Book Title:

"Computational Drug Discovery and Design"
in *Methods in Molecular Biology*, Springer Protocols

Chapter Title:

Automated Inference of Chemical Discriminants of Biological Activity

Authors:

Sebastian Raschka[1], Anne M. Scott[2], Mar Huertas[2,4], Weiming Li[2], and Leslie A. Kuhn[1,2,3,5]

Affiliations:

[1]Department of Biochemistry and Molecular Biology, Michigan State University, East Lansing, MI 48824, USA

[2]Department of Fisheries and Wildlife, Michigan State University, East Lansing, MI 48824, USA

[3]Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA

[4]Current address: Department of Biology, Texas State University, San Marcos, TX 78666, USA

[5]Corresponding author; KuhnL@msu.edu

Summary/Abstract

Ligand-based virtual screening has become a standard technique for the efficient discovery of bioactive small molecules. Following assays to determine the activity of compounds selected by virtual screening, or other approaches in which dozens to thousands of molecules have been tested, machine learning techniques make it straightforward to discover the patterns of chemical groups that correlate with the desired biological activity. Defining the chemical features that generate activity can be used to guide the selection of molecules for subsequent rounds of screening and assaying, as well as help design new, more active molecules for organic synthesis.

The quantitative structure-activity relationship machine learning protocols we describe here, using decision trees, random forests, and sequential feature selection, take as input the chemical structure of a single, known active small molecule (for example, an inhibitor, agonist, or substrate) for comparison with the structure of each tested molecule. Knowledge of the atomic structure of the protein target and its interactions with the active compound are not required. These protocols can be modified and applied to any data set that consists of a series of measured structural, chemical, or other features for each tested molecule, along with the experimentally measured value of the response variable you would like to predict or optimize for your project, for instance, inhibitory activity in a biological assay or $\Delta G_{binding}$. To illustrate the use of different machine learning algorithms, we step through the analysis of a dataset of inhibitor candidates from virtual screening that were tested recently for their ability to inhibit GPCR-mediated signaling in a vertebrate.

Key Words:

Virtual screening, ligand-based screening, quantitative structure-activity relationship, pharmacophore, fingerprint analysis, machine learning, random forest, GPCR, invasive species control

Abbreviations:

2D: two-dimensional; 3D: three-dimensional; 3kPZS: 7α,12α, 24-trihydroxy-5α-cholan-3-one 24-sulfate; CAS: Chemical Abstracts Service Registry, CSD: Cambridge Structural Database, DKPES: 3,12-diketo-4,6-petromyzonene-24-sulfate, EOG: electro-olfactogram; GPCR: G-protein coupled receptor; QSAR: quantitative structure-activity relationship; SFS: sequential feature selection; SBS: sequential backward selection; VS: virtual screening, ZINC12: Zinc is Not Commercial Database, version 12

1. Introduction

In this chapter, we will apply machine learning to analyze the results from a virtual screening (VS) project for discovering inhibitors of GPCR signaling in a vertebrate, to infer the importance of functional groups for their biological activity. Computer-based ligand screening, also known as ligand-based screening, is frequently used in pharmaceutical discovery because it performs robustly in identifying active molecules from the top-scoring set and does not require the availability of an atomic structure of the protein target (1, 2). Further, it has been shown that ligand-based virtual screening is capable of exploring different active scaffolds, making it a valuable alternative to structure-based methods such as molecular docking, even when atomic structures of the target are known (3, 4).

However, scientists typically focus on the most active handful of compounds and test their closest analogs while not making use of the activity data available from all the tested compounds to identify correlations

between their chemical groups and activity values. Part of this may be due to the need to establish spatial correspondences between chemical groups in compounds containing different molecular scaffolds (for example, comparing substituents on a steroid ring system versus a purine nucleotide). This problem has been circumvented in the protocols presented here by considering all molecules as fully flexible 3D structures and determining their optimal overlay based on the volumes and partial charges of the atoms, followed by comparing the chemical identities of neighboring atoms and small organic groups such as –NH2. We will use the term "functional groups" to refer to single or small groups of atoms that are being compared between molecules. This flexible overlay procedure provides a rational and quantitative way of comparing chemical groups between compounds.

Discovering Biologically Active Molecules through Virtual Screening

The most prominent approaches in the computer-aided discovery of biologically active molecules are *structure-based* screening (5–8) and *ligand-based* screening (1, 2, 9, 10) as well as hybrids thereof (11, 12). Traditionally, structure-based screening is restricted to applications where an experimentally determined, high-resolution three-dimensional (3D) structure of the ligand's binding partner (usually a protein or nucleic acid) is available from X-ray crystallography or nuclear magnetic resonance experiments.

While ligand-based screening does not require knowledge of the binding target, it assumes that active molecules are likely to share shape and chemical similarities with a known, biologically active ligand. In short, ligand-based screening can be described as a similarity search between a known ligand and the molecules in a database (Figure 1).

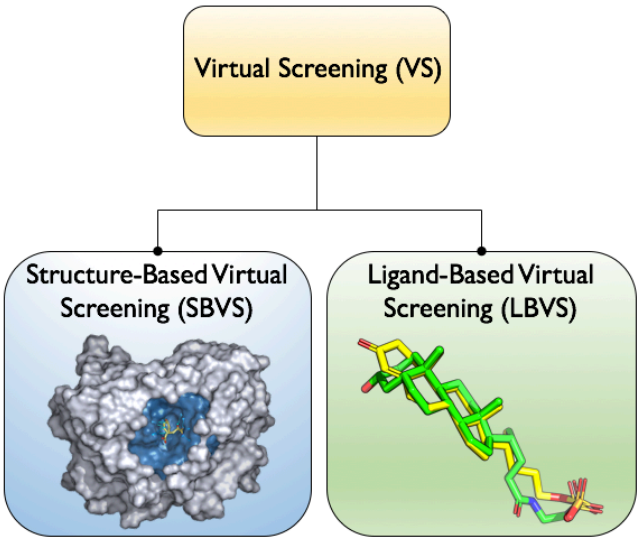

Figure 1. An illustration of the two broad categories of virtual screening: Structure-based virtual screening involves docking into a binding site to maximize protein-ligand surface complementarity, and ligand-based virtual screening involves evaluating small-molecule similarity with a known ligand.

1.1. Using Machine Learning to Identify Functional Groups Associated with Biological Activity

To guide virtual screening, understand biological mechanisms, and aid the design of more potent inhibitors or activators of molecular processes, several different techniques have been developed to analyze datasets of molecular descriptors and measured activity. A common goal in quantitative structure-activity relationship (QSAR) modeling includes the prediction of the *in vitro* or *in vivo* activity of molecules given their features. Another common goal is to gain insights into the importance of individual functional groups for binding or chemical activity; such insights are invaluable for the discovery and optimization of potent agonists or inhibitors. More detailed discussions of QSAR can be found in Kubinyi *et al.* 2006 (13) and Verma *et al.* 2010 (14).

To infer which functional groups are most important for biological activity, this chapter focuses on the use of supervised machine learning algorithms to discover functional group matching patterns that explain the relative activity of the tested inhibitor candidates. Primarily, the analysis of the discriminants of biological activity presented here employs tree-based machine learning algorithms. A decision tree (15) that separates active from non-active molecules provides a model that is readily interpretable, resulting in a set of decision rules that if chained together, can explain the hierarchy of features in a molecule that are most important for distinguishing actives from non-actives. Secondly, multiple decision trees will be combined via the random forest method (16). Each decision tree in a random forest is fit to a random sample of the training data and feature set. This produces an ensemble of different decision trees, which together provide a robust predictive model that is less prone to overfitting the training data than any individual decision tree (16). Furthermore, a random forest facilitates the computation of feature importance as the average information gain over the individual trees, as it will be explained in more detail in the Methods section. Lastly, we will utilize an implementation of sequential backward selection, a sequential feature selection algorithm that identifies subsets of features to maximize the performance of a given model in a greedy (fastest improvement, rather than exhaustive) fashion (17, 18). Sequential feature selection algorithms can be combined with any machine learning algorithm, and hence, they provide a flexible, model-agnostic solution for the analysis of combinations of functional groups that explain biological activity.

1.2. Predicting the Essential Features of GPCR Inhibitors: A Real-World Case Study

This chapter presents an automated, machine learning-based approach to infer the discriminants of activity in molecules from assays performed on compounds prioritized by ligand-based screening. To explain the methodology behind this approach, we will consider a novel dataset of 56 molecules that have been prioritized as candidates for inhibiting GPCR-mediated pheromone signaling in an invasive species control project. Readers can access the same data and software and then perform the same analyses and compare their results with ours.

The goal of this invasive species control project is to inhibit a pheromone-induced GPCR olfactory signaling pathway. We hypothesized that the inhibition of pheromone detection by the olfactory system will prevent mature female sea lamprey from reaching mature males at spawning grounds in tributaries of the Laurentian Great Lakes, and thus reduce the invasive sea lamprey population. Controlling the sea lamprey with pesticide applications currently costs millions of dollars per year, with native fish populations and commercial fishing continuing to be impacted by sea lamprey parasitism (19). The rationale behind the screening side of this project is based on a recently completed project (10), focusing

on inhibiting the GPCR signaling pathway induced by a male sea lamprey mating pheromone, 7α,12α, 24-trihydroxy-5α-cholan-3-one 24-sulfate (3kPZS).

The dataset analyzed here consists of the chemical structures and assay data for another male sea lamprey mating pheromone, the sulfate-conjugated bile alcohol, 3,12-diketo-4,6-petromyzonene-24-sulfate (DKPES, Figure 3). The 56 molecules prioritized by ligand-based screening according to their degree of 3D DKPES similarity were assayed for their ability to block the *in vivo* sea lamprey olfactory response to DKPES, as measured by an electro-olfactogram assay (EOG). The activity data was then analyzed using machine learning algorithms to uncover structure-function patterns.

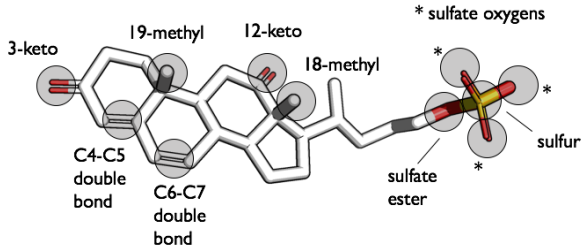

Figure 2. 3D structure of a favorable (low-energy) DKPES conformer, where the functional group features corresponding to the columns in the olfactory response dataset are highlighted with gray circles. [insert high res] White indicates carbon, red indicates oxygen, and yellow indicates sulfur.

A brief summary of the virtual screening approach that we used to identify inhibitory mimics of DKPES is provided in Figure 3.

