

CHAPTER 1

WHAT ARE OUR MODELS REALLY TELLING US? A PRACTICAL TUTORIAL ON AVOIDING COMMON MISTAKES WHEN BUILDING PREDICTIVE MODELS

W. PATRICK WALTERS

1.1 INTRODUCTION

Predictive models have become a common part of modern day drug discovery [1]. Models are used to predict a range of key parameters including:

- Physical properties such as aqueous solubility or octanol/water partition coefficients [2–4]
- Off-target activities such as CYP or hERG inhibition [5–7]
- Binding geometry and affinity of small molecules in protein targets [8].

When building these models, it is essential that the cheminformatics practitioner be aware of factors that could potentially mislead and confuse those using the models. In this chapter, we will focus on some common traps and pitfalls and discuss strategies for realistic evaluation of models.

We will consider a few important, and often overlooked, issues in the model-building process.

- How does the dynamic range of the data being modeled impact the apparent performance of the model?
- How does experimental error impact the apparent predictivity of a model?

- How can we determine whether a model is applicable to a new dataset?
- How should we compare the performance of regression models?

The chapter will take a tutorial format. We will analyze some commonly used datasets and use this analysis to make a few points about the process of building and evaluating predictive models. One of the most important aspects of scientific investigation is reproducibility. As such, all of the analyses discussed in this chapter were performed using readily available, open source software. This makes it possible for the reader to follow along, carry out the analyses, and experiment with the datasets. All of the code used to perform the analyses is available in the listings section at the end of the chapter. The datasets and scripts used in this chapter can also be downloaded from the author's website <https://github.com/PatWalters/cheminformaticsbook>. It is hoped that these scripts will kindle an appreciation for aspects of the model-building process and will provide the basis for further exploration.

The software tools required for the analyses are

The Python programming language – <http://www.python.org>

The RDKit cheminformatics programming library – <http://www.rdkit.org>

The R statistics program – <http://www.r-project.org>

Python scripts can be run by executing the command

```
python script_name.py (Unix and OS-X)
python.exe script_name.py (Windows)
```

where `script_name.py` is the name of the script to run.

R scripts can be run by executing the following two commands within the R console.

```
setwd("directory_path")
source("script.R")
```

In the aforementioned commands, “`directory_path`” is the full path to the directory (folder) containing the scripts and data, and “`script.R`” is the name of the script to execute. The R scripts used in this chapter utilize a number of libraries that are not included as part of the base R distribution. These libraries can be easily installed by typing the command

```
source("install_libraries.R")
```

in the R console. Since these libraries are being downloaded from the Internet, it is necessary for your computer to be connected to the Internet when executing the aforementioned command.

Those unfamiliar with Python or R are urged to consult references associated with those languages [9–12]. We now live in a data rich world where every cheminformatics practitioner should possess at least rudimentary programming skills.

1.2 PRELIMINARIES

In order to better understand some of the nuances associated with the construction and evaluation of predictive models, it is useful to consider actual examples. In this chapter, we will examine a number of datasets containing measured values for aqueous solubility and use these datasets to build and evaluate predictive models. Solubility in water or buffer is an important parameter in drug discovery [13]. Poorly soluble compounds tend to have poor pharmacokinetics and can precipitate or cause other problems in assays. As such, the prediction of aqueous solubility has been an area of high interest in the pharmaceutical industry. Over the last 15 years, numerous papers have been published on methods for predicting aqueous solubility [2, 3, 14]. Although many papers have been published and commercial software for predicting aqueous solubility has been released, reliable solubility prediction remains a challenge.

The challenges in developing models for predicting solubility can arise from a number of experimental factors. The aqueous solubility of a compound can vary depending on a number of factors including:

- Temperature at which the solubility measurement is performed
- Purity of the compound
- Crystal form—different polymorphs of the same compound can have vastly different solubilities.

In addition to confounding experimental factors, a number of published solubility models are somewhat misleading due to a lack of proper computational controls. While we sometimes have limited control over the experimental data used to build models, we have complete control over the way models are evaluated and should always employ appropriate means of evaluating our models. In subsequent sections, we will use solubility datasets to examine some of these control strategies.

1.3 DATASETS

In this chapter, we will consider three different, publicly available, solubility datasets.

The Huuskonen Dataset This set of 1274 experimental solubility values (Log *S*) was one of the first large solubility datasets published [15, 16] and has subsequently been used in a number of other publications [14, 17]. The data in this set was extracted from the AQUASOL [18, 19] database, compiled by the Yalkowsky group at the

University of Arizona and the PHYSPROP [20] database, compiled by the Syracuse Research Corporation.

The JCIM Dataset This is a set of 94 experimental solubility values that were published as the training set for a “blind challenge” published in 2008 [21]. All of the solubility values reported in this paper were measured by a single group under a consistent set of conditions. The objective of this challenge was for groups to use a consistently measured set of solubility values to build a model that could subsequently be used to predict the solubility of a set of test compounds. Results of the challenge were reported in a subsequent paper in 2009 [22].

The PubChem Dataset A randomly selected subset of 1000 measured solubility values selected from a set of 58,000 values that were experimentally determined using chemiluminescent nitrogen detection (CLND) by the Sanford-Burnham Medical Research Institute and deposited in the PubChem database (AID 1996) [23]. This dataset is composed primarily of screening compounds from the NIH Molecular Libraries initiative and can be considered representative of the types of compounds typically found in early stage drug discovery programs. Values in this dataset were reported with a qualifier “>”, “=”, “<” to indicate whether the values were below, within, or above the limit of detection for the assay. Only values within the limit of detection (designated by “=”) were selected for the subset used in this analysis.

In order to compare predictions with these three datasets, we first need to format the data in a consistent fashion. We begin by formatting all of the data as Log *S*, the log of the molar solubility of the compounds. Data in the PubChem and JCIM datasets were originally reported in µg/ml, so the data was transformed to Log *S* using the formula

$$\text{Log}S = \log_{10}((\text{solubility in } \mu\text{g/ml}) / (1000.0 * \text{MW}))$$

Where log₁₀ is the base 10 logarithm and MW is the molecular weight.

1.3.1 Exploring Datasets

One of the first things to consider in evaluating a new dataset is the range and distribution of values reported. An excellent tool for visualizing data distributions is the boxplot [24, 25]. The “box” at the center of the boxplot shows the range covered by the middle 50% of the data, while the “whiskers” show the maximum and minimum values (discounting the presence of outliers). Outliers in the boxplot are drawn as circles. More information on boxplots can be found on the Wikipedia page [26] and references therein. The anatomy of a boxplot is detailed in Figure 1.1.

Figure 1.2 shows a boxplot of the data distributions for the three solubility datasets mentioned earlier. Numeric summaries of the same data are shown in Table 1.1. Listing 1 provides the R code for loading and annotating the data, as well as generating Figure 1.2 and Table 1.1. The lower whisker and lower hinge in Table 1.1 define the lower extents of the boxplot, while the upper hinge and upper whisker define the upper extents. The interquartile range (IQR) defines the distance between the upper and lower hinges, while the range defines the distance between the upper and lower whiskers.

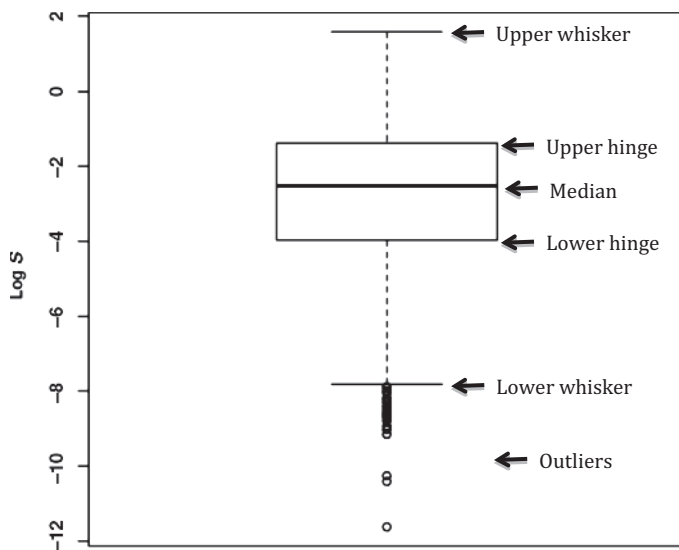


FIGURE 1.1 The anatomy of a boxplot.

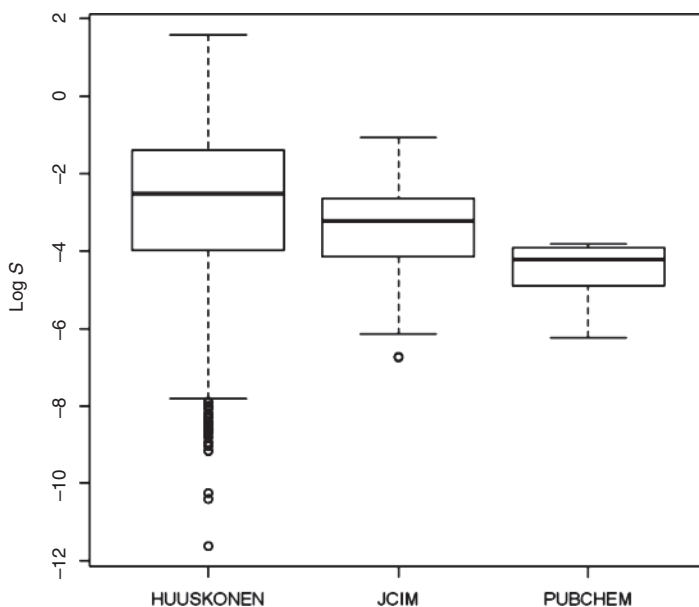


FIGURE 1.2 A boxplot comparison of Log S for the three datasets studied in this chapter.

In examining the datasets, we can see that the Huuskonen dataset spans more than 9 logs, while the JCIM dataset set spans 5 logs, and the PubChem dataset spans a much smaller 2.4 logs. Note that the IQR for the PubChem dataset is only about one log. Measured solubilities in drug discovery programs typically range between 1 and

TABLE 1.1 Boxplot Statistics for the Three Datasets Studied in This Chapter

Dataset	Lower Whisker	Lower Hinge	Median	Upper Hinge	Upper Whisker	Interquartile Range	Range
Huuskonen	-7.82	-3.98	-2.53	-1.39	1.58	2.59	9.4
JCIM	-6.14	-4.15	-3.23	-2.65	-1.06	1.5	5.08
PubChem	-6.23	-4.89	-4.22	-3.91	-3.82	0.98	2.4

100 μM (Log S -6 to -4), so the PubChem dataset can be considered more representative of data that is commonly encountered in drug discovery than the other two datasets. As we will see, the range of data covered by a dataset can have a significant impact on the perceived performance of a model.

1.4 BUILDING PREDICTIVE MODELS

In order to build a predictive model, we need three things:

- Reliable experimental data (see earlier)
- Sets of molecular descriptors [27, 28] that can be derived from chemical structure
- Statistical or machine-learning [29] methods that can be used to model the association between the molecular descriptors and the experimental data.

Models can take a variety of forms, but are typically divided into two categories.

- *Classification models* attempt to determine the membership of an object in a group or category. In the case of aqueous solubility, we might attempt to classify molecules as soluble or insoluble based on a predetermined cutoff like 100 μM . We could classify all molecules with solubility greater than or equal to 100 μM as “soluble,” and all molecules with solubility less than 100 μM as “insoluble.” Of course, this cutoff can vary based on a particular situation or the needs of a project. Classification models can also contain multiple categories. For instance, molecules could be classified as “soluble,” “moderately soluble,” or “insoluble” based on two different cutoffs. There are a few problems with classification models. The first relates to what are called “edge effects.” In setting an arbitrary cutoff, we will invariably run into cases where similar values on either side of the cutoff will be placed into different classes. Let us consider a case where we have a two-class system with a cutoff of 100 μM . A value of 99 μM will be considered insoluble while a value of 101 μM will be considered soluble. Given the experimental error in the measurements, the values are the same, but are put into different categories. The other difficulty with classification models is that they provide limited direction for improving the properties of a compound. Ideally, a scientist on a drug discovery program would like to use a predictive model to

design new compounds with improved properties. While a classification model might be useful for designing a “soluble” compound from an “insoluble” compound, it will not allow one to improve the solubility of an already soluble compound.

- *Regression models* attempt to predict a real value (e.g. Log S) based on existing data and molecular descriptors. While regression models can be quite useful as part of an optimization effort. It is often difficult to create a regression model given data with a limited dynamic range. Many of the ADME assays used in drug discovery programs (e.g. solubility, hERG inhibition, CYP inhibition, metabolic stability, permeation) typically report data within a 2-log range. As we will see in subsequent sections of this chapter, a limited dynamic range makes it extremely difficult to produce reliable regression models.

1.5 EVALUATING THE PERFORMANCE OF PREDICTIVE MODELS

1.5.1 Pearson's r

By far, the most common method for evaluating regression models in the cheminformatics literature is Pearson's product-moment correlation [30], more commonly referred to as Pearson's r , or its square r^2 . Pearson's r can be calculated in a number of ways; one of the most straightforward is shown next.

If we have paired values X and Y (e.g. predicted and corresponding experimental values), then we can calculate Pearson's r as:

$$r = \frac{1}{n-1} \sum_{i=1}^n \left(\frac{X_i - \bar{X}}{S_x} \right) \left(\frac{Y_i - \bar{Y}}{S_y} \right)$$

where $\bar{X}$ and $\bar{Y}$ are the means for X and Y , S_x and S_y are the corresponding standard deviations, and n is the number of data points.

Values of r can vary between -1 and 1 , with 1 indicating a perfect linear correlation, -1 being a perfect inverse linear correlation, and 0 indicating an absence of correlation. The definition of what constitutes a “good” value for r is somewhat subjective and situation dependent, and could easily fill an entire chapter or even a book. As we will see in subsequent sections, the dynamic range of the data being considered can have a dramatic effect on Pearson's r . We will also see that when comparing values of Pearson's r for different models, we must consider the confidence intervals around r .

1.5.2 Kendall's Tau

One of the drawbacks of Pearson's r is that it is sensitive to outliers and to the distribution of the underlying data [31]. More recently, many in the cheminformatics community have begun to follow an example set many years ago by statisticians, and

reporting nonparametric measures of correlation like Kendall's tau [32]. Because these nonparametric methods employ the rank orders of values rather than the values themselves, they are less sensitive to data distribution or outliers. If we have a paired set of values X and Y , we can define Kendall's tau by counting the number of concordant and discordant pairs in the data. Pairs are considered concordant if their rank orders agree

$$x_i > x_j \text{ and } y_i > y_j \quad \text{or} \quad x_i < x_j \text{ and } y_i < y_j.$$

Pairs are considered discordant if their rank orders disagree

$$x_i > x_j \text{ and } y_i < y_j \quad \text{or} \quad x_i < x_j \text{ and } y_i > y_j.$$

Kendall's tau is then evaluated by considering all pairs,

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{(1/2)n(n-1)}$$

where n is the number of pairs.

As we will see in subsequent code listings, Kendall's tau can be easily calculated by using the "Kendall" function in the "Kendall" library in R.

1.5.3 Root-Mean-Square Deviation (RMSD)

In addition to defining the correlation between predicted and experimental values, we need a means of measuring the magnitude of the error in the prediction. The most commonly used error measure in the cheminformatics literature is the root-mean-square deviation (RMSD), which is also known as the RMS error [33]. If we consider paired values X and Y , RMSD can be calculated using the following equation

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^n (x_i - y_i)^2}{n}}$$

where x_i and y_i are paired values (predicted and experimental) and n is the number of data points.

1.6 MOLECULAR DESCRIPTORS

Over the last 20 years, a vast array of molecular descriptors and machine-learning methods have been applied to the problem of building predictive models [27, 28]. A complete treatment of molecular descriptors and machine-learning methods is

beyond the scope of this chapter. Rather than explore the pros and cons of various descriptors and machine-learning methods, we will focus on model-building approaches that can be applied with any descriptors or machine-learning method. In the interest of reproducibility, we will employ a set of molecular descriptors and a machine-learning method that have been widely used and are readily available. The molecular descriptors we will use are the default set calculated by the RDKit cheminformatics programming library [34, 35]. This descriptor set contains a variety of topological indices and encodings for atom environments. As mentioned earlier, our focus here is on the analysis of the results rather than the specifics of the descriptors. The RDKit user manual contains a complete listing of the descriptors as well as the corresponding literature references. Listing 2 provides the Python code for the descriptor calculations used in this chapter.

1.7 BUILDING AND TESTING A RANDOM FOREST MODEL

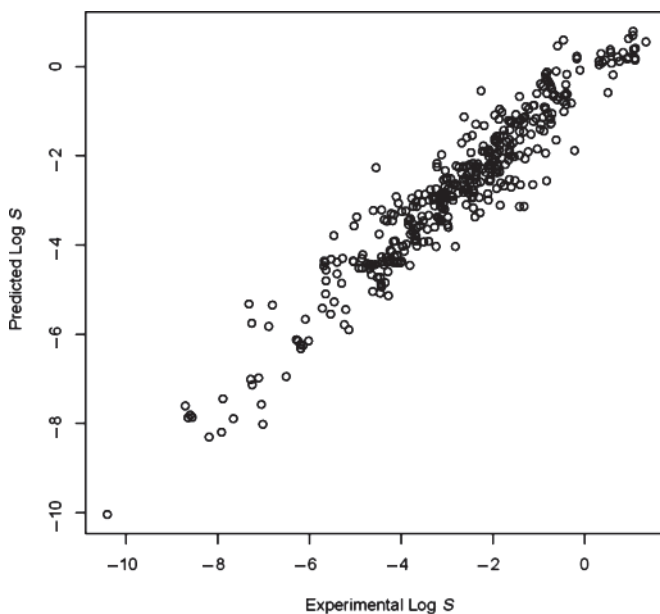
A variety of machine-learning approaches have been applied to develop models relating chemical structure to physical properties and biological activity. These methods range from relatively simple approaches like linear regression to more sophisticated methods like support-vector machines. In this example, we will utilize the random forest machine-learning method that was originally published by Brieman [36]. The random forest method works by constructing an ensemble of decision trees. In practice, the method typically employs hundreds of decision trees and the consensus of multiple trees is used to generate a prediction. The random forest method as implemented in the “randomForest” library for the R statistical software can be used to perform either classification or regression [37, 38]. The choice of classification versus regression is made based on the type of the variable being predicted. If a categorical variable (e.g. “soluble” or “insoluble”) is being predicted, the method will perform classification. If a real valued variable is being predicted, the method will perform regression. In this example, we will generate a regression model. Listing 3 provides R code that demonstrates the construction and testing of a regression model using the random forest method. The steps involved in training and testing the model are listed in the following.

1. Integrate the experimental data and molecular descriptors
2. Divide the data into training and test sets
3. Build a model from the training set
4. Use this model to predict the test set

Note that training and test sets are produced by randomly sampling the dataset. In practice, we would perform this sampling multiple times (typically 50–100) to get a better idea of the overall performance of the model. Some authors have proposed other strategies for constructing training and test sets. There are publications and commercial software packages that advocate clustering the data and selecting one

TABLE 1.2 Statistics for Models Build Based on the Full Huuskonen Dataset and Subset with Log S Between -6 and -3

	Full Huuskonen Set	Realistic Subset
Training set size	891	303
Test set size	383	130
Pearson r	0.95	0.75
Kendall tau	0.81	0.56
RMS error	0.64	0.52

**FIGURE 1.3** A plot of experimental versus predicted Log S for the Huuskonen test set.

training and one test compound from each cluster [39, 40] In the opinion of this chapter's author, this is a terrible idea. By sampling training and test sets in this fashion, one artificially inflates the performance of the method and creates unrealistic expectations from those using the model.

Now that we have built the random forest model, we can evaluate its performance on training and test sets generated from the Huuskonen dataset. Statistics for the test set performance are shown in the first column of Table 1.2. Figure 1.3 shows a plot of the predicted versus experimental Log S for the Huuskonen test set. The plot shows the performance of a model generated using the code in Listing 3. The model was trained using 70% of the compounds in the Huuskonen dataset and tested on the remaining 30%. At first glance, the performance appears quite good. The r for the test set is 0.95 and the RMS error is 0.62. These results are similar to the performance of literature methods on this dataset.

As an aside to those following along with the code in the listings, please note that your results will not exactly match those listed here. There is a stochastic component to the selection of training and test sets, as well as the construction of the random forest models. While your results probably won't match exactly what is reported here, they should be similar.

As we mentioned earlier, the Huuskonen dataset spans a wide range of solubility values. The range observed in this dataset is much larger than what is typically seen in drug discovery projects. Let's take a look at what would happen if we reduced this dataset to only those compounds with Log *S* between -6 and -3 , a more typical range. Listing 4 demonstrates the selection of only those values within this range, and the subsequent construction of a model using a procedure identical to that described earlier for the full Huuskonen dataset. Statistics for model performance are shown in the second column of Table 1.2. In this case, the RMS error of the prediction for the subset is lower at 0.53 than that obtained with the larger set. However, the *r* for the subset is now 0.76, lower than the value for the full Huuskonen set that spans a larger dynamic range.

As we can see, the dynamic range in a dataset can have a large impact on the apparent correlation between experimental and predicted activity. The literature is replete with examples of what appear to be impressive correlations on datasets that span an unrealistically high range. Authors who generate predictive models for protein–ligand binding affinity often use datasets that span up to 12 orders of magnitude [8]. The reality is that the binding affinity of compounds typically encountered in drug discovery programs may span 5–6 orders of magnitude. When data within this typical range is considered, these apparent correlations decrease dramatically. In practice, the utility of models for predicting–binding affinity is extremely limited.

Unfortunately, many cheminformatics practitioners have become enamored with *r* values, and linear plots of model performance. This has caused them to, consciously or unconsciously, choose datasets that often provide an unrealistic view of model performance. When building a predictive model, one should consider the dynamic range of the data being used to build the model and how this range compares with the range of the data to be predicted.

1.8 EXPERIMENTAL ERROR AND MODEL PERFORMANCE

One critical factor to keep in mind when building and evaluating predictive models is that every experimental data point has an error associated with it. For example, if we measure the Log *S* of a compound as -6 and that data point has an error of 0.3 log units, the actual value could be anywhere between -6.3 and -5.7 . In a 2009 paper, Brown and coworkers [41] examined the relationship between experimental error and model performance. They carried out a series of theoretical experiments where Gaussian distributed random values were added to data to simulate experimental errors. The authors then calculated the correlation between the measured values and the same values with this simulated error. This correlation can be thought of as the maximum correlation possible given the error in the measurement. As we saw earlier,

TABLE 1.3 Maximum Possible Correlations for the Three Datasets Studied When Experimental Error Is 0.3, 0.5, and 1.0 Log

Dataset	Max@0.3	Max@0.5	Max@1.0
Huuskonen	0.98	0.94	0.81
JCIM	0.95	0.86	0.60
PubChem	0.80	0.60	0.27

this maximum correlation is a function not only of the error but also of the number of datapoints and the dynamic range of the data. The authors did a survey of predictive models in the molecular modeling and cheminformatics literature and found that correlations reported in 8 of 16 papers examined exceeded what could be expected based on experimental error alone.

Listing 5 shows how we can use R to add a random error to each data point and examine the correlation between experimental data and the data with added error. In Listing 5, we calculate the mean error for 1000 such simulations. Table 1.3 shows the maximum possible correlation for each of the three solubility datasets we have been examining when experimental errors of 0.3, 0.5, and 1.0 log are considered. Note that the error has much more of an impact on a dataset like the PubChem set, which has a limited dynamic range. If our experimental error is 0.5 log, then the best r^2 we can hope to achieve is 0.60. Also note that when the error is 1.0 log, almost half the dynamic range of the dataset, it is impossible to obtain any sort of meaningful correlation. Because the dynamic range of the Huuskonen dataset is (some might say unrealistically) large, the impact of error on the correlation is much less significant.

When building any sort of predictive model, it is important to carry out the type of analysis outlined earlier in order to account for experimental error. Any predictive models that perform better than the theoretical limit should be viewed with extreme skepticism.

1.9 MODEL APPLICABILITY

Although we would like them to be, predictive models are not universal. Regardless of the statistical or machine-learning method used, models built from molecular descriptors tend to do a reasonable job of predicting the activity of molecules similar to those in the training set. However, these models often perform poorly on molecules that bear little resemblance to those in the training set. This relationship between chemical similarity and model performance is detailed in a 2004 paper by Sheridan and coworkers [42]. In this paper, the authors examined 20 different predictive models and found that the best predictors of model performance were the similarity to the nearest neighbor in the training set and the number of neighbors in the training set. Neighbors are defined as molecules that are similar within a particular threshold (typically a Tanimoto similarity of 0.7–0.85). Molecules that were similar to those in the training set were predicted well, while those that were not similar were predicted poorly.

We can use the datasets employed so far to examine the relationship between similarity to the training set molecules and model performance. As before, we will

build a random forest model using 70% of the Huuskonen dataset (we will refer to this as our training set) and use this model to predict the solubilities of

- Remaining 30% of the Huuskonen dataset
- JCIM dataset
- PubChem dataset

For each of these datasets, we will also calculate the similarity of each molecule to every molecule in the training set. We will report the maximum similarity to any training set molecule as an indicator the molecule's similarity to the training set.

There are many ways to calculate molecular similarity, and a complete discussion of the topic is beyond the scope of this chapter. The interested reader is urged to consult some of the reviews referenced here [43–48] For purposes of illustration, we will calculate molecular similarity using 2D pharmacophore fingerprints as implemented in the RDKit library. Listing 6 provides the code for performing the similarity comparisons.

Once we have calculated the maximum similarity of each training set structure to each test set structure, we can use a boxplot to compare the similarities of our test sets. Listing 7 provides the R code for reading the similarity data, assigning labels to the datasets and plotting boxplots.

An examination of Figure 1.4 and Table 1.4 can give us some idea of the expected performance of a model built from the Huuskonen training set on our three test sets.

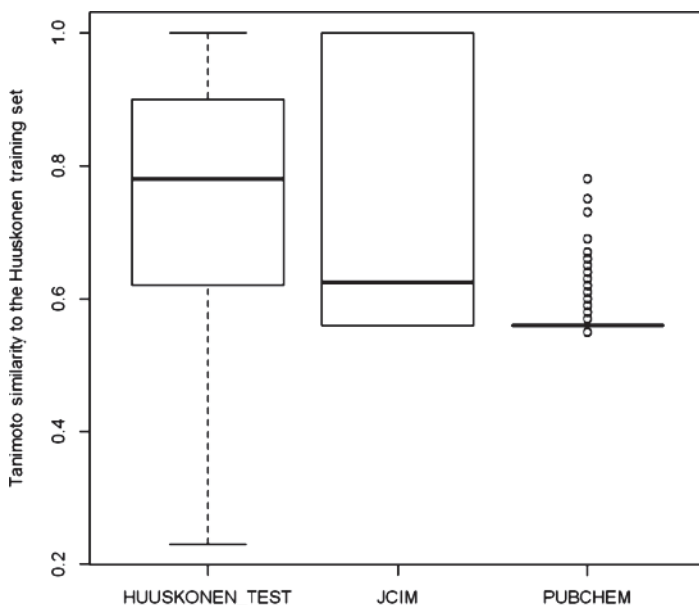


FIGURE 1.4 Tanimoto similarity of each of the three test sets to the Huuskonen training set.

TABLE 1.4 Similarity of Each Test Set to the Huuskonen Training Set

Dataset	Mean	Median
Huuskonen_Test	0.76	0.78
JCIM	0.74	0.62
PubChem	0.56	0.56

TABLE 1.5 Model Performance for the Three Test Sets

Dataset	R^2	Kendall	RMS Error
Huuskonen_Test	0.92	0.82	0.58
JCIM	0.58	0.59	0.83
PubChem	0.11	0.22	1.12

- The Huuskonen test set is similar to the training set. The median similarity for the training set is 0.78 and the mean is 0.76. By most standards, one would assume that approximately half of the molecules in the Huuskonen test set are similar to molecules in the training set. We would expect the performance of the model on this dataset to be reasonably good.
- The JCIM set is less similar to the training set. The median is 0.62 and the mean is 0.74. The boxplot and higher mean indicate that there are a number of compounds in this set that are similar to the training set. The absence of whiskers in the boxplot indicates that the similarities are more narrowly distributed. We would expect moderate performance from this dataset.
- The PubChem dataset appears to be very different from the training set. Note that the boxplot is almost flat, with a few outliers drawn as circles. The mean and median similarities to the training set are both 0.56. Similarity values in this range are what we would tend to expect from pairs of random compounds. We would expect extremely poor performance from this dataset.

Now that we have these predictions of performance in hand, we can generate a model based on our training set, use it to predict the test sets, and see if the model performance meets our expectations. Listing 8 shows how we can use R to build a model from our training set and test on each of the three test sets. Table 1.5 shows the r^2 , Kendall tau, and RMS error for the prediction of each of the three datasets earlier. The results are in line with our expectations based on the similarity of test set compounds to the training set. As was predicted earlier, the Huuskonen test set gives a reasonably good prediction with an r^2 of 0.92 and an RMS error of 0.58. Again, as expected based on similarity to the training set, the performance of the model on the JCIM dataset can be considered moderate. Finally, as expected based on similarity to the training set, we are unable to obtain reasonable predictions for the PubChem dataset.

When evaluating the potential performance of a model on a new dataset, it is important to perform this type of analysis to gauge whether one can expect useful

predictions to come from a model. This is particularly important when working with collaborators who may have a limited understanding of computational methods. Many pharmaceutical companies have set up automated prediction methods that allow scientists to simply sketch a molecule and obtain a prediction. While tools of this sort can make computation available to a wider audience, they can also mislead a naïve user. Any predictive tool, whether designed for an expert user or a novice, should provide some indication of whether the molecule being predicted falls within the applicability domain of the model.

1.10 COMPARING PREDICTIVE MODELS

In this final section, we will consider the comparison of two predictive models. The cheminformatics literature is replete with papers comparing predictive models. When developing a new method, it is always important to examine how the method compares with the current state of the art. However, when making comparisons, one must remember that correlations have an associated error. This error is a function of both the correlation coefficient and the number of data points used to obtain the correlation coefficient. When comparing correlation coefficients, we must not only consider the value of the correlation coefficient, but also the confidence intervals around the correlation coefficient. When we have a larger number of data points or a higher correlation coefficient, we are more confident in the correlation and our confidence interval is relatively narrow. When we have a smaller number of data points or our correlation coefficient is lower, the confidence interval around the correlation is larger. If the confidence intervals of two correlations overlap, we cannot claim that one predictive model is superior to another.

While the calculation of confidence intervals for a correlation is straightforward, it is rarely used in the cheminformatics literature. As such, we will provide a brief review of the method for calculating a confidence interval on a Pearson r . Since values of Pearson's r cannot exceed 1, its distribution is not normal. The distribution is closer to normal for lower values of r and becomes more skewed as r approaches 1. In order to calculate a confidence interval, values of r must be converted to Fisher's z' distribution using Equation 1.10.1.

$$z' = .5[\ln(1+r) - \ln(1-r)] \quad (1.10.1)$$

An r of 0.7 would then equate to a z' of 0.87. The z' distribution is normal and its standard deviation can be calculated as

$$\sigma_{z'} = \frac{1}{\sqrt{N-3}} \quad (1.10.2)$$

where N is the number of pairs used to calculate r . Confidence intervals can then be calculated as

$$z' \pm z\sigma_{z'} \quad (1.10.3)$$

For this example, let us assume we are dealing with a dataset where we are predicting the solubility of 23 molecules. According to standard tables of z values, we should use a value of 2.08 for a t test at 95% confidence for a two-tailed distribution with a sample size of 23. The value of $\sigma_{z'}$ would then be 0.23. For a Pearson r of 0.7, the confidence intervals on z' would be

$$\text{Lower limit} = 0.87 - (2.08 \times 0.23) = 0.39$$

$$\text{Upper limit} = 0.87 + (2.08 \times 0.23) = 1.35$$

Transforming the values from a z' distribution back to an r distribution, we end up with a confidence limit on r of

$$0.37 \leq r \leq 0.87$$

The confidence intervals for a Person's r , Kendall's tau, or Spearman's rho can be easily calculated using the function `cor.test` in R.

We will now consider the case of two predictive models, Model A and Model B, for aqueous solubility. We will use R to compare the performance of these models when tested on 25, 50, and 100 compounds. Listing 9 provides an example of how this comparison can be performed in R. In this listing, we first calculate the Pearson r and the upper and lower 95% confidence intervals for the Pearson r . Table 1.6 and Figure 1.5 show the correlations and associated bar plots. The bar plots show the value of Pearson r for each subset and the associated “whiskers” show the upper and lower limits of the 95% confidence interval.

From Figure 1.5 we can see that, for the subset of 25 compounds, the 95% confidence intervals overlap. When the confidence intervals overlap like this, we cannot say that one correlation is superior to the other. For the subset of 50 compounds, there is a very small difference between the upper bound of the 95% confidence interval for Method A (0.91) and the lower bound of the confidence interval for Method B (0.92). While there is some separation, we may still have doubts as to whether one method outperforms the other. The bars on the far right in Figure 1.5 show the correlations for the subset of 100 compounds and the associated

TABLE 1.6 Pearson r , Upper and Lower Bound for 95% Confidence Intervals for Regression Models

Dataset	Method	r	Confidence Interval	
			Lower Bound	Upper Bound
subset25	Method_A	0.80	0.60	0.91
subset25	Method_B	0.93	0.84	0.97
subset50	Method_A	0.85	0.75	0.91
subset50	Method_B	0.96	0.92	0.97
subset100	Method_A	0.81	0.73	0.87
subset100	Method_B	0.94	0.91	0.96

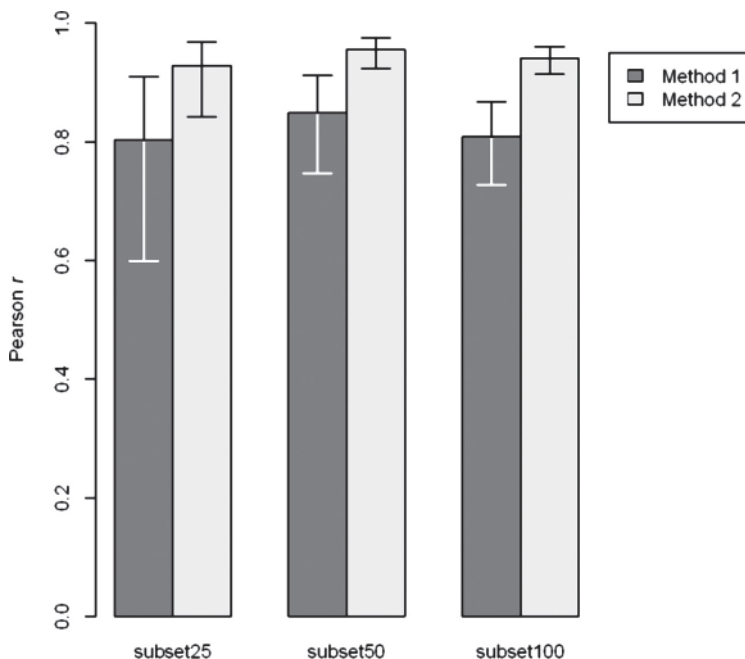


FIGURE 1.5 Comparing the errors associated with regression for two predictive models evaluated on three different datasets.

confidence intervals. In this case, there is a clear separation between the confidence intervals. As such, we can have some confidence that there is a difference between the correlation coefficients.

1.11 CONCLUSION

As mentioned in the introduction, cheminformatics and predictive modeling have become an integral part of modern drug discovery. The advent of “easy to use” software has opened the world of predictive modeling to a much wider audience. While it has become easier to generate and apply predictive models, it is still essential that we remain vigilant in evaluating both the statistical and chemical validity of the models. This chapter has presented a few factors that must be considered when evaluating the performance of predictive models:

- Dynamic range of the data
- Experimental error
- Applicability domain of the model
- Confidence intervals on correlation coefficients

It is hoped that the scripts presented in this chapter will provide the basis for a toolset that cheminformatics practitioners will use to generate and critically evaluate predictive models.

Cheminformatics is a relatively new discipline that encompasses a number of different fields. Many people come to cheminformatics from other fields, or use cheminformatic methods as an adjunct to experimental work. Standards for how cheminformatics professionals should be trained are still emerging. Hopefully as the field matures, practitioners will begin to appreciate the importance of a firm grounding in Statistics. Without an appreciation for the statistical foundations in our methods, it is difficult to see how our field will progress. This paper should not be considered a comprehensive guide to model building or evaluation. The objective here was just to point out a few commonly encountered pitfalls. The interested reader is urged to consult any of a number of excellent Biostatistics or data analysis texts [31, 49]. Another excellent additional source of information on model evaluation and comparison is the recent work of Anthony Nicholls [50–52].

REFERENCES

1. Gleeson MP, Hersey A, Montanari D, et al. Probing the links between in vitro potency, ADMET and physicochemical parameters. *Nat Rev Drug Discov* 2011;10:197–208.
2. Delaney JS. Predicting aqueous solubility from structure. *Drug Discov Today* 2005;10:289–295.
3. Wang J, Hou T. Recent advances on aqueous solubility prediction. *Comb Chem High Throughput Screen* 2011;14:328–338.
4. Hughes LD, Palmer DS, Nigsch F, et al. Why are some properties more difficult to predict than others? A study of QSPR models of solubility, melting point, and log P. *J Chem Info Model* 2008;48:220–232.
5. Hudelson MG, Ketkar NS, Holder LB, et al. High confidence predictions of drug-drug interactions: Predicting affinities for cytochrome P450 2C9 with multiple computational methods. *J Med Chem* 2008;51:648–654.
6. Aronov AM. Predictive in silico modeling for hERG channel blockers. *Drug Discov Today* 2005;10:149–155.
7. Song M, Clark M. Development and evaluation of an in silico model for hERG binding. *J Chem Info Model* 2006;46:392–400.
8. Smith RD, Dunbar JB, Jr, Ung PM, et al. CSAR benchmark exercise of 2010: Combined evaluation across all submitted scoring functions. *J Chem Info Model* 2011;51:2115–2131.
9. Ceder V. *The Quick Python Book*. 2nd ed. Greenwich: Manning Publications; 2010. p 400.
10. Beazley DM. *Python Essential Reference*. 4th ed. Addison-Wesley Professional; 2009. p 717.
11. Adler J. *R in a Nutshell: A Desktop Quick Reference*. O'Reilly Media; 2010. p 636.
12. Chambers J. *Software for Data Analysis: Programming with R* (Statistics and Computing). New York: Springer; 2010. p 514.

13. Lipinski C. Drug-like properties and the causes of poor solubility and poor permeability. *J Pharmacol Toxicol Methods* 2000;44:235–249.
14. Hou TJ, Xia K, Zhang W, et al. ADME evaluation in drug discovery. 4. Prediction of aqueous solubility based on atom contribution approach. *J Chem Info Model* 2004;44:266–275.
15. Huuskonen J. Estimation of aqueous solubility for a diverse set of organic compounds based on molecular topology. *J Chem Info Model* 2000;40:773–777.
16. Huuskonen J, Salo M, Taskinen J. Aqueous solubility prediction of drugs based on molecular topology and neural network modeling. *J Chem Info Model* 1998;38:450–456.
17. Tetko IV, Tanchuk VY, Kasheva TN, et al. Estimation of aqueous solubility of chemical compounds using E-state indices. *J Chem Info Model* 2001;41:1488–1493.
18. Myrdal P, Ward GH, Dannenfelser RM, et al. AQUAFAC 1: Aqueous functional group activity coefficients; application to hydrocarbons. *Chemosphere* 1992;24:1047–1061.
19. <http://www.pharmacy.arizona.edu/outreach/aquasol/index.html>. Accessed 2013 May 14.
20. <http://www.srcinc.com/what-we-do/product.aspx?id=133>. Accessed 2013 May 14.
21. Llinàs A, Glen RC, Goodman JM. Solubility challenge: Can you predict solubilities of 32 molecules using a database of 100 reliable measurements? *J Chem Info Model* 2008;48:1289–1303.
22. Hopfinger AJ, Esposito EX, Llinàs A, et al. Findings of the challenge to predict aqueous solubility. *J Chem Info Model* 2009;49:1–5.
23. Guha R, Dexheimer TS, Kestranek AN, et al. Exploratory analysis of kinetic solubility measurements of a small molecule library. *Bioorg Med Chem* 2011;19:4127–4134.
24. <http://flowingdata.com/2008/02/15/how-to-read-and-use-a-box-and-whisker-plot/>. Accessed 2013 May 14.
25. <http://flowingdata.com/2012/05/15/how-to-visualize-and-compare-distributions/>. Accessed 2013 May 14.
26. http://en.wikipedia.org/wiki/Box_plot. Accessed 2013 May 14.
27. Todeschini R, Consonni V. *Molecular Descriptors for Chemoinformatics* (Methods and Principles in Medicinal Chemistry). Weinheim: Wiley-VCH; 2009. p 1257.
28. Karelson M. *Molecular Descriptors in QSAR/QSPR*. New York: Wiley-Interscience; 2000. p 448.
29. Goldman BB, Walters WP. Machine learning in computational chemistry. *Annu Rep Comput Chem* 2006;2:127–140.
30. http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient. Accessed 2013 May 14.
31. Glantz, S. *Primer of Biostatistics*. New York: McGraw-Hill Medical; 2011. p 320.
32. http://en.wikipedia.org/wiki/Kendall_tau_rank_correlation_coefficient. Accessed 2013 May 14.
33. http://en.wikipedia.org/wiki/Root-mean-square_deviation. Accessed 2013 May 14.
34. Landrum G, Lewis R, Palmer A, et al. Making sure there's a “give” associated with the “take”: Producing and using open-source software in big pharma. *J Cheminform* 2011;3:1–1.
35. <http://www.rdkit.org/>. Accessed 2013 May 14.
36. Breiman L. Random forests. *Mach Learn* 2001;45:5–32.

37. Liaw A, Wiener M. Classification and regression by random forest. *R News* 2002;2:18–22.
38. Svetnik V, Liaw A, Tong C, et al. Random forest: A classification and regression tool for compound classification and QSAR modeling. *J Chem Info Comput Sci* 2003;43:1947–1958.
39. Yan A, Wang Z, Cai Z. Prediction of human intestinal absorption by GA feature selection and support vector machine regression. *Int J Mol Sci* 2008;9:1961–1976.
40. Yan A, Gasteiger J. Prediction of aqueous solubility of organic compounds based on a 3D structure representation. *J Chem Info Model* 2003;43:429–434.
41. Brown SP, Muchmore SW, Hajduk PJ. Healthy skepticism: Assessing realistic model performance. *Drug Discov Today* 2009;14:420–427.
42. Sheridan RP, Feuston BP, Maiorov VN, et al. Similarity to molecules in the training set is a good discriminator for prediction accuracy in QSAR. *J Chem Info Comput Sci* 2004;44:1912–1928.
43. Grant J, Haigh J, Pickup B, et al. Lingos, finite state machines, and fast similarity searching. *J Chem Inf Model* 2006; 46:1912–1918.
44. Hert J, Willