong, W.; Veith, G.; Yang, C., Current status of methods for defining the applicability domain of (quantitative) structure-activity relationships. The report and recommendations of ECVAM Workshop 52. *Altern Lab Anim* **2005**, *33* (2), 155-173.
183. Sushko, I.; Novotarskyi, S.; Korner, R.; Pandey, A. K.; Cherkasov, A.; Li, J.; Gramatica, P.; Hansen, K.; Schroeter, T.; Muller, K. R.; Xi, L.; Liu, H.; Yao, X.; Oberg, T.; Hormozdiari, F.; Dao, P.; Sahinalp, C.; Todeschini, R.; Polishchuk, P.; Artemenko, A.; Kuz'min, V.; Martin, T. M.; Young, D. M.; Fourches, D.; Muratov, E.; Tropsha, A.; Baskin, I.; Horvath, D.; Marcou, G.; Muller, C.; Varnek, A.; Prokopenko, V. V.; Tetko, I. V., Applicability domains for classification problems: Benchmarking of distance to models for Ames mutagenicity set. *J. Chem. Inf. Model.* **2010**, *50* (12), 2094-111.
184. Sushko, I. Applicability domain of QSAR models. Technical University of Munich, Munich, 2011.
185. Thorne, N.; Inglese, J.; Auld, D. S., Illuminating insights into firefly luciferase and other bioluminescent reporters used in chemical biology. *Chem. Biol.* **2010**, *17* (6), 646-57.
186. Wang, T. T. Y., β -Naphthoflavone, an Inducer of Xenobiotic Metabolizing Enzymes, Inhibits Firefly Luciferase Activity. *Anal. Biochem.* **2002**, *304* (1), 122-126.
187. Bakhtiarova, A.; Taslimi, P.; Elliman, S. J.; Kosinski, P. A.; Hubbard, B.; Kavana, M.; Kemp, D. M., Resveratrol inhibits firefly luciferase. *Biochem. Biophys. Res. Commun.* **2006**, *351* (2), 481-484.
188. Auld, D. S.; Southall, N. T.; Jadhav, A.; Johnson, R. L.; Diller, D. J.; Simonov, A.; Austin, C. P.; Inglese, J., Characterization of chemical libraries for luciferase inhibitory activity. *J. Med. Chem.* **2008**, *51* (8), 2372-86.
189. Sadowski, J.; Gasteiger, J.; Klebe, G., Comparison of Automatic Three-Dimensional Model Builders Using 639 X-ray Structures. *J. Chem. Inf. Com-*

- put. Sci.* **1994**, 34 (4), 1000-1008.
190. Schrödinger, L. L. C., The PyMOL Molecular Graphics System, Version 1.8.6.0. In *The PyMOL Molecular Graphics System, Version 1.8.6.0*, 2015.
191. Sushko, I.; Novotarskyi, S.; Körner, R.; Pandey, A. K.; Rupp, M.; Teetz, W.; Brandmaier, S.; Abdelaziz, A.; Prokopenko, V. V.; Tanchuk, V. Y.; Todeschini, R.; Varnek, A.; Marcou, G.; Ertl, P.; Potemkin, V.; Grishina, M.; Gasteiger, J.; Schwab, C.; Baskin, I. I.; Palyulin, V. A.; Radchenko, E. V.; Welsh, W. J.; Kholodovych, V.; Chekmarev, D.; Cherkasov, A.; Aires-de-Sousa, J.; Zhang, Q.-Y.; Bender, A.; Nigsch, F.; Patiny, L.; Williams, A.; Tkachenko, V.; Tetko, I. V., Online chemical modeling environment (OCHEM): web platform for data storage, model development and publishing of chemical information. *Journal of Computer-Aided Molecular Design* **2011**, 25 (6), 533-554.
192. Tetko, I. V., Associative neural network. *Neural Processing Letters* **2002**, 16 (2), 187-199.
193. Schuffenhauer, A.; Ertl, P.; Roggo, S.; Wetzel, S.; Koch, M. A.; Waldmann, H., The scaffold tree—visualization of the scaffold universe by hierarchical scaffold classification. *J. Chem. Inf. Model.* **2007**, 47.
194. Feng, B. Y.; Shelat, A.; Doman, T. N.; Guy, R. K.; Shoichet, B. K., High-throughput assays for promiscuous inhibitors. *Nat. Chem. Biol.* **2005**, 1 (3), 146-148.
195. Irwin, J. J.; Duan, D.; Torosyan, H.; Doak, A. K.; Ziebart, K. T.; Sterling, T.; Tumanian, G.; Shoichet, B. K., An Aggregation Advisor for Ligand Discovery. *J. Med. Chem.* **2015**, 58 (17), 7076-7087.
196. Pearce, B. C.; Sofia, M. J.; Good, A. C.; Drexler, D. M.; Stock, D. A., An empirical process for the design of high-throughput screening deck filters. *J. Chem. Inf. Model.* **2006**, 46 (3), 1060-8.
197. Yasgar, A.; Jadhav, A.; Simeonov, A.; Coussens, N. P., AlphaScreen-Based Assays: Ultra-High-Throughput Screening for Small-Molecule Inhibitors of Challenging Enzymes and Protein-Protein Interactions. *Methods Mol. Biol.* **2016**, 1439, 77-98.
198. Eglen, R. M.; Reisine, T.; Roby, P.; Rouleau, N.; Illy, C.; Bossé, R.; Bielefeld, M., The use of AlphaScreen technology in HTS: current status. *Current chemical genomics* **2008**, 1, 2-10.
199. Brenke, J. K.; Salmina, E. S.; Ringelstetter, L.; Dornauer, S.; Kuzikov, M.; Rothenaigner, I.; Schorpp, K.; Giehler, F.; Gopalakrishnan, J.; Kieser, A.; Gul, S.; Tetko, I. V.; Hadian, K., Identification of Small-Molecule Frequent Hitters of Glutathione S-Transferase-Glutathione Interaction. *J. Biomol.*

- Screen*. **2016**, *21* (6), 596-607.
200. Mayfield, J. W.; Sayle, R. A., Technical implications of new IUPAC elements in cheminformatics. *J. Cheminformatics* **2017**, *9* (1), 10.
201. Kim, S.; Chen, J.; Cheng, T.; Gindulyte, A.; He, J.; He, S.; Li, Q.; Shoemaker, B. A.; Thiessen, P. A.; Yu, B.; Zaslavsky, L.; Zhang, J.; Bolton, E. E., PubChem 2019 update: improved access to chemical data. *Nucleic Acids Res.* **2019**, *47* (D1), D1102-D1109.
202. Karpov, P.; Godin, G.; Tetko, I. V., Transformer-CNN: Swiss knife for QSAR modeling and interpretation. *J. Cheminformatics* **2020**, *12* (1), 17.
203. Sosnin, S.; Vashurina, M.; Withnall, M.; Karpov, P.; Fedorov, M.; Tetko, I. V., A Survey of Multi-task Learning Methods in Chemoinformatics. *Mol. Inform.* **2019**, *38* (4), e1800108.
204. Ghosh, D.; Tetko, I.; Klebl, B.; Nussbaumer, P.; Koch, U. In *Analysis and Modelling of False Positives in GPCR Assays*, Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions, Cham, 2019//; Tetko, I. V.; Kůrková, V.; Karpov, P.; Theis, F., Eds. Springer International Publishing: Cham, 2019; pp 764-770.
205. Hauser, A. S.; Attwood, M. M.; Rask-Andersen, M.; Schioth, H. B.; Gloriam, D. E., Trends in GPCR drug discovery: new agents, targets and indications. *Nat. Rev. Drug Discov.* **2017**, *16* (12), 829-842.
206. Fredriksson, R.; Lagerström, M. C.; Lundin, L.-G.; Schiöth, H. B., The G-Protein-Coupled Receptors in the Human Genome Form Five Main Families. Phylogenetic Analysis, Paralogon Groups, and Fingerprints. *Mol. Pharmacol.* **2003**, *63* (6), 1256-1272.
207. Zhang, R.; Xie, X., Tools for GPCR drug discovery. *Acta Pharmacologica Sinica* **2012**, *33*, 372.
208. Hu, G. M.; Mai, T. L.; Chen, C. M., Visualizing the GPCR Network: Classification and Evolution. *Sci Rep* **2017**, *7* (1), 15495.
209. Stevens, R. C.; Cherezov, V.; Katritch, V.; Abagyan, R.; Kuhn, P.; Rosen, H.; Wüthrich, K., The GPCR Network: a large-scale collaboration to determine human GPCR structure and function. *Nature Reviews Drug Discovery* **2012**, *12*, 25.
210. Kim, S.; Chen, J.; Cheng, T.; Gindulyte, A.; He, J.; He, S.; Li, Q.; Shoemaker, B. A.; Thiessen, P. A.; Yu, B.; Zaslavsky, L.; Zhang, J.; Bolton, E. E., PubChem 2019 update: improved access to chemical data. *Nucleic Acids Res.* **2018**, *47* (D1), D1102-D1109.
211. Basith, S.; Cui, M.; Macalino, S. J. Y.; Park, J.; Clavio, N. A. B.; Kang, S.; Choi, S., Exploring G Protein-Coupled Receptors (GPCRs) Ligand Space via

Cheminformatics Approaches: Impact on Rational Drug Design. *Frontiers in pharmacology* **2018**, *9*, 128-128.

Appendix I: PCA Pipeline with Jupyter Notebook

This is an example of a pipeline built and executed in Jupyter Notebook for ingesting folder full of assay data, analysing the data to find all active and inactive datapoints, and performing a Principal Component Analysis to find patterns in the data.

To start with, we import the relevant packages, and initialize a few global variables.

```
In [ ]:  
  
import pandas as pd  
import os, pathlib, functools  
import numpy as np  
  
root_dir = "I:\\Backup\\Data\\Alphascreen\\Data\\public\\"  
his_dir = root_dir + "His\\"
```

A1.1. Reading the csv files

Then, we use `read_csv()` to read csv all the csv files in a given directory, and load them up as Pandas Dataframe object. We load up just the first 5 columns, as that is what we need. We also load everything as string, because there are mixed data in every column, and so pandas issues us a warning if we don't specify the datatype. Some columns are empty, and some unnecessary rows are present. We can clean the data and typecast properly post-import.

```
In [ ]:  
  
def read_csv_print_msg(filename):  
    print ("Reading {}".format(filename))  
    return pd.read_csv(filename, usecols = range(5), dtype = str)  
  
his_flist = [read_csv_print_msg(i) for i in pathlib.Path(his_dir).glob("*  
csv")]
```

```
Reading I:\Backup\Data\Alphascreen\Data\public\His\485360.csv  
Reading I:\Backup\Data\Alphascreen\Data\public\His\504333.csv  
Reading I:\Backup\Data\Alphascreen\Data\public\His\504339.csv  
Reading I:\Backup\Data\Alphascreen\Data\public\His\540317.csv  
Reading I:\Backup\Data\Alphascreen\Data\public\His\623870.csv  
Reading I:\Backup\Data\Alphascreen\Data\public\His\651724.csv  
Reading I:\Backup\Data\Alphascreen\Data\public\His\651725.csv
```

Taking a look at the data, we can see some rows we do not need. We will clean the dataframes next.

In []:

```
his_flist[0].head(10)
```

Out[]:

	PUBCHEM_RESULT_TAG	PUBCHEM_SID	PUBCHEM_CID	PUBCHEM_ACTIVITY_OUTCOME	PUBCHEM_ACTIVITY_SCORE
0	RESULT_TYPE	NaN	NaN	NaN	NaN
1	RESULT_DESCR	NaN	NaN	NaN	NaN
2	RESULT_UNIT	NaN	NaN	NaN	NaN
3	RESULT_IS_ACTIVE_CONCENTRATION	NaN	NaN	NaN	NaN
4	RESULT_ATTR_CONC_MICROMOL	NaN	NaN	NaN	NaN
5	1	842122	6602571	Inactive	0
6	2	842123	6602616	Inactive	0
7	3	842124	644371	Inactive	0
8	4	842125	6603132	Inactive	0
9	5	842126	2850911	Inactive	0

A1.2. Cleaning the Dataframes

From the output above, we can see that we do not need the first five rows. We also should use int datatype for all the columns except PUBCHEM_ACTIVITY_OUTCOME. We can also use the PUBCHEM_RESULT_TAG as the index for our dataframe, and thus removing duplicated columns.

In []:

```
def clean_dataframe(inp_df):
    inp_df = inp_df.iloc[5:].dropna()
    dtype_dict = {"PUBCHEM_RESULT_TAG": int, "PUBCHEM_SID": int,
                  "PUBCHEM_CID": int, "PUBCHEM_ACTIVITY_OUTCOME": str,
                  "PUBCHEM_ACTIVITY_SCORE": int}
    inp_df = inp_df.astype(dtype_dict)
    return inp_df.set_index("PUBCHEM_RESULT_TAG")

clean_his_flist = [clean_dataframe(i) for i in his_flist]
```

We can see that the dataframes are now clean

```
In []:  
clean_his_flist[0].head()
```

Out[]:

PUBCHEM_RE-SULT_TAG	PUBCHEM_SID	PUBCHEM_CID	PUBCHEM_ACTIVITY_OUTCOME	PUBCHEM_ACTIVITY_SCORE
1	842122	6602571	Inactive	0
2	842123	6602616	Inactive	0
3	842124	644371	Inactive	0
4	842125	6603132	Inactive	0
5	842126	2850911	Inactive	0

and has the proper datatypes

```
In []:  
clean_his_flist[0].dtypes
```

Out[]:

```
PUBCHEM_SID          int32  
PUBCHEM_CID          int32  
PUBCHEM_ACTIVITY_OUTCOME  object  
PUBCHEM_ACTIVITY_SCORE  int32  
dtype: object
```

A1.3. Getting the active and inactive Datapoints

Next we split the dataframes based on compound activity. Pandas provides excellent methods for filtering dataframes based on a column value, so that is what we will use.

```
In []:  
  
def getactives(df):  
    return df.loc[df["PUBCHEM_ACTIVITY_OUTCOME"] == "Active"]  
  
def getinactives(df):  
    return df.loc[df["PUBCHEM_ACTIVITY_OUTCOME"] == "Inactive"]  
  
his_flist_actives = [getactives(i) for i in clean_his_flist]  
his_flist_inactives = [getinactives(i) for i in clean_his_flist]
```

In []:

```
his_flist_actives[0].head(3)
```

Out[11]:

PUBCHEM_RESULT_TAG	PUBCHEM_SID	PUBCHEM_CID	PUBCHEM_ACTIVITY_OUTCOME	PUBCHEM_ACTIVITY_SCORE
221	842394	644662	Active	40
558	842823	5767374	Active	42
713	843005	767427	Active	84

In []:

```
his_flist_inactives[0].head(3)
```

Out[18]:

PUBCHEM_RESULT_TAG	PUBCHEM_SID	PUBCHEM_CID	PUBCHEM_ACTIVITY_OUTCOME	PUBCHEM_ACTIVITY_SCORE
1	842122	6602571	Inactive	0
2	842123	6602616	Inactive	0
3	842124	644371	Inactive	0

A1.4. Getting Assay Data Statistics

Now that we have all the different dataframe objects set up, we can easily generate a nice table detailing statistics for all the imported assay data.

Our dataframe objects has no information about the assay ID, which is also the filename. This has to be extracted from the filename in a separate step. We are relying on the fact the we never changed the order in which the files are imported. This works in this case, but may not be the best approach.

In []:

```
assaydetails = {}
for i, df in enumerate(his_flist):
    assayid = list(pathlib.Path(his_dir).glob("*.csv"))[i].name[:-4]
    activitystat = (len(getactives(df)), len(getinactives(df)))
    assaydetails[assayid] = activitystat
```

In []:

```
pd.DataFrame.from_dict(assaydetails, orient='index', columns=["actives", "inactives"])
```

Out[42]:

	actives	inactives
485360	1495	217165
504333	15779	312744
504339	17028	342184
540317	2150	371468
623870	2595	390301
651724	1654	328376
651725	2028	355075

A1.5. Merging the Dataframes

We need to get the compound IDs for all the compounds that shows activity. We will use this list of ids to download SMILES from the Pubchem compound sownload service. To do this, we need to merge all the dataframes in our active dataframe list. How we do this merge operation determines what we will get. If we want all unique compounds what has activity at least once, we can do an intersection operation between PUBCHEM_CID columns of all the dataframes in the list. If we do an union, we will get all datapoints, which we can use to count how many times a compound has shown activity. For our purposes, we will do an intersection.

```
In [ ]:
def getUniqueList(dfList):
    cidlists = [i["PUBCHEM_CID"] for i in dfList]
    reduced = functools.reduce(np.union1d, cidlists)
    return reduced

activelist = getUniqueList(his_flist_actives)
inactivelist = getUniqueList(his_flist_inactives)
```

Doing this we get a list of all the active compounds and all the inactive compounds

```
In [ ]:
print ("We have {} actives and {} inactives".format(len(activelist), len(inac-
tivelist)))

activelist, inactivelist
```

We have 27122 actives and 487608 inactives

Out[34]:

```
(array([ 86, 109, 253, ..., 56642969, 56835566, 56835586]),
array([ 6, 19, 40, ..., 73265308, 73265382, 85090241]))
```

A1.6. Exporting the lists

Now, we export these lists as text files that we can load into the PubChem Compound Download Service. This is the better way to work, as trying to access such quantity of data through PUG is not possible due to Request Volume Limitations.

```
In []:
with open("actives_list.txt", "w") as f:
    f.write("\n".join(map(str, activelist)))

with open("inactives_list.txt", "w") as f:
    f.write("\n".join(map(str, inactivelist)))
```

Continuing for here, we download the compound data, and read that data to continue the analysis. For convenience, this should be done in a separate Notebook.

```
In []:
import pandas as pd
from rdkit import Chem
from rdkit.Chem import rdMolDescriptors as desc
import numpy as np
from tqdm import tqdm

#root_dir = "I:\\Backup\\Data\\Alphascreen\\Data\\public\\"
root_dir = "D:\\FreqHitterProject\\Alphascreen\\Data\\public\\"
gst_dir = root_dir + "GST\\analyzed\\"
his_dir = root_dir + "His\\analyzed\\"
```

A1.7. Reading in the Smiles

After downloading the data from PubChem Compound Download Service, we need to read the data back in. We use Pandas once again, and set the CID column to be the index.

```
In []:
inactivesdf = pd.read_csv(his_dir + "inactives_smiles.csv", index_col =
"CID")
activesdf = pd.read_csv(his_dir + "actives.csv", index_col = "CID")
```

We verify the data has come in properly:

```
In []:
activesdf.head(5)
```

Out[]:


```
totalfp = fplist_actives + fplist_inactives
```

```
100%|          | 27061/27061 [00:24<00:00, 1122.29it/s]
100%|          | 462890/462890 [07:35<00:00, 1015.89it/s]
```

If we want, we can save the fingerprints to disk with using pickle

```
In []:
import pickle
# with open("activesfp_his", "rb") as f:
#     activesfp = pickle.load(f)
# with open("inactivesfp_his", "wb") as f:
#     pickle.dump(inactivesfp, f)
# with open("inactivesfp_his", "rb") as f:
#     inactivesfp = pickle.load(f)
```

Principal Component Analysis

With the fingerprints calculated, we can get started on modelling.

A1.9. Creating and Fitting the PCA

We use the PCA object from the decomposition module of Scikit-learn. We want to plot the data as a scatter plot, so two dimensions are needed. We fit the model to all the fingerprints. Given our volume of data (~ 500K datapoints), this process requires nearly 25GB of RAM at runtime. This is an important consideration. If the system does not have enough memory, a `MemoryException` will be raised.

```
In []:
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca.fit(totalfp)
```

The fitting process is very memory and processor intensive, so we should save the model to disk as a serialized object. The `joblib` module provides wrapper functions for this. The standard `pickle` module can also be used.

```
In []:
```

```
import joblib
joblib.dump(pca, "his-pca.pca")
pca = joblib.load("his-pca.pca")
```

```
C:\Users\dipan\.conda\envs\keras\lib\site-packages\sklearn\base.py:306: UserWarning:
Trying to unpickle estimator PCA from version 0.20.3 when using version 0.21.1. This
might lead to breaking code or invalid results. Use at your own risk.
UserWarning)
```

A1.10. Dimensionality Reduction

With the model ready, now we can express our Morgan Fingerprint data in two dimensions. The PCA object has a transform method which does just this.

In []:

```
X_pca_actives = pca.transform(fplist_actives)
X_pca_inactives = pca.transform(fplist_inactives)
```

This process gives us a numpy array with the transformed data.

In []:

```
X_pca_actives
```

Out[11]:

```
array([[ -0.54778732,  -0.62410512],
       [-0.69148143,  0.19595738],
       [ 0.07530125, -0.66790086],
       ...,
       [-0.18036748,  0.84135547],
       [-0.83018194, -0.3115237 ],
       [-0.64451642, -0.50277112]])
```

A1.11. Plotting the transformed data

Our transformed data can be treated as a list of (x,y) coordinates, and that is how we will plot it. Matplotlib has a scatter method which can take a list of coordinates and display a scatter plot. We set up the various labels and axes properly, and plot the graph

In []:

```
from matplotlib import pyplot as plt
import matplotlib

scale = 2
matplotlib.rc('xtick', labelsz=30/scale)
```

```

matplotlib.rc('ytick', labelsize=30/scale)
plt.figure(figsize=(10,10))
ac = plt.scatter(*zip(*X_pca_inactives), marker = ".", s=1)
ina = plt.scatter(*zip(*X_pca_actives), marker = ".", s=1)
plt.xlabel("PC1", fontsize = 40/scale)
plt.ylabel("PC2", fontsize = 40/scale)
plt.legend((ac, ina),
           ('Inactives', 'Actives'),
           scatterpoints=1,
           loc='upper right',
           ncol=1,
           fontsize=35/scale,
           markerscale=30/scale
           )
plt

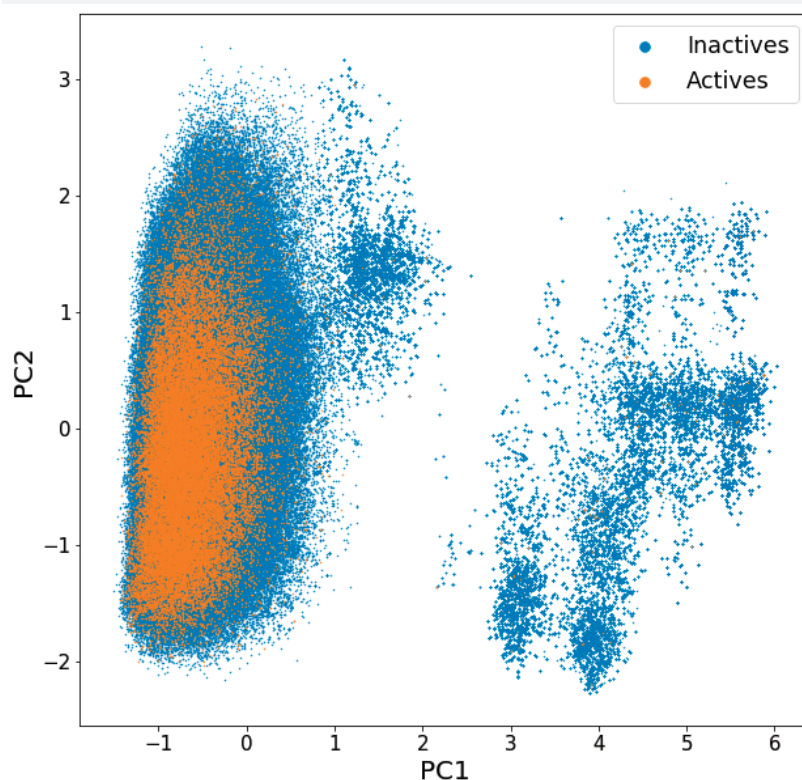
```

Out[12]:

```

<module 'matplotlib.pyplot' from 'C:\\Users\\dipan\\.conda\\envs\\keras\\lib\\
site-packages\\matplotlib\\pyplot.py'>

```



Visualizing Molecules from the clusters

A1.12. Filtering CIDs based on the transformed fingerprint

From our PCA plot, we can see several clusters, and let us find out what type of molecules are there in these clusters. To do this, we must first find out the CIDs of the molecules that are found in the cluster we want to probe. We can do this by querying the fingerprint dictionary we made, transforming the fingerprint, and checking if it belongs in the specified range. The transformation is fairly fast, but a progressbar helps judge progress.

```
In []:
cluster = []
for cid, fp in tqdm(inactivesfp.items()):
    fp = fp.reshape(1, -1)
    transformed = pca.transform(fp)[0]
    if (transformed[0] > 2.9 and transformed[0] < 3.1 ):
        cluster.append(cid)
```

100% | 462890/462890 [00:24<00:00, 19021.06it/s]

A1.13. Downloading smiles

With the CIDs in hand, we now think about getting the structures of these compounds from PubChem. Just for visualization, we do not need all the compounds, just a random subsample will do. We then use the requests module to query the PubChem PUG API and get the SMILES. The PUG requests can timeout if we query too many molecules at once, so to avoid that, we split up the CID list into chunks of 200, and concatenate the results.

In this example, we already have the structure information of the molecules from the csv file we imported in the beginning. However, here we document the process for downloading smiles directly from PubChem.

```
In []:
def chunks(l, n):
    # For item i in a range that is a length of l,
    for i in range(0, len(l), n):
        # Create an index range for l of n items:
        yield l[i:i+n]

def getSmilesFromPubchem(CIDList):
    import requests
    import pandas as pd
```

```

import time
from tqdm import tqdm_notebook as tqdm
cid_chunks = list(chunks(CIDList, 200))
dflist = []
for chunk in tqdm(cid_chunks):
    cids = ",".join(chunk)
    response = requests.get ("https://pubchem.ncbi.nlm.nih.gov/rest/pug/
compound/cid/"+cids+"\
                                property/CanonicalSMILES/CSV")
    smiles_strings = response.content.decode("utf-8").split("\n")
    df = pd.DataFrame([sub.replace(`"`,`").split(",") for sub in smiles_
strings[1:len(smiles_\
                                strings)-1]], columns=["CID", "SMILES"])
    dflist.append(df)
    time.sleep(2)
return pd.concat(dflist)

import random
rand_cidlist = random.choices(cluster, k=25)
smilesdf = getSmilesFromPubchem(list(map(str, rand_cidlist)))
smilesdf

```

	CID	SMILES
0	54629731	<chem>CCCC#CC1=CC2=C(C=C1)S(=O)(=O)N(CC(C(O2)CN(C)C(=O)CC)C(C)CO</chem>
1	54619912	<chem>CC1CN(S(=O)(=O)C2=C(C=C(C=C2)C3=CCCC3)OC1CN(C)C(=O)COC)C(C)CO</chem>
2	54629449	<chem>CC1CN(S(=O)(=O)C2=C(C=C(C=C2)C#CCN(C)C)OC1CN(C)C(=O)CN(C)C)C(C)CO</chem>
3	54618805	<chem>CC1CN(S(=O)(=O)C2=C(C=C(C=C2)C#CC(C)C)OC1CN(C)C(=O)C3=CC=NC=C3)C(C)CO</chem>
4	54627754	<chem>CCC(=O)N(C)CC1C(CN(S(=O)(=O)C2=C(O1)C=C(C=C2)C#C-C3=CN=CC=C3)C(C)CO)C</chem>

A1.14. Drawing Molecules

Now that we have structure information of the molecules encoded as smiles, we can create molecule objects using RDKit. RDKit offers various functionalities for the molecules to be drawn in 2D and 3D. RDKit provides a very handy way of rendering the molecules directly in notebook.

In []:

```

from rdkit import Chem
from rdkit.Chem import rdDepictor

```

```

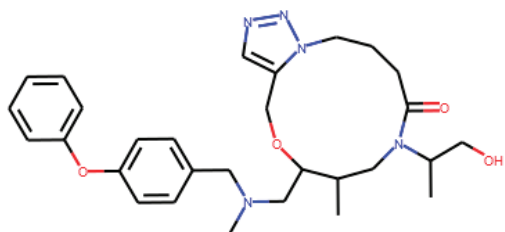
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
from rdkit.Chem.Draw.MolDrawing import MolDrawing
from rdkit.Chem.Draw import rdMolDraw2D
from IPython.display import SVG

IPythonConsole.molSize = (400, 300)

x = "CC1CN(C(=O)CCCN2C(=CN=N2)COC1CN(C)CC3=CC=C(C=C3)OC4=CC=CC=C4)C(C)CO"
m = Chem.MolFromSmiles(x)
m

```

Out[15]:



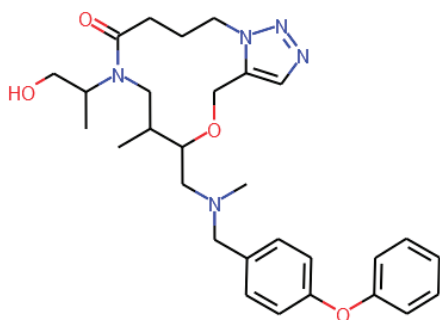
However, the default coordination generation does not work very well with macrocycles. We can use the `rdDepictor` module to create the 2D coordinates. Setting `rdDepictor.SetPreferCoordGen` to `True` allows us to calculate the 2D coordinates of molecules using the new `CoordGen` package in `RDKit`, which yields better results.

```

In [ ]:
rdDepictor.SetPreferCoordGen(True)
rdDepictor.Compute2DCoords(m)
m

```

Out[16]:



Now, we can apply the same operation to all the molecules in the dataframe we created.

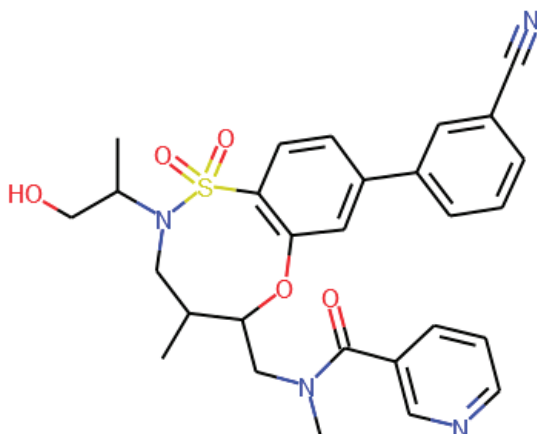
```
In [ ]:
rdDepictor.SetPreferCoordGen(True)
dt = smilesdf
mols = [Chem.MolFromSmiles(x) for x in dt['SMILES']]
list(map(rdDepictor.Compute2DCoords, mols)); #Semicolon to Suppress Output
```

8.14.1. Drawing Molecule in Notebook

As displayed above, we can render a molecule directly in notebook using IPythonConsole from the Draw module. This is very good for displaying individual molecules and inspecting their structures.

```
In [ ]:
mols[10]
```

Out[18]:

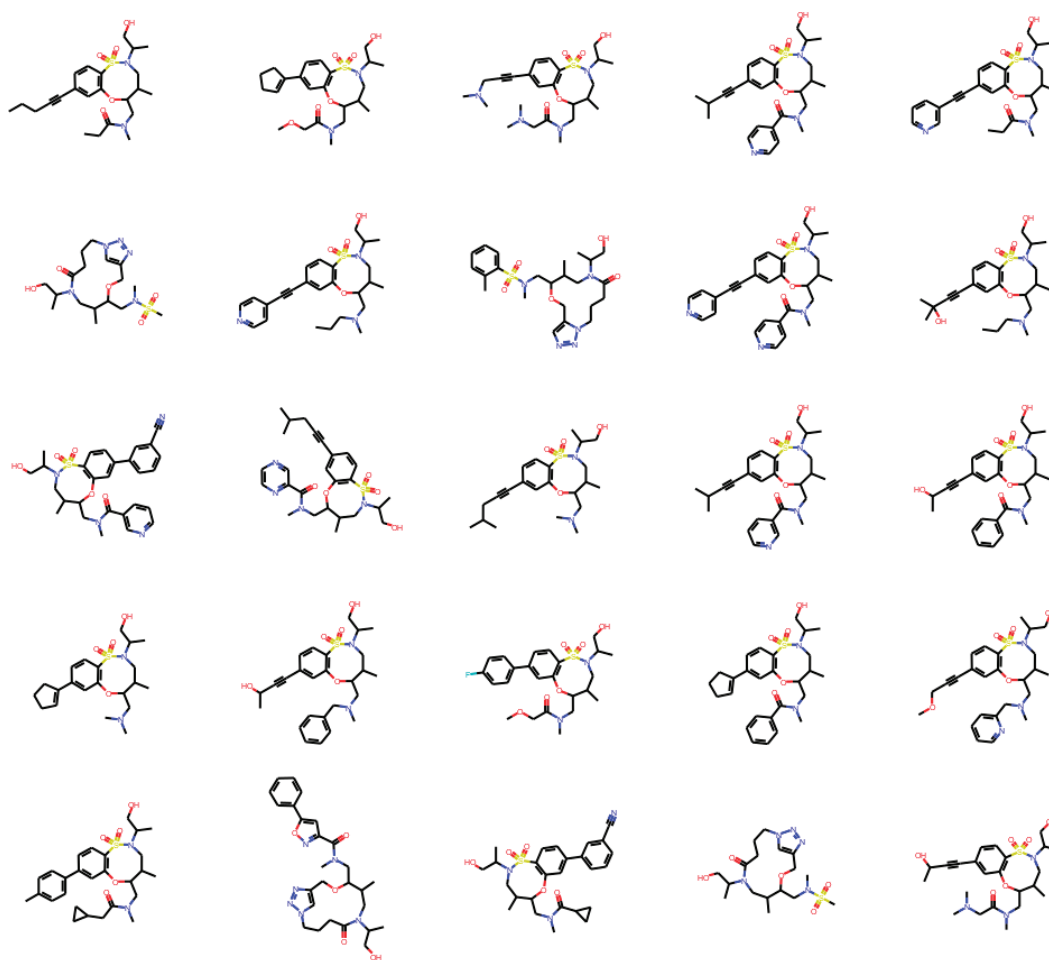


8.14.2. Drawing molecules as grid

What we would like here, is an imagegrid for all the molecules. RDKit conveniently provides a method for doing just this. The MolsToGridImage method returns a PIL image object. We can also use the useSVG parameter to export this as vector graphics.

```
In [ ]:
scale = 1
MolDrawing.atomLabelMinFontSize = 250
MolDrawing.atomLabelFontSize=45
img=Draw.MolsToGridImage(mols[:100],molsPerRow=5,subImgSize=(200*scale,175*s-
cale), useSVG=False, maxMols=100)
img
```


Out[19]:



Appendix II

Table S1: All relevant assays from PUBCHEM found as a part of our search for a test set.

Assay ID	Source	Target	Number tested	Number Active	Date	Donor	Acceptor
504332	NCGC	euchromatic histone-lysine N-methyltransferase 2 [Homo sapiens]	353740	31109	2011-02-22	Streptavidin	Anti-Rabbit IgG
743279	NCGC	interleukin-1 beta pro-protein [Homo sapiens]	362359	17206	2014-02-11	Streptavidin	AlphaLisasa-SA
504339	NCGC	Chain A, Jmjd2a Tandem Tudor Domains In Complex With A Trimethylated Histone H4-K20 Peptide	388413	17028	2011-02-22	Streptavidin	Nickel chelate
504333	NCGC	bromodomain adjacent to zinc finger domain 2B [Homo sapiens]	359824	15779	2011-02-22	Streptavidin	Nickel chelate
623870	Broad Institute	aryl hydrocarbon receptor nuclear translocator [Homo sapiens]; ... Total	392905	2595	2012-03-27	GST coated	Ni chelate AlphaLISA
540317	NCGC	chromobox protein homolog 1 [Homo sapiens]	387034	2150	2011-07-28	Streptavidin	Nickel chelate
651725	NCGC	six1 [Homo sapiens]	364407	2028	2012-10-29	GST coated	Nickel chelate
743445	Broad Institute	histone-lysine N-methyltransferase NSD2 isoform 1 [Homo sapiens]	309832	1662	2014-04-10	Streptavidin	Biotin
651724	NCGC	CtBP interacting protein CtIP [Homo sapiens]	335532	1654	2012-10-29	GST coated	Nickel chelate
651704	Broad Institute	protein AF-9 isoform a [Homo sapiens]	348218	1633	2012-10-26	Streptavidin	Biotin
485360	NCGC	lethal(3)malignant brain tumor-like protein 1 isoform I [Homo sapiens]	225505	1495	2010-10-05	Streptavidin	Nickel chelate
504329	Southern Research Specialized Biocontainment Screening Center	nonstructural protein 1 [Influenza A virus (A/California/07/2009(H1N1))]	335445	1013	2011-02-22	Unknown (Perkin-elmer)	Unknown (Perkin-elmer)
485290	NCGC	Chain A, Crystal Structure Of Human Tyrosyl-Dna Phosphodiesterase (Tdp1)	352260	986	2010-09-28	Streptavidin	anti-FITC
624168	Burnham Center for Chemical Genomics	LARGE [Homo sapiens]	364168	805	2012-05-22	Streptavidin	Anti-Mouse IgM AlphaLISA

651723	Broad Institute	target unknown	349095	741	2012-10-28	Antibody	Antibody
--------	-----------------	----------------	--------	-----	------------	----------	----------

Table S2: Selection criteria of FH from PubChem assays. The activity fraction indicates the ratio of the number of times active / number of times tested.

Activity Fraction	Observed	95% upper confidence	Enrichment
1/6	87	907	0.1
1/7	652	2897	0.2
1/8	2008	6765	0.3
1/9	6666	16468	0.4
2/7	104	163	0.6
1/10	18635	27955	0.7
1/11	10254	12679	0.8
2/9	1957	1591	1.2
1/5	5365	4291	1.3
0	297343	277634	1.1
2/11	2529	1548	1.6
1/4	2513	606	4.2
1/3	1360	137	9.9
3/8	210	19	11
3/10	2125	170	12
3/11	1217	95	13
2/5	867	33	26
5/8	54	0	Inf ^f
3/4	371	0	Inf
1/2	920	0	Inf
5/9	130	0	Inf
7/9	23	0	Inf
3/5	133	0	Inf
2/3	212	0	Inf
5/11	194	0	Inf
6/11	95	0	Inf
4/9	306	0	Inf
4/11	492	0	Inf
9/10	9	0	Inf
1	52	0	Inf
9/11	7	0	Inf
5/7	14	0	Inf
3/7	52	0	Inf
4/7	27	0	Inf
4/5	25	0	Inf
8/11	17	0	Inf
8/9	6	0	Inf

Appendix II

7/11	32	0	Inf
7/10	35	0	Inf
10/11	2	0	Inf
7/8	6	0	Inf

¹Inf - the enrichment was > 1000.