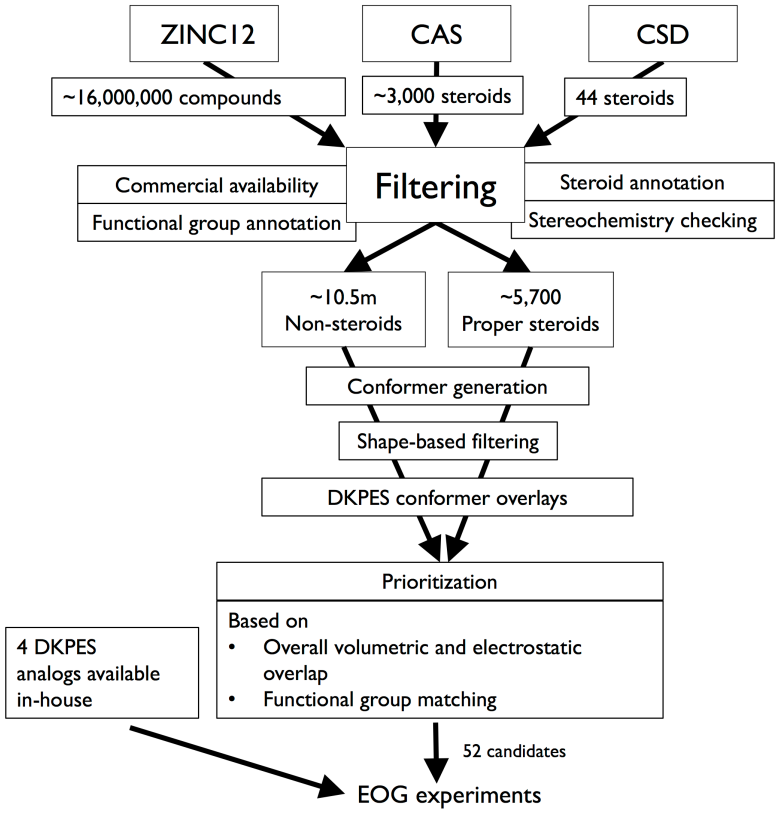

Figure 3. Summary of the virtual screening workflow to prioritize molecules for electro-olfactogram (EOG) assays. The Screenlamp toolkit (10) (https://github.com/psa-lab/screenlamp) was used to prepare the virtual screening pipeline, including ROCS v3.2.0.4 (OpenEye Scientific Software, Santa Fe, NM; https://www.eyesopen.com/rocs), OMEGA v2.4.6 (OpenEye Scientific Software, Santa Fe, NM; https://www.eyesopen.com/omega), and QUACPAC/molcharge (OpenEye Scientific Software, Santa Fe, NM; https://docs.eyesopen.com/toolkits/python/quacpactk/molchargetheory.html). The screening databases of small molecules, mostly commercially available, were the drug-like molecules in ZINC12 (http://zinc.docking.org) (20), steroid structures from Chemical Abstracts Service Registry (CAS; https://www.cas.org), and steroid structures from the Cambridge Structural Database (CSD; https://www.ccdc.cam.ac.uk/solutions/csd-system/components/csd/) (21).

As a result, 56 candidate molecules were prioritized for biological assays based on the following criteria:

- Steroidal substructure containing molecules with a high degree of shape and charge match to the reference pheromone, DKPES (Figure 2).
- Four DKPES analogs with oxidized (double bond-containing) rings that are naturally produced by mature male sea lamprey (molecule IDs: ENE 1-4, Figure 4) (22).
- Diverse compounds to test the hypothesis that compounds with the best charge and shape match will mimic DKPES (without requiring a steroid core or sulfate tail match).
- Non-steroid compounds having 3-keto or 3-hydroxy and 12-keto or 12-hydroxy matches and at least one sulfate oxygen match that overlay on the corresponding oxygen atoms in DKPES (Figure 2).

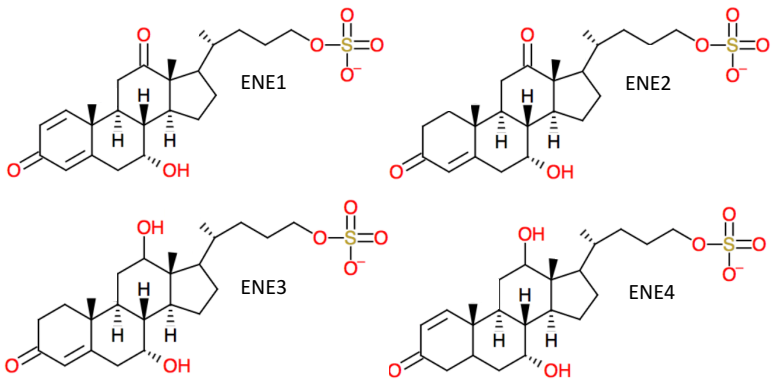

Figure 4. 2D structures of the 4 combinatorial DKPES analogs ("ENE" compounds) (22)
ENE1: 7,24-dihydroxy-3,12-diketo-1,4-choladiene-24-sulfate;
ENE2: 7,24-dihydroxy-3,12-diketo-4-cholene-24-sulfate;
ENE3: 7,12,24-trihydroxy-3-keto-4-cholene-24-sulfate;
ENE4: 7,12,24-trihydroxy-3-keto-1-cholene-24-sulfate;

To measure the biological activity of the 56 DKPES inhibitor candidates selected with the above screening and prioritization criteria, we used an electro-olfactogram (EOG) (10). The measured EOG response, acting as the target variable in this dataset, was the percentage reduction of the standard DKPES signal when the sea lamprey nose was perfused with a known concentration (10^{-6} M) of inhibitor candidate (computed as the average of 2 to 5 experimental replicates). Figure 5 shows four of the 56 molecules for illustrative purposes, two actives and two non-actives, with the percent DKPES olfactory inhibition for each. In the context of this project, *non-actives* were defined as molecules that block the olfactory response by less than 40% in EOG assays, and molecules that block the signaling response by at least 60% were defined as *actives*.

The DKPES dataset for analysis by machine learning contains the ROCS overlay scores from ligand-based screening (Figure 3) as well as the functional group matching information provided by Screenlamp in tabular form (10) (https://github.com/psa-lab/predicting-activity-by-machine-learning).

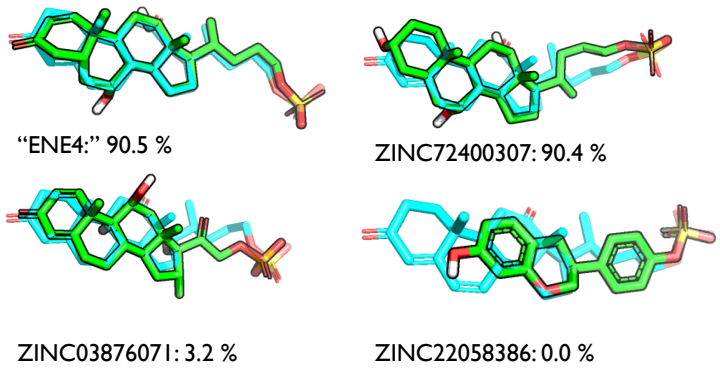

Figure 5. 3D structures and percent DKPES olfactory inhibition of the two most active molecules (actives, top row) and two low-activity molecules (non-actives, bottom row) from the screening set, shown in green as overlayed with the best-matching DKPES 3D conformer (cyan).

Using the DKPES dataset as a case study, the Methods section will explain how to work with such tabular datasets consisting of samples and molecular features using open source libraries for data parsing, visualization, and machine learning. The code and data used in the following section is freely available at https://github.com/psa-lab/predicting-activity-by-machine-learning.

2. Materials

2.1. Python Interpreter

To perform the analyses described in the Methods section, a recent Python (23) version (3.5 or newer) is required (Python 3.6 is recommended). A Python installer for all major operating systems (macOS, Windows, and Linux) can be downloaded from https://www.python.org/downloads/.

2.2. Python Libraries for Scientific Computing

The following list specifies the Python libraries used in this chapter, the recommended version number, and a short description of their use:

- NumPy version 1.13.0 or newer (http://www.numpy.org); numerical array library (24)
- SciPy version 0.19.0 or newer (https://www.scipy.org); advanced functions for scientific computing (25)
- Pandas version 0.20.1 or newer (http://pandas.pydata.org); handling of CSV files and working with data frames (26)
- Matplotlib version 2.0.2 or newer (https://matplotlib.org); 2D plotting (27)
- Scikit-learn version 0.18.1 or newer (http://scikit-learn.org/stable/); algorithms for machine learning (28)
- MLxtend version 0.7.0 or newer (http://rasbt.github.io/mlxtend/); sequential feature selection algorithms (18)

The scientific computing libraries listed above can be installed using Python's in-built Pip module (https://pypi.python.org/pypi/pip) by executing the following line of code directly from a macOS/Unix, Linux, or Windows MS-DOS terminal command line:

```
pip install numpy scipy pandas matplotlib scikit-learn pydotplus mlxtend
```

If you encounter problems with version incompatibilities, you can specify the package versions explicitly, as shown in the following terminal command example:

```
pip install numpy==1.13.0 scipy==0.19.0 pandas==0.20.1 matplotlib==2.0.2 scikit-
learn==0.18.1 pydotplus==2.0.2 mlxtend==0.7.0
```

2.3. Graph Visualization Software

To visualize the decision trees later in this chapter, an installation of GraphViz is needed. The GraphViz downloader is freely available at http://www.graphviz.org with the installation and setup instructions.

2.4. Dataset

The datasets being used in this chapter, as well as the source files of all the accompanying code, are available online under a permissive open source license at https://github.com/psa-lab/predicting-activity-by-machine-learning.

2.5 Additional Resources

If you are unfamiliar with Python and the Python libraries that you installed in section *2.2. Python Libraries for Scientific Computing*, it is highly recommended to familiarize yourself with their basic functionality by reading these freely available resources:

- *Python Beginner Guide*: https://wiki.python.org/moin/BeginnersGuide
- *NumPy Quickstart Tutorial*: https://docs.scipy.org/doc/numpy-dev/user/quickstart.html
- *Introduction to NumPy*: https://sebastianraschka.com/pdf/books/dlb/appendix_f_numpy-intro.pdf
- *10 Minutes to pandas*: http://pandas.pydata.org/pandas-docs/stable/10min.html
- *Matplotlib Tutorials*: https://matplotlib.org/users/index.html
- *An introduction to machine learning using scikit-learn*: http://scikit-learn.org/stable/tutorial/basic/tutorial.html

3. Methods

This section walks through the individual steps involved in a typical analysis pipeline for identifying which functional groups and atoms (or other molecular properties or *features*) are predictive of the measured biological activity of the molecules. The first section explains how the tabular DKPES dataset can be loaded into a Python session for analysis.

3.1 Loading and Inspecting the Biological Activity Dataset

This section explains how to load a CSV-formatted dataset table into a current Python session. A convenient way to parse a dataset from a tabular plaintext format, such as CSV, is to use the `read_csv` function from the Pandas library as shown in the code example in Figure 6, which loads the DKPES dataset into a Pandas `DataFrame` object (`df`) for further processing. (See Note 1 in section 4.1 for information on reading datasets from different file formats.)

```
>>> import pandas as pd
>>> df = pd.read_csv('../data/csvs/dkpes.csv')
>>> df.head(10)
```

	index	Signal-inhibition	3-Keto	3-Hydroxy	12-Keto	12-Hydroxy	19-Methyl	18-Methyl	Sulfate-Ester	Sulfate-Oxygens	...
0	ENE4	0.905	1	0	0	1	1	1	1	3	...
1	ZINC72400307	0.904	0	0	0	1	1	1	1	3	...
2	ENE3	0.897	1	0	0	1	1	1	1	3	...
3	ENE1	0.893	1	0	1	0	1	1	1	3	...
4	ENE2	0.845	1	0	1	0	1	1	1	3	...
5	ZINC12494532	0.741	0	0	0	0	0	0	1	3	...
6	ZINC35044325	0.739	0	1	0	0	1	1	0	3	...
7	ZINC04095893	0.722	0	0	0	0	1	1	0	3	...
8	ZINC01532179	0.686	0	0	0	0	0	0	1	3	...
9	ZINC70666191	0.627	0	0	0	1	1	1	0	0	...

Figure 6. Code for reading in the DKPES dataset into a data frame. The characters >>> denote a Python interpreter prompting for a command to enter and execute. The table resulting from the execution of this code example (`df.head(10)`) shows an excerpt from the DKPES data table sorted by signal inhibition: the 10 most active molecules from the EOG experiments and their functional group matching patterns.

As a result from executing the code shown in Figure 6, the `df.head(10)` call will display the first ten rows in the dataset, to confirm that the data file has been parsed correctly. The DKPES dataset consists of 56 rows, where each row stores the functional group matching information for an assayed molecule with the reference molecule DKPES. (See Note 2 in section 4.2 for information on sorting data frames by specific columns.)

Please note that this work assumes that a tabular dataset containing information of the molecules as well as the assay response have already been collected in tabular form. However, the analysis approach outlined in this chapter is a general one, and it is not restricted to the specific functional group matching patterns shown in Figure 6. For more information on how this functional group matching data can be generated from a ligand-based screening, see (10) (https://github.com/psa-lab/screenlamp).

The first column of the DKPES data table (Figure 6), "index," numbers each molecule. The "Signal Inhibition" column contains the response variable measured by the biological assay, in this case ranging from 0 (non-active) to 1 (highly active, with 100% DKPES signal inhibition). For instance, we can see from the table (Figure 6) that ENE4 and ZINC72400307 (petromyzonal sulfate) were the most promising candidate inhibitors, as they reduced the olfactory response to DKPES by 90.5% and 90.4%, respectively, when each inhibitor candidate was used at the same equimolar concentration (10^{-6} M) as DKPES. The consequent columns, labeled as 3-Keto, 3-Hydroxy, and so forth, contain information about whether an atom or functional group in the candidate molecule overlayed (within 1.3 Å) with the same group in DKPES. This functional group matching data is stored as a binary variable, where 0 indicates "no overlay" and 1 indicates "overlay." In addition, the ROCS shape and charge (“color”) overlay scores were appended to the dataset. For information on how the overlay scores are computed, the reader is referred to https://docs.eyesopen.com/rocs/shape_theory.html and Hawkins *et al.* 2007 (4); however, it shall be noted that the ROCS scoring data is not essential for this analysis. (See Notes 3-5 in section 4.1 for information on working with structural information and notes about obtaining and computing molecular features.)

It is always helpful to perform exploratory analyses when working with a new dataset. The following code snippet shown in Figure 7 will generate the histogram of the signal inhibition values shown, plus the 2D scatter plot comparing the signal inhibition values with the molecular similarity measured in the overlays. First, the signal inhibition data from the data frame (`df`) is assigned to a variable `y`, and the functional group columns of interest to a variable `X`. Next, the code in Figure 7 demonstrates how to use matplotlib to create two sublots, `ax[0]` and `ax[1]`, showing a histogram of the signal inhibition and a scatter plot of the signal inhibition versus molecular similarity side by side. (If you are new or unfamiliar with the matplotlib syntax, it is recommended to consult the tutorials and resources listed in section *2.5 Additional Resources*.)

```
>>> y = df['Signal-inhibition'].values
>>> fgroup_cols = ['3-Keto', '3-Hydroxy', '12-Keto',
...                '12-Hydroxy', '19-Methyl', '18-Methyl', 'Sulfate-Ester',
...                'Sulfate-Oxygens', 'C4-C5-DB', 'C6-C7-DB', 'Sulfur']
>>> X = df[fgroup_cols].values
>>> fig, ax = plt.subplots(1, 2, figsize=(10, 3))
>>> ax[0].hist(y, bins=np.arange(0, 1.1, 0.1))
>>> ax[0].set_ylabel('Molecule count')
>>> ax[0].set_xlabel('Signal inhibition')
>>> ax[1].scatter(y, df['TanimotoCombo'].values)
>>> ax[1].set_ylabel('Tanimoto similarity')
>>> ax[1].set_xlabel('Signal inhibition')
>>> plt.show()
```

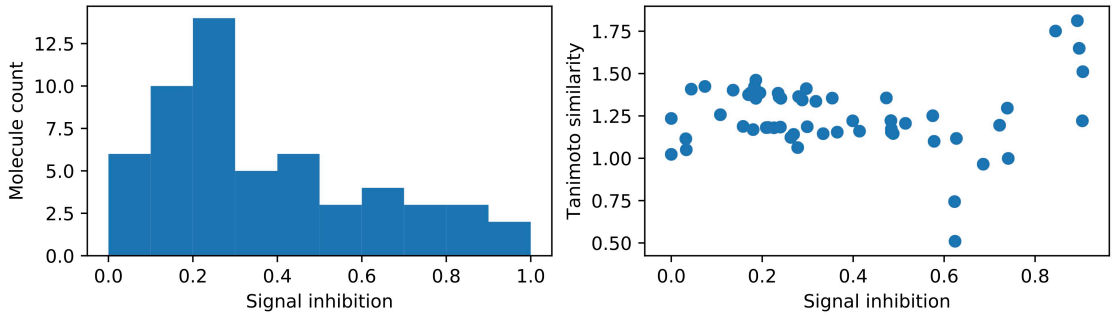

Figure 7. Code for performing exploratory analysis in Python using the matplotlib library to plot a histogram of the "Signal Inhibition" data and a scatter plot to inspect the relationship between the signal inhibition and overlay scores. In the corresponding programming code, the "Signal Inhibition" column is first assigned to a variable `y`, and the functional groups of interest are assigned to the variable `fgroup_cols`, which is then used to create the matrix `X` that stores the functional groups matching patterns of those functional groups of interest. Next, a figure with two subplots is initialized by calling `plt.subplots` from matplotlib. The `plt.hist` function adds the histogram to the first subplot (`ax[0]`), and the `plt.scatter` function draws the scatter plot in the subplot to the right (`ax[1]`). The resulting plots show the DKPES inhibitor activity distribution for the 56 compounds that were assayed (left) and the relationship between activity and overlay similarity from ROCS (right), given as the TanimotoCombo score in the range 0 to 2, where 2 means that two 3D structures have an identical volume and partial charge distribution.

From the histogram (left panel of Figure 7), we can see that most molecules inhibit the DKPES signal by less than 50% in *in vivo* EOG assays. The scatter plot in Figure 7 shows that 4 out of the 5 most active molecules have a high overlay similarity value of 1.5 or greater. The *TanimotoCombo* value is the sum of the volumetric and chemical similarity components, where an exact match (two identical molecules in the same conformation) will result in a maximum score of 1 for each, summing to a maximum score of 2. While the top 4 most active molecules have the highest overlay similarity, no correlation between overlay similarity and signal inhibition can be observed across the remaining 52 molecules. This indicates that more specific determinants of activity are at play, motivating the pattern analysis of functional groups

matching DKPES. Interestingly, the outlier with a very low Tanimoto similarity score and a moderately high signal inhibition value of 0.62 is a sulfate tail-containing natural product produced by sea squirts (ZINC14591952) (29), shown in Figure 8.

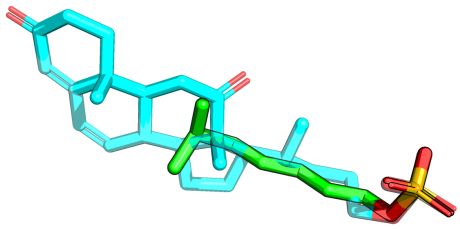

Figure 8. Sulfate tail compound sodium 6-methylheptyl sulfate (carbon atoms shown in green; ZINC14591952, average of 62% signal inhibition in EOG experiments) overlaid with the most similar DKPES conformer (carbon atoms shown in cyan).

From this molecule we can conclude that mimicking the sulfate group in DKPES alone can block the olfactory response of DKPES by approximately 60%, likely by competing with interactions of the similar tail in DKPES with the GPCR ligand binding site.

3.2 Chemical and Functional Groups

This section explains how to visualize the other features in the dataset: the functional group matches with the DKPES reference molecule (Figure 2). Using the code in Figure 9, we will plot the functional group matching pattern of the top 10 most active and 10 least active molecules via two heat maps shown side by side for a visual comparison. (See Note 1 in section 4.2 for information about the distance threshold chosen for assigning positive and negative functional group matches in this dataset.)

```
>>> fig, ax = plt.subplots(1, 2, figsize=(10, 3))
>>> ax[0].imshow(X[:10], cmap='binary')
>>> ax[0].set_title('10 most active molecules')
>>> plt.sca(ax[0])
>>> plt.xticks(range(len(fgroup_cols)),
...            fgroup_cols, rotation='vertical')
>>> plt.yticks(range(10), df['index'][:10])
>>> ax[1].imshow(X[-10:], cmap='binary')
>>> ax[1].set_title('10 least active molecules')
>>> plt.sca(ax[1])
>>> plt.xticks(range(len(fgroup_cols)),
...            fgroup_cols, rotation='vertical')
>>> plt.yticks(range(10), df['index'][-10:])
>>> plt.show()
```

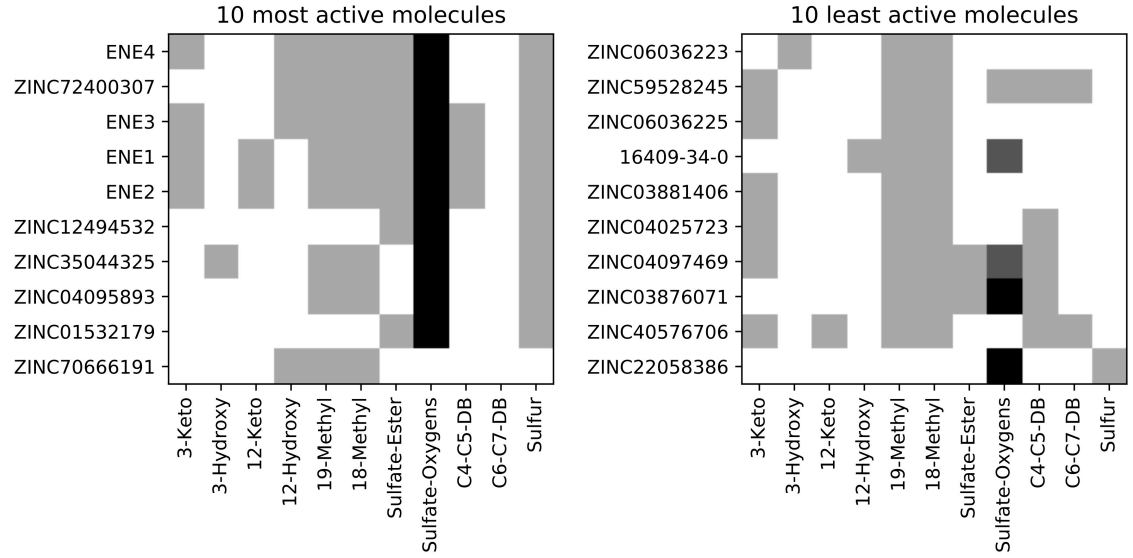

Figure 9. Code to generate heat maps showing matches of functional groups in DKPES by the 10 most active (left) and 10 least active (right) molecules tested in EOG assays. Using the `matplotlib.pyplot` function that was important as `plt` earlier (Figure 7), we create two subplots stored in the array `ax`. Using `matplotlib.pyplot`'s `imshow`, we plot the functional group patterns of the ten most active molecules (`X[:10]`, the first ten elements in the sorted data array) as a heat map in left subplot (`ax[0]`). Similarly, we plot the ten least active molecules (the last ten molecules in the array, `X[-10:]`) as a heat map in the right subplot (`ax[1]`). As the heat maps show, all features except sulfate-oxygens are encoded as binary variables (0: white cell background, no match; 1: light gray, match). Sulfate-oxygens refers to the three terminal oxygens, excluding the sulfate ester oxygen. This variable has values from 0-3 (up to all 3 terminal oxygens being matched), where black cell backgrounds correspond to 3 matches, dark gray corresponds to 2 matches, and light gray to 1 match, respectively.

Looking at the heat maps in Figure 9, the following conclusions can be drawn:

- The top 9 most active molecules have a sulfur match and match three of the oxygen atom in the DKPES sulfate group.
- Sulfur and sulfate oxygen atom matches alone are not sufficient for activity. From the previous scatter plot analysis (Figure 7), we know that the sulfate tail analog alone (Figure 8; ZINC14591952) shows a signal inhibition of 60%. It matches the 3 terminal sulfate-oxygens and sulfur atom. However, a compound with the same matching pattern (ZINC22058386 in Figure 9) has no biological activity in the same assay, likely due to its greater bulk (Figure 5).

However, casual inspection of the data does not always lead to insights that apply to all of the compounds, and it can miss interesting trends, especially for large datasets. The next section will introduce several machine learning approaches for deducing the importance of functional groups for biological activity.

3.3 Tracing Preferential Chemical Group Patterns using Decision Trees

Decision tree classifiers are a good choice if we are concerned about the interpretability of the combinations of features used to predict activity. While decision trees can be trained to predict outcomes on a continuous scale (regression analysis), we fill focus on decision trees for classification in this chapter, that is, predicting whether a molecule is active or non-active. While the discretization of the continuous target variable (here: *signal inhibition in percent*) is to some extent arbitrary, it helps with improving the interpretability of the selected features as they can be directly interpreted as discriminants of active and non-active molecules. For the following analysis, we considered molecules with a signal inhibition of 60% or greater as active molecules.

As you will see, within a tree it is easy to trace the path of decisions comprising the model that best separates different classes of molecules (here: active vs non-active). In other words, based on the functional group matching information in the DKPES dataset, the decision tree model poses a series of questions to infer the discriminative properties between active and non-active molecules. (See Note 1 in section 4.3 for more information about appropriate dataset sizes and tips for dealing with imbalanced datasets).

The learning algorithm that is constructing a nonparametric decision tree model from the dataset works as follows. Starting at the tree root and splits the dataset (the active and non-active molecules) on the feature (for example, *presence of a sulfur match*) that results in the largest information gain. In other words, the objective function of a decision tree is to learn, at each step, the splitting criterion (or decision rule) that maximizes the information gain upon splitting a parent node into two child nodes. The information gain is computed as the difference between the impurity of a parent node and the sum of its child node impurities. Intuitively, we can say the lower the impurity of the child nodes, the larger the information gain. The impurity itself is a measure of how diverse the subset of samples is, in terms of the class label proportion, after splitting. For example, after asking the question "does a molecule have a positive sulfur match?" a pure node would only contain either active or non-active molecules when answering this question with a "yes." A node that consisted of 50% non-active and 50% active samples after applying a splitting criterion would be most impure — such a result would indicate that it was not a useful criterion for distinguishing between active and non-active molecules. In the decision tree implementation that we are going to use in this chapter, the metric for computing the impurity of a given node is measured as *Gini impurity* as used in the CART (classification and regression tree) algorithm (15). Gini impurity is defined as follows:

$$\text{Impurity}(t) \;=\; \sum_{i=1}^{c} p(i|t)(1 - p(i|t)) \;=\; 1 \;-\; \sum_{i=1}^{c} p(i|t)^2$$

Here, *t* stands for a given node, *i* is a class label in *c={active, non-active}*, and *p(i|t)* is the proportion of the samples that belongs to class *i* for a particular node *t*. Looking at the previous equation, it is easy to see that the impurity of a given node is minimal if the node is pure and only contains samples from one class (for examples, actives), since $1 - (1^2 + 0^2) = 0$. Vice versa, if samples at one node are perfectly mixed, the Gini impurity of a node is maximal: $1 - (0.5^2 + 0.5^2) = 0.5$. In an iterative process, the splitting

procedure is then repeated at each child node until the leaves of the tree are pure, which means that the samples at each node all belong to the same class (either *active* or *non-active*), or cannot be separated further due to the lack of discriminatory information in the dataset. For more information about the decision tree learning, see (30, 31).

To build a decision tree classifier (as opposed to a decision tree regressor), we discretize the signal inhibition variable, creating a binary target variable `y_binary`. Using the code in Figure 10, active molecules are specified as molecules with signal inhibition of 60% or greater (class 1), and molecules with less than 60% signal inhibition are labeled as non-active (class 0):

```
>>> y_binary = np.where(y >= 0.6, 1, 0)
>>> np.sum(y_binary)
12
```

Figure 10. Code for discretizing the continuous signal inhibition variable. The `numpy.where` function creates a new array, `y_binary`, where all molecules with more than 60% signal inhibition will be labeled as 1 (active), and all other molecules will be labeled with a 0 (non-active).

As can be seen from computing the sum of values in the `y_binary` array (`np.sum(y_binary)`, Figure 10), discretization of the continuous signal inhibition variable resulted in 12 molecules labeled as active; consequently, the remaining 44 molecules in the dataset are now labeled as non-active. In the next step, we will initialize a decision tree classifier from scikit-learn with default values, let it learn the decision rules that discriminate between actives and non-actives from the dataset, and export the model and display it as a decision tree (Figure 11). (See Note 2 in section 4.3 for a comment about decision trees for regression analysis.)

```
>>> from sklearn import tree
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, y_binary)
>>> dot_data = tree.export_graphviz(clf, out_file=None,
...                                 feature_names=fgroup_cols,
...                                 class_names=['non-active', 'active'],
...                                 filled=True, rounded=True)
>>> graph = pydotplus.graph_from_dot_data(dot_data)
>>> graph.write_pdf("tree.pdf")
```

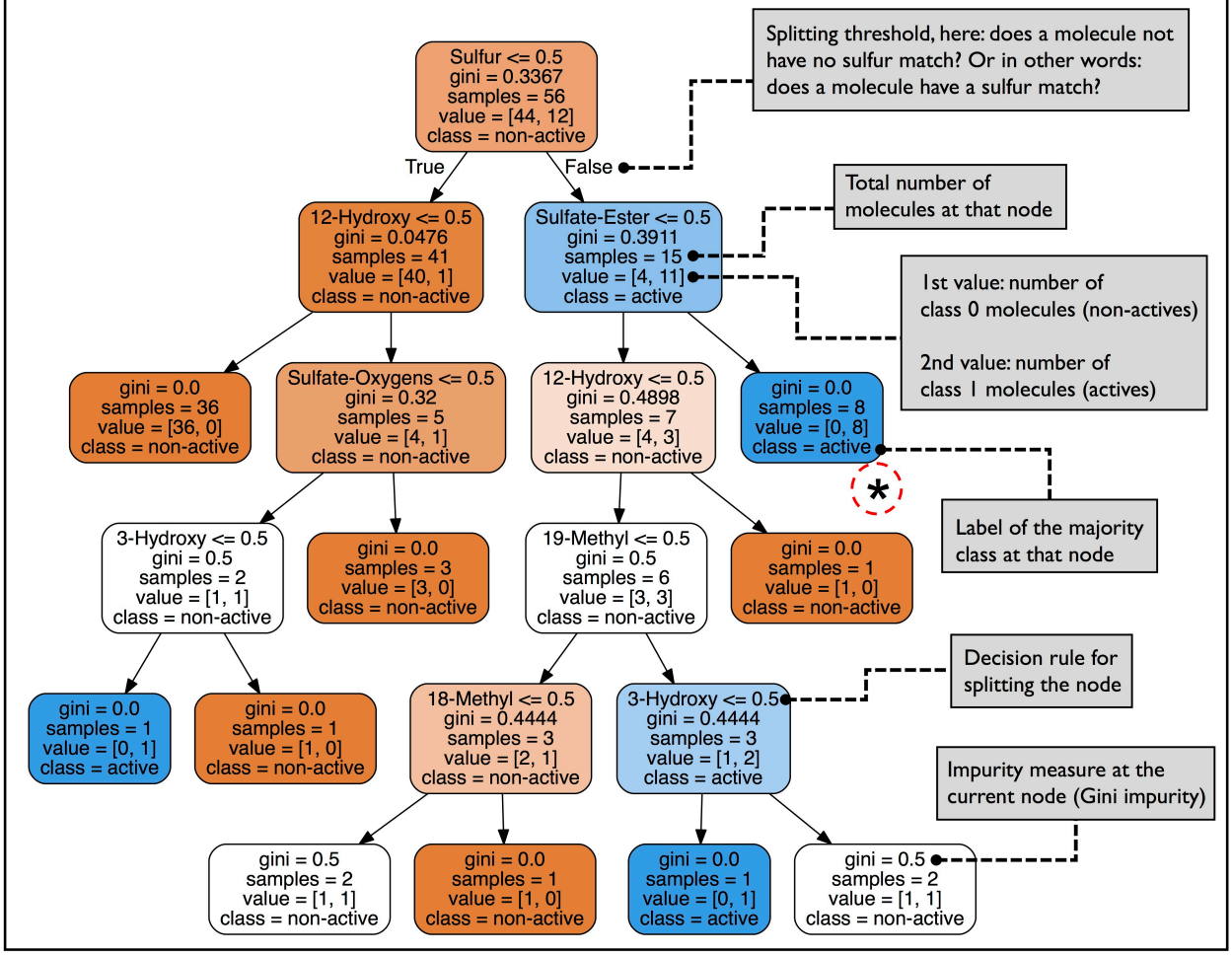

Figure 11. Binary classification tree separating active from non-active compounds. After importing the `tree` submodule from the scikit-learn machine learning library, the first line of code initializes a new `DecisionTreeClassifier` object that is then learning the decision rules from the functional group matching pattern array (`X`) and the discretized response variable (binary labels of the active and non-active molecules, `y_binary`) by calling the `fit` method. The last three lines of code then export the fitted decision tree as a PDF image, which is shown here. The first node at the top of the tree, for example, uses a decision rule asking which molecules in the 56-molecule dataset (44 actives and 12 non-actives) match a sulfur group in DKPES. Note that this question is posed as a conditional (true/false) statement "Molecules do *not* contain a sulfur group match", due to the implementation of the decision tree in scikit-learn. The molecules for which the condition is "False" — that is, molecules that do match the sulfur group in DKPES — are then passed to the child node on the right (here: 4 non-actives and 11 actives), where the next conditional statement is "Molecules do not contain a 'Sulfate-Ester' match." Each node in the tree contains the impurity measure after the split (Gini impurity), reflecting the degree of separation between active and non-active compounds; a Gini impurity value of 0 reflects a set containing purely active or non-active compounds. The number of samples refers to the compounds at each node that pass the filtering criteria. The first value within brackets in the bottom row in each terminal node denotes the number of non-active compounds at that node, and the second number denotes the number of active compounds. Highlighted with an asterisk is the terminal node (to the

center-right of the plot), which contains eight active compounds and no non-active compounds. For visual clarity, containing more non-active molecules than actives are labeled in orange, and nodes that contain more actives than actives are colored in blue. The higher the color intensity, the higher the ratio of active molecules or non-active molecules, respectively.

We conclude from the binary classification tree (Figure 11) that a majority of the active inhibitors (8 of 12) share a sulfur atom and a sulfate ester group that overlay with the respective functional groups in DKPES; none of the non-active compounds have these characteristics. With decision trees, the resulting models can offer intuitive insights into the hypothesis space. Specifically, the tree in Figure 11 indicates that, given a set of molecules initially selected as having high volumetric and chemical similarity with DKPES, the presence of a sulfur atom and sulfate ester group matching those two groups in DKPES predicts the subset of molecules that are active as DKPES inhibitors. Using machine learning to derive decision rules objectively and automatically is convenient and less error-prone in providing insights compared with visual analysis of functional group patterns in a heat map. (See Note 3 in section 4.3 for a comment on building predictive classifiers.)

3.4 Deducing the Importance of Chemical Groups via Random Forest

To estimate the relative importance of the different functional groups based on active and non-active labels, we will now construct a random forest model (16), which is an ensemble method based on multiple decision trees. In the random forest models, the feature importance is measured as the averaged impurity decrease computed from multiple decision trees. In the following code example (Figure 12), we will use the random forest algorithm implemented in scikit-learn to create and ensemble of 1,000 decision trees, which are grown from different bootstrap samples of the compound dataset and randomly selected subsets of functional group feature variables. (A bootstrap sample is generated by randomly drawing samples from the original dataset with replacement to generate a resampled dataset of the same size as the original one.)

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> forest = RandomForestClassifier(n_estimators=1000,
...                                 random_state=0,
...                                 n_jobs=-1)
>>> forest.fit(X, y_binary)
```

Figure 12. Similar to fitting a `DecisionTreeClassifier` (Figure 11), we first initialize a new `RandomForestClassifier` object from scikit-learn and fit it to the functional group matching pattern array (`X`) and labels of the active and non-active molecules (`y_binary`). By setting `n_estimators=1000`, we will use 1000 decision trees for the forest. `n_jobs=-1` means that we are utilizing all processors on our machine to fit those decision trees in parallel to speed up the computation. The `random_state` parameter accepts an arbitrary integer for the bootstrap sampling and feature selection in the decision tree to make the experiment deterministic and reproducible.

Based on the random forest model, we can infer feature importance by averaging the impurity decrease for each feature split from all 1000 trees in the forest. Conveniently, the random forest implementation in scikit-learn already computes the feature importance upon model fitting, so that we can access this information from the forest, after calling the `fit()` method via its `feature_importances_` attribute. The code in Figure 13 will create a bar plot of the feature importance values, which are normalized to sum up to 1 for easier interpretation.

```
>>> importances = forest.feature_importances_
>>> indices = np.argsort(importances)[::-1]
>>> feature_labels = np.array(fgroup_cols)
>>> plt.bar(range(X.shape[1]),
>>>         importances[indices],
>>>         align='center')
>>> plt.xticks(range(X.shape[1]),
...             feature_labels[indices], rotation=90)
>>> plt.xlim([-1, X.shape[1]])
>>> plt.ylabel('Relative feature importance')
>>> plt.tight_layout()
>>> plt.show()
```

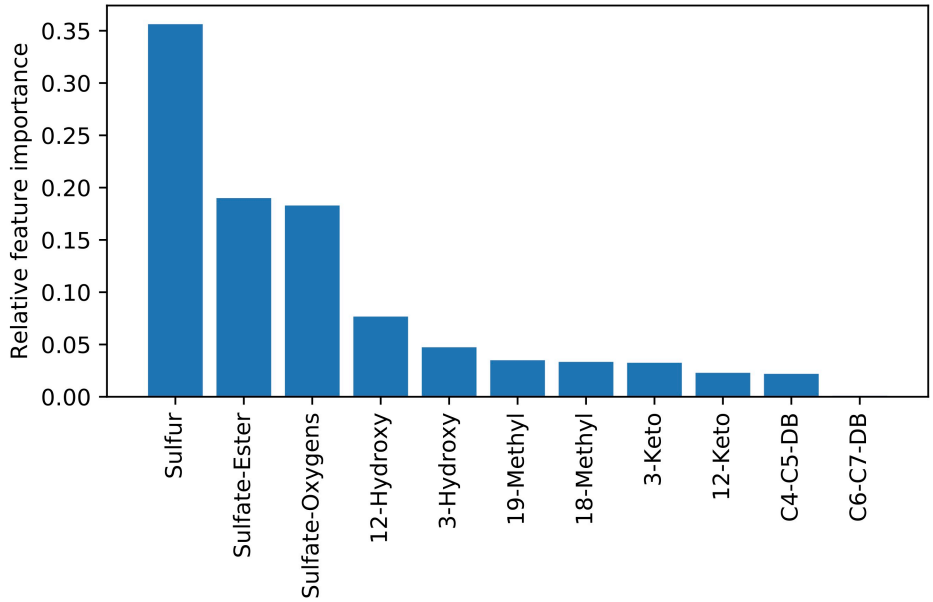

Figure 13. Relative feature importance of the functional group matches inferred from the random forest model that was trained to discriminate between active and non-active molecules. First, the `importances` values are sorted from highest to lowest using NumPy's `argsort` function. Next, we summarize the computed feature importance in a bar plot using matplotlib's `pyplot` submodule, which was imported as `plt` earlier.

As shown by the bar plot in Figure 13, the feature importance values computed from the 1,000 regression trees agree with the conclusions drawn previously in the Methods sections 3.3 and 3.4: sulfur, sulfate ester, and sulfate oxygen groups are the most important functional group features for DKPES inhibitor activity (Figure 11). (See Notes 1 and 2 in section 4.4 for additional tips concerning the interpretation of random forest feature importance values.)

3.5 Sequential Feature Selection with Logistic Regression

As an alternative approach and to probe the robustness of our conclusions, we will apply a Sequential Backward Selection (SBS) algorithm combined with logistic regression (32) for the classification of active versus non-active compounds. SBS is a model-agnostic feature selection algorithm that evaluates different combinations of features, shrinking the subset of features to be considered one by one. Here, model-agnostic refers to the fact that SBS can be combined with any machine learning algorithm for classification or regression.

The SBS algorithm works as follows. This removal of features is repeated in each iteration of the algorithm until the desired, pre-specified size of the feature subset is reached. The reason why we chose sequential feature selection to deduce functional group matching patterns that are predictive of active and non-active molecules is that it presents an intuitive method that has shown to produce accurate and robust results. Alternative approaches are discussed in Note 1 in section 4.5. For more information on sequential feature selection, please read (17).

Logistic regression is one of the most widely used classification algorithms in academia and industry. One of the reasons why logistic regression is a popular choice for predictive modeling is that it is easy to interpret as a generalized linear model: The output always depends on the sum of the inputs and model parameters. However, note that sequential feature selection can be used with many different machine learning algorithms for supervised learning (classification and regression).

To introduce the main concept behind logistic regression, which is a probabilistic model, we need to introduce the so-called *odds ratio* first. The odds ratio computes the *odds* in favor of a particular event *E*, which is defined as follows, based on the probability *p* of a positive outcome (for instance, the probability that a molecule is active):

$$\text{odds} = \frac{p}{(1-p)}$$

Next, we define the *logit* function, which is the logarithm of the odds ratio:

$$\text{logit}(p) \; = \; \log\frac{p}{(1-p)}$$

The logit function takes values over the range 0 to 1 (the probability *p*) and transforms them to real numbers that describe the relationship between the functional group matching patterns, multiplied with weight coefficients (that need to be learned) and the odds that a given molecule is active:

$$\text{logit}(p(y=1|x)) \; = \; w_1x_1 \; + \; w_2x_2 \; + \cdots + w_mx_m + \text{b} = \sum_{i=1}^{m} w_ix_i + \text{b}$$

Here, *m* is an index over the input features (functional group matches, *x*), *w* refers to the weight parameters of the parametric logistic regression model, and *b* refers to the y-axis intercept (typically referred to as *bias* or *bias unit* in literature). The input to the logit function, $p(y=1|x)$, is the conditional probability that a particular molecule is active, given that its functional group matches *x*.

However, since we are interested in modeling the probability that a given molecule is active, we need to compute the function inverse ϕ of the logit function, which we can compute as:

$$\phi(z) \; = \; \frac{1}{1+e^{-z}}$$

Here, *z* is a placeholder variable defined as follows:

$$z = \sum_{i=1}^{m} w_ix_i + \text{b}$$

The logistic regression implementation used in this section learns the weights for the parameters (matched chemical features) of the logistic regression model that minimizes the logistic cost function, which is the probability of making a wrong prediction given the number of *n* active and non-active molecule labels in the set of compounds, where the binary vector *y* stores the class labels (1=active, 0=non-active):

$$l(w) = \sum_{i=1}^{n} \left[y^{(i)} \log(\phi(z^{(i)})) + (1 - y^{(i)}) \log(1 - \phi(z^{(i)})) \right]$$

For more information about logistic regression, see reference (32).

Now, by combining a logistic regression classifier with a sequential feature selection algorithm, we can identify a fixed-size subset of functional groups that maximizes the probability of correct prediction of which compounds are active.

Since we are interested in comparing feature subsets of different sizes to identify the smallest feature set with the best performance, we can run the SBS algorithm stepwise down to a set with only one feature, allowing it to evaluate feature subsets of all sizes, by using the code shown in Figure 14. Furthermore, the SBS implementation uses k-fold cross-validation for internal performance validation and selection. In particular, we are going to use 5-fold cross-validation.

In 5-fold cross-validation, the dataset is randomly split into k non-overlapping subsets or folds (a molecule cannot be in multiple subsets). From the five splits, four folds are used to fit the logistic regression model, and one fold is used to compute the predictive performance of the model on held-out (test) data. 5-fold cross-validation repeats this splitting procedure five times so that we obtain five models and performance estimates. The model performance is then computed as the arithmetic average of the five performance estimates. For more details about k-fold cross-validation, please see the online article, "Model evaluation, model selection, and algorithm selection in machine learning — Cross-validation and hyperparameter tuning" at https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html. Also, see Note 2 in section 4.5 for more information on choosing the number of folds in k-fold cross-validation.

```
>>> from mlxtend.feature_selection import SequentialFeatureSelector as SFS
>>> from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
>>> from sklearn.linear_model import LogisticRegression
>>> classifier = LogisticRegression()
>>> sfs = SFS(classifier,
...           k_features=1,
...           forward=False,
...           floating=False,
...           scoring='accuracy',
...           verbose=0,
...           cv=5)
>>> sfs = sfs.fit(X, y_binary)
>>> fig1 = plot_sfs(sfs.get_metric_dict(), kind='std_err')
>>> plt.ylim([0.5, 1])
>>> plt.ylabel('Accuracy')
>>> plt.xlabel('Number of features in the selected subset')
>>> plt.grid()
>>> plt.show()
```

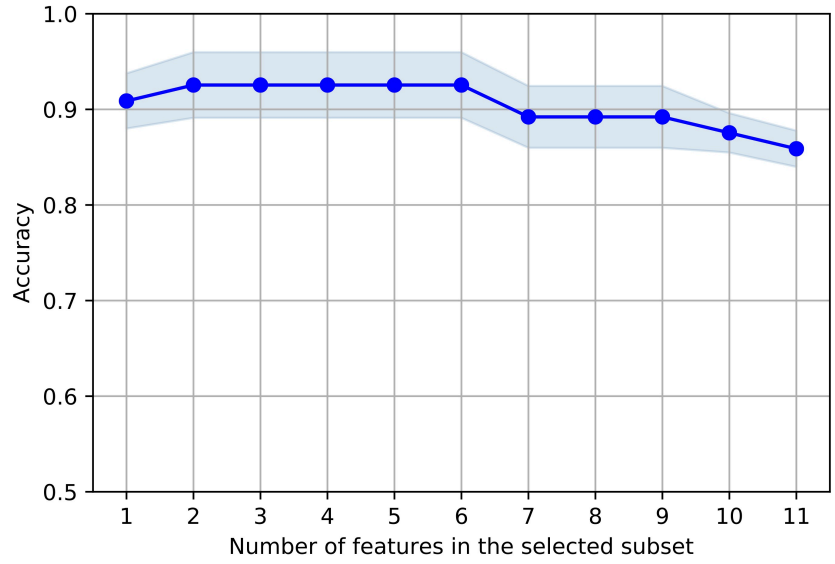

Figure 14. Performing sequential feature selection using logistic regression to identify features that discriminate between active and non-active molecules. After importing the Python classes for fitting the `LogisticRegression` classifier within the `SequentialFeatureSelector`, by setting `forward=False` and `floating=False`, we specify that the sequential feature selector should perform regular backward selection. Then we use the `plot_sfs` function to visualize the results with matplotlib's pyplot submodule. The resulting plot in this figure shows the classification accuracy of the logistic regression models trained on different feature subsets (functional group matching patterns) via sequential backward selection. The prediction accuracy (0=worst, 1=best), where 1 corresponds to 100% accuracy in predicting active versus non-active compounds across the input set, was then computed via 5-fold cross validation. The plot presents the average prediction accuracy (whether the model can predict held-out active and non-active molecules given their functional group matching patterns) across the 5 different test sets. The error margin (pale blue region above and below the dark blue average points) shows the standard error of the mean for the 5-fold cross validation.

As can be seen in Figure 14, the performance of the classification algorithm does not change significantly across the different feature subset sizes. The feature subsets with size 2-6 have the highest accuracy, indicating that adding more features to the 2-feature subset does not provide additional discrimination between active and non-active molecules. The decline in accuracy after adding a seventh feature to the set is likely due to the curse of dimensionality (33). In brief, the curse of dimensionality describes the phenomenon that a feature space becomes increasingly sparse if we increase the number of dimensions (for instance, by adding additional functional group matching features) given a fixed-number of samples

in the training set, which will more likely result in overfitting and less accurate results. While the execution of the code in Figure 14 provided us with insights regarding the best-performing feature subset sizes via SBS in predicting active or non-active molecules, we haven't determined what those features are. Since there is no information gain by going beyond two-feature set (Figure 14), we will use the following code (Figure 15) to extract the feature names:

```
>>> sfs.subsets_[2]
{'avg_score': 0.92545454545454553,
 'cv_scores': array([ 1., 1., 0.81818182, 0.90909091, 0.9 ]),
 'feature_idx': (10, 6)}
```

Figure 15. Code to obtain the feature names of the best-performing feature subset from sequential backward selection (Figure 14). The `subsets_` attribute of the sequential feature selector (`sfs`) refers to a Python dictionary that stores the feature (functional group matches) indices and cross-validation information. By looking up the dictionary entry at index position `2`, we can access the feature indices of the 2-feature subset, `10` and `6`, and by using `sfs.subsets_[2]` as an index to the `feature_labels` array that we defined earlier (Figure 13) and reporting the feature labels, we can see that `"Sulfur"` and `"Sulfate-Ester"` matches are the most discriminatory features of active and non-active molecules.

The output from the code executed in Figure 14 shows that the 2-feature subset consisting of 'Sulfur' and 'Sulfate-Ester' matches has the most discriminatory information for separating active and non-active molecules as DKPES mimics. This information is consistent with the conclusions drawn from the previous random forest and decision tree analyses.

Now we have shown how to use decision trees, random forest models, and logistic regression to analyze which features can best discriminate between active and inactive compounds, and to assess the relative importance of the different features for discrimination. Such methods provide clearly interpretable information on chemical features important for activity, and concurrence between the methods strengthens the conclusions. In a related pheromone inhibitor project, we used the results of feature importance analysis to drive the selection of compounds in a subsequent round of virtual screening that required few compounds to be assayed and resulted in significant enhancement of activity and new knowledge about functional group importance. Those compounds are now being tested by members of our research team for invasive species behavioral modification in the tributaries of the Laurentian Great Lakes under an EPA permit (10). Analysis of whether the set of features and their relative importance hold equally well for different subsets of assayed compounds (e.g., steroids versus non-steroids) is another valuable direction of inquiry that is described under Note 3 in section 4.5.

3.6 Conclusion

From the decision tree analysis (section 3.3), random forest feature importance estimation (section 3.4), and sequential feature selection results (sections 3.5), we can conclude that the sulfate groups (*Sulfur*, *Sulfate-Ester,* and *Sulfur-Oxygen* features) are the most discriminatory features for distinguishing active from non-active compounds in DKPES-mediated olfactory responses. From the inspection of heat maps showing the top 10 active and 10 least active molecules (section 3.2), we also observed that presence of sulfate tails are a consistent determinant of activity. One compound consisting only of a sulfate tail (ZINC14591952, Figure 8) resulted in 62% signal inhibition, which supports the hypothesis that sulfate groups are a key feature of active molecules. Figure 16 summarizes the results from the random forest feature importance estimation by comparing the importance values to the proportion of functional group matches in active and non-active molecules.

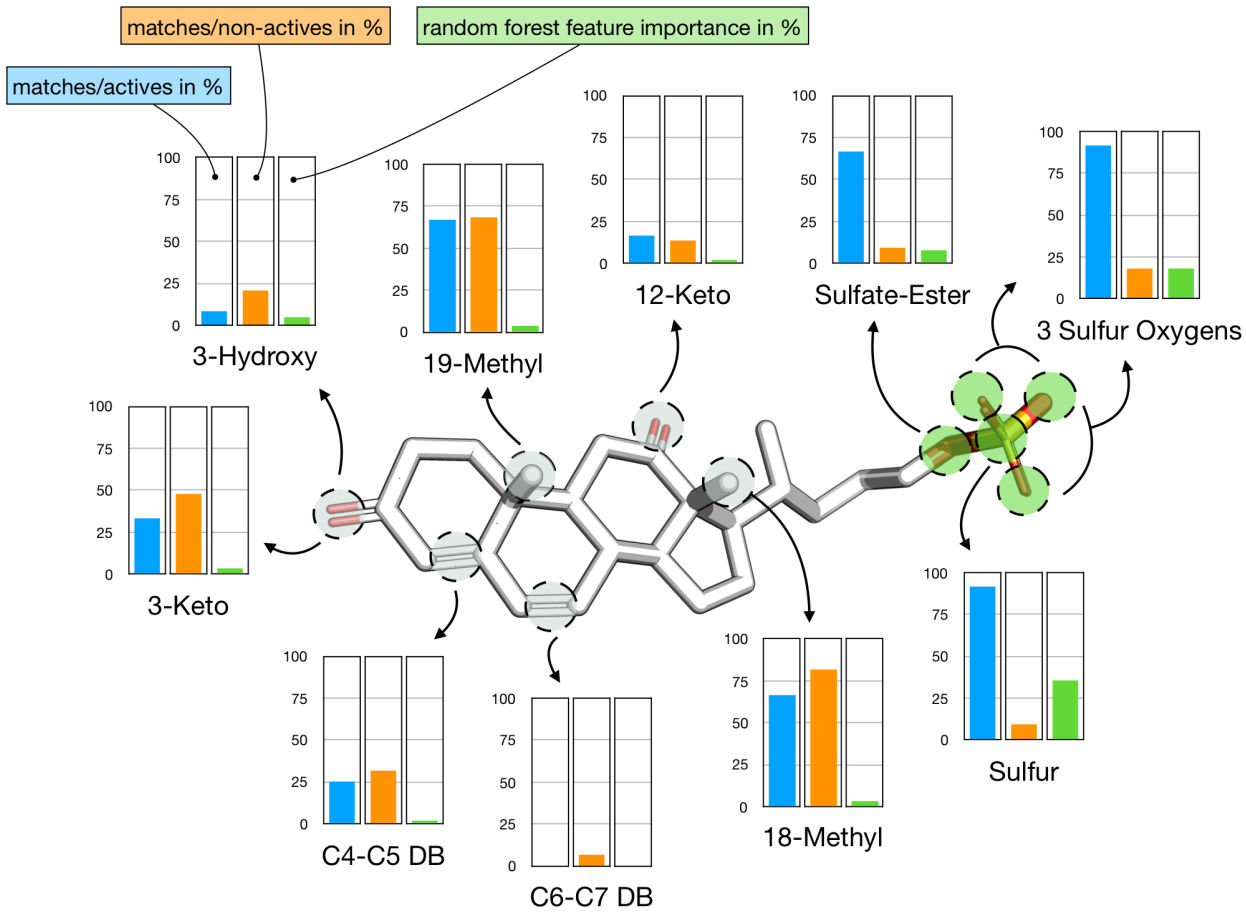

Figure 16. Proportion of functional group matches across the 12 active and 44 inactive molecules and relative functional group feature importance (from the random forest analysis, Figure 13) mapped onto the DKPES reference molecule. DB refers to "double bond."

The data in Figure 16 shows that matching 'Sulfur' and 'Sulfur Oxygens' are the most discriminatory features for a random forest to distinguish actives from non-actives, and both features also have a high rate of occurrence in active versus non-active molecules. Features that do not appear substantially more frequently in active molecules than in non-actives (are not discriminatory of activity), for example, 18-methyl, 19-methyl, 3-keto, or the presence of either the C4-C5 or C6-C7 double bond ("DB") also have a low random forest feature importance. Interestingly, the feature importance of Sulfate-Ester is much less than the feature importance of Sulfur or Sulfur-Oxygens, which may be because it is highly correlated with the sulfur and sulfur oxygen matches in the sulfate group and thereby, to some extent, redundant. An alternative explanation is that the ester oxygen is less highly charged than the terminal sulfate oxygens (causing it to make weaker hydrogen bonds) and is also less accessible for interaction with the receptor.

The machine learning techniques presented in this chapter can be used for *any* kind of data for which a set of feature values across a set of objects is used to predict activity (or any observable value determined by an experimental technique, for example, solubility, selectivity, reactivity, and so forth). We hope this chapter has whetted your appetite for machine learning, which provides robust models that relate features of interest to molecular activity and other observables. The code provided here and on the corresponding website (https://github.com/psa-lab/predicting-activity-by-machine-learning) makes it possible for you to learn and then use these techniques in your own research. For further information about machine learning,

and to carry out further explorations with prepared datasets or your own data, we recommend the following tutorials and references: Raschka & Mirjalili 2017 (34), Raschka *et al.* 2016 (35), Friedman *et al.* 2001 (36), Mueller & Guido 2016 (37), and the scikit-learn online tutorials (http://scikit-learn.org/stable/tutorial/index.html).

4. Notes

4.1 Loading and Inspecting the Biological Activity Dataset

1. For this section, we used a CSV file where the features and target variable (signal inhibition) were stored as columns separated by commas. Note that the `read_csv` function does not strictly require this input format. For instance, pandas's `read_csv` function supports any possible column delimiter (for example, tabs, whitespaces, and so forth), which can be specified via the `delimiter` function argument. For more information about the `read_csv` function, please refer to the official documentation at https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html. Furthermore, if you are planning to work with datasets where the features are stored as rows as opposed to columns, you can use the transpose method (`df = df.transpose()`) after loading a dataset to transpose the data frame index and columns.

2. Throughout the Methods sections, we assumed that the data frame of activity data was already sorted by signal inhibition in decreasing order. While sorting the data frame is not essential for fitting the machine learning models in the later section, you may consider sorting your datasets for the heat map visualization, to show the 10 molecules with the highest inhibition activity, for example. To sort the data frame `df`, you can use `sort_values` method of a given pandas data frame object. For example, the following code sorts the molecules stored as a data frame `df` from most active to least active: `df = df.sort_values('Signal-Inhibition', ascending=False)`. More information about this `sort_values` method can be found in the official pandas documentation at https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sort_values.html.

3. While we recommend working with 3D structures because they provide spatial relationships between chemical groups, molecular features can also be derived from 1D string representations of molecules or 2D structural representations. For example, the presence of certain substructures or atom types, using so-called molecular fingerprints, can be computed using the open-source toolkit OpenBabel (https://openbabel.org/docs/dev/Fingerprints/intro.html).

4. To convert a 1D or 2D representation of a molecule into a 3D structure as input for the spatial functional group matching in the DKPES dataset that was done via Screenlamp (10) using ROCS overlays (OpenEye Scientific Software, Santa Fe, NM; https://www.eyesopen.com/rocs), you may find the following tools helpful:

    - The CACTUS online SMILES translator and structure file generator (https://cactus.nci.nih.gov/translate/).
    - OMEGA (OpenEye Scientific Software, Santa Fe, NM; https://www.eyesopen.com/omega), which creates multiple favorable 3D conformers of a given structure from 1D, 2D, or 3D representations (38, 39). This software is available free for academic researchers upon completion of a license agreement with OpenEye.

5. Further, you may find the BioPandas toolkit (40) helpful (http://rasbt.github.io/biopandas/), which reads 3D structures from the common MOL2 file format into the pandas data frame format. This allows users can be useful if you are working with large MOL2 databases that contain thousands or millions of structures that you want to filter for certain properties prior to generating overlays via ROCS or compute the functional group matching patterns via Screenlamp (https://github.com/psa-lab/screenlamp).

4.2 Chemical and Functional Groups

1. We chose a 1.3 Å cut-off between overlayed atoms to identify functional group matches in 3D. If two molecules share the same atom type at a distance greater than 1.3 Å, this was not considered a functional group match. This relatively generous distance cut-off (nearly a covalent bond-length) was chosen to account for minor deviations in the crystal structures and overlays when comparing functional groups between pairs of molecules. Note that changing the distance threshold generally will affect the resulting functional group matching patterns. For instance, the 3-hydroxy group in ZINC72400307 (Figure 5) does not overlay with the 3-keto group of the DKPES query (Figure 2) in our analysis since the distance between those two atoms is 1.7 Å. We recommend choosing distance thresholds up to 1.3 Å.

4.3 Tracing Preferential Chemical Group Patterns using Decision Trees

1. While there is technically no minimum number of molecules required for using the techniques outlined in this chapter, we recommend collecting datasets of at least 30 structures for the automatic inference of functional groups that discriminate between active and non-active molecules. Although this is difficult to achieve in practice, an ideal dataset for supervised machine learning would be balanced, that is, with an equal number of positive (active) and negative (non-active) training examples. While there is no indication that class imbalance was in issue for the DKPES dataset, as the results of the decision tree analysis were unambiguous, imbalance may be an issue in other datasets. There are many different techniques for dealing with imbalanced datasets, including several resampling techniques (oversampling of the minority class or under-sampling of the majority class), the generation of synthetic training samples, and reweighting the influence of different class labels during the model fitting. A comprehensive review of techniques for working with imbalanced datasets can be found in (34). For machine learning with scikit-learn, a compatible Python library that has been developed to deal with imbalanced datasets (http://contrib.scikit-learn.org/imbalanced-learn/) (35). Also note that classifiers in scikit-learn, including the `DecisionTreeClassifier`, accept a `class_weight` argument, which can be used to put more emphasis on a particular class (e.g., active or non-active) during model fitting, thereby preventing that the decision tree algorithm becomes biased towards the most frequent class in the dataset. For more information on how to use the `class_weight` parameter of the `DecisionTreeClassifier`, refer to the documentation at http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

2. Deep, unpruned decision trees with many decision points are notoriously prone to overfitting. This is analogous to the overfitting problem in parametric regression, where including more terms with adjustable weights allows better fit to a set of training data, while resulting in complex decision rules that are hard to interpret and do not perform well on held-out or new data. This is why we preferred classification trees over decision trees for regression analysis for the single decision tree and random forest analyses in this chapter.

3. Note that the problem analyzed here as a case study is not a classical example of machine learning, in which a classifier is fit to a training dataset, and then its accuracy of prediction (and generalizability to new data) is estimated on held-out data by using a test set or cross-validation techniques. In this chapter, we are describing general approaches for analyzing the importance of various functional groups for the activity of molecules. Our primary goal is not to build a predictor to classify new molecules as active or non-active, although the models developed in this chapter could indeed be used in such a way.

4.4 Deducing the Importance of Chemical Groups via Random Forests

1. While the feature importance values provide us with a numeric value to quantify the importance of features, these quantities do not provide information about whether the presence or absence of the particular functional group matches are characteristic of the active molecules. However, we can easily determine whether active molecules match a certain functional group by inspecting the heat map visualizations of active and non-active molecules (Figure 9).

2. Concerning the interpretation of feature importance values from random forests, note that if two or more features are highly correlated, one feature may be ranked much higher than the other feature, or both features may be equally ranked. In other words, the importance or information in the second feature may not be fully captured. The potential bias in interpreting the feature importance from random forest models has been discussed in more detail by Strobl *et al.* (41). In general, this issue can be pre-assessed by measuring the degree to which series of values for two features across a set of compounds are correlated by calculating the Pearson linear correlation coefficient or by calculating the Spearman rank correlation coefficient to assess similar ranking of values between the features across a set of compounds (which does not assume co-linearity). The Spearman and Pearson correlation coefficients can be computed using the `peasonr` and `spearmanr` functions from the `scipy.stats` package (please refer to the official SciPy documentation at https://docs.scipy.org/ for more information). While the predictive performance of a random forest is generally not negatively affected by high correlation among feature variables (multicollinearity), it is recommended to exclude highly correlated features from the dataset for feature importance analysis, for instance, via recursive feature importance pruning (42).

4.5 Sequential Feature Selection with Logistic Regression

1. Sequential feature selection constitutes just one of many approaches to select feature subsets. Univariate feature selection methods that consider one variable at a time and select features based on univariate statistical tests, for example, percentile thresholds or p-values. A good review of feature selection algorithms can be found in Saeys et al. 2007 (43). However, the main advantage of sequential feature selection over univariate feature selection techniques is that sequential feature selection analyzes the effect of features on the performance of a predictive model considering the features as a synergistic group. Other techniques, related to sequential feature selection, are genetic algorithms, which have been successfully used in biological applications to find optimal feature subsets in high-dimensional datasets as discussed in Raymer *et al* 1997 & 2000 (44, 45).

2. We chose 5-fold cross-validation to evaluate the logistic regression models in the sequential backward selection, since k=5 it is a commonly used default value in k-fold cross-validation. Generally, small values for k are computationally less expensive than larger values of k (due to the smaller training set sizes and fewer iterations). However, a choosing a small value for k increases the pessimistic bias, which means the performance estimate underestimates the true generalization performance of a model. On the other hand, increasing the size of k increases the variance of the

estimate. Unfortunately, the No Free Lunch Theorem (46) — stating that there is no algorithm or choice of parameters that is optimal for solving all problems — also applies here (as shown in (47). For an empirical study of bias, variance, and bias-variance trade-offs in cross-validation, also see (48).

3. The chemical features identified as most important by machine learning will depend on the chemical diversity within the set of molecules for which assay results and chemical structures are analyzed. For instance, if only steroid compounds are tested versus only non-steroids, likely the chemical features found to be most important will differ. In our case, for the steroid set, the side groups providing specific interactions were most important (since the steroid scaffold is in common to all of them), whereas for the non-steroids, compounds that mimic and shape and hydrophobic interactions of the steroidal pheromone may also be important. Thus, considering the set of compounds to be analyzed, and testing the generalizability of the features derived is worth some thought. If you have different chemical classes of compounds to analyze, and a significant number of compounds in each, you can carry out the machine learning analysis of the most important features for each of the groups of compounds separately, as well as all of the compounds together, to discern the extent to which highly ranked features that discriminate between actives and inactives are shared among compounds based on different chemical scaffolds.

Acknowledgements

This research was supported by funding from the Great Lakes Fishery Commission from 2012-present (Project ID: 2015_KUH_54031). We gratefully acknowledge OpenEye Scientific Software (Santa Fe, NM) for providing academic licenses for the use of their ROCS, Omega, QUACPAC (molcharge), and OEChem toolkit software. We also wish to express our special appreciation to the open source community for developing and sharing the freely accessible Python libraries for data processing, machine learning, and plotting that were used for the data analysis presented in this chapter.

5. References

1. Ripphausen P, Nisius B, Bajorath J (2011) State-of-the-art in ligand-based virtual screening. Drug Discov Today 16:372–376.
2. Geppert H, Vogt M, Bajorath J (2010) Current trends in ligand-based virtual screening: molecular representations, data mining methods, new application areas, and performance evaluation. J Chem Inf Model 50:205–216.
3. Pérez-Nueno VI, Ritchie DW, Rabal O, Pascual R, Borrell JI, Teixidó J (2008) Comparison of ligand-based and receptor-based virtual screening of HIV entry inhibitors for the CXCR4 and CCR5 Receptors using 3D Ligand shape matching and ligand-receptor docking. J Chem Inf Model 48:509–533.
4. Hawkins PCD, Skillman AG, Nicholls A (2007) Comparison of shape-matching and docking as virtual screening tools comparison of shape-matching and docking as virtual screening tools. 50:74–82.
5. Sukuru SCK, Crepin T, Milev Y, Marsh LC, Hill JB, Anderson RJ, Morris JC, Rohatgi A, O'Mahony G, Grøtli M, others (2006) Discovering new classes of Brugia malayi asparaginyl-tRNA synthetase inhibitors and relating specificity to conformational change. J Comput Aided Mol Des 20:159–178.
6. Lyne PD (2002) Structure-based virtual screening: An overview. Drug Discov Today 7:1047–1055.

7. Ghosh S, Nie A, An J, Huang Z (2006) Structure-based virtual screening of chemical libraries for drug discovery. Curr Opin Chem Biol 10:194–202.
8. Li Q, Shah S (2017) Structure-based virtual screening. Protein Bioinforma From Protein Modif Networks to Proteomics 111–124.
9. Yan X, Liao C, Liu Z, T Hagler A, Gu Q, Xu J (2016) Chemical structure similarity search for ligand-based virtual screening: Methods and computational resources. Curr Drug Targets 17:1580–1585.
10. Raschka S, Scott AM, Liu N, Gunturu S, Huertas M, Li W, Kuhn LA (2017) Enabling hypothesis-driven prioritization of small molecules in big databases: Screenlamp and its application to GPCR inhibitor discovery. Submitted
11. Zavodszky MI, Rohatgi A, Van Voorst JR, Yan H, Kuhn LA (2009) Scoring ligand similarity in structure-based virtual screening. J Mol Recognit 22:280–292.
12. Buhrow L, Hiser C, Van Voorst JR, Ferguson-Miller S, Kuhn LA (2013) Computational prediction and in vitro analysis of potential physiological ligands of the bile acid binding site in cytochrome c oxidase. Biochemistry 52:6995–7006.
13. Kubinyi H, Folkers G, Martin YC (2006) 3D QSAR in drug design: recent advances. Springer Science & Business Media, Berlin.
14. Verma J, Khedkar VM, Coutinho EC (2010) 3D-QSAR in drug design-a review. Curr Top Med Chem 10:95–115.
15. Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC Press, Boca Raton
16. Breiman L (2001) Random forests. Mach Learn 45:5–32.
17. Ferri F, Pudil P, Hatef M, Kittler J (1994) Comparative study of techniques for large-scale feature selection. Pattern Recognit Pract IV 1994:403–413.
18. Raschka S, (2017) rasbt/mlxtend: Version 0.7.0, doi: 10.5281/zenodo.816309
19. Hansen GJA, Jones ML (2008) A rapid assessment approach to prioritizing streams for control of Great Lakes sea lampreys (Petromyzon marinus): a case study in adaptive management. Can J Fish Aquat Sci 65:2471–2484.
20. Irwin JJ, Shoichet BK (2005) ZINC—a free database of commercially available compounds for virtual screening. J Chem Inf Model 45:177–82.
21. Allen F (2002) The Cambridge Structural Database: A quarter of a million crystal structures and rising. Acta Crystallogr Sect B Struct Sci 58:380–8.
22. Johnson NS, Yun S-S, Li W (2014) Investigations of novel unsaturated bile salts of male sea lamprey as potential chemical cues. J Chem Ecol 40:1152–1160.
23. Van Rossum G (2007) Python programming language. In: USENIX Annu. Tech. Conf. p 36
24. Van Der Walt S, Colbert SC, Varoquaux G (2011) The NumPy array: a structure for efficient numerical computation. Comput Sci Eng 13:22–30.
25. Jones E, Oliphant T, Peterson P (2001) SciPy: Open source scientific tools for Python. http//www.scipy.org/
26. McKinney W, others (2010) Data structures for statistical computing in Python. In: Millman J, vand der Walt S (eds) Proc. 9th Python Sci. Conf. pp 51–56
27. Hunter JD (2007) Matplotlib: A 2D graphics environment. Comput Sci Eng 9:90–95.
28. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830.
29. Aiello A, Carbonelli S, Esposito G, Fattorusso E, Iuvone T, Menna M (2000) Novel bioactive sulfated alkene and alkanes from the Mediterranean ascidian Halocynthia papillosa. J Nat Prod 63:1590–1592.
30. Raschka S (2015) Python Machine Learning, 1st ed. Packt Publishing, Birmingham
31. Louppe G (2014) Understanding random forests: From theory to practice. Ph.D. thesis.
32. Walker SH, Duncan DB (1967) Estimation of the probability of an event as a function of several

independent variables. Biometrika 54:167–179.
33. Hughes G (1968) On the mean accuracy of statistical pattern recognizers. IEEE Trans Inf theory 14:55–63.
34. Raschka S, Mirjalili V (2017) Python machine learning, 2nd ed. Packt Publishing, Birmingham
35. Raschka S, Julian D, Hearty J (2016) Python: Deeper insights into machine learning, 1st ed. Packt Publishing, Birmingham
36. Hastie T, Tibshirani R, Friedman J, Hastie T, Tibshirani R (2001). Springer series in statistics, New York
37. Müller AC, Guido S (2017) Introduction to machine learning with Python: a guide for data scientists. O'Reilly Media, Sebastopol.
38. Hawkins PCD, Skillman AG, Warren GL, Ellingson BA, Stahl MT (2010) Conformer generation with OMEGA: Algorithm and validation using high quality structures from the Protein Databank and Cambridge Structural Database. J Chem Inf Model 50:572–84.
39. Hawkins PCD, Nicholls A (2012) Conformer generation with OMEGA: learning from the data set and the analysis of failures. J Chem Inf Model 52:2919–2936.
40. Raschka S (2017) BioPandas: Working with molecular structures in pandas DataFrames. J Open Source Softw.
41. Strobl C, Boulesteix A, Kneib T, Augustin T, Zeileis A (2008) Conditional variable importance for random forests. BMC bioinformatics 9:307.
42. Strobl C, Malley J, Tutz G (2009) An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. Psychol Methods 14:323.
43. Saeys Y, Inza I, Larrañaga P (2007) A review of feature selection techniques in bioinformatics. Bioinformatics 23:2507–2517. doi: 10.1093/bioinformatics/btm344
44. Raymer ML, Punch WF, Goodman ED, Kuhn LA, Jain AK (2000) Dimensionality reduction using genetic algorithms. IEEE Trans Evol Comput 4:164–171.
45. Raymer ML, Sanschagrin PC, Punch WF, Venkataraman S, Goodman ED, Kuhn LA (1997) Predicting conserved water-mediated and polar ligand interactions in proteins using a K-nearest-neighbors genetic algorithm. J Mol Biol 265:445–464.
46. Wolpert DH (1996) The lack of a priori distinctions between learning algorithms. Neural Comput 8:1341–1390.
47. Bengio Y, Grandvalet Y (2004) No unbiased estimator of the variance of k-fold cross-validation. J Mach Learn Res 5:1089–1105.
48. Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. Int Jt Conf Artif Intell 14:1137–1143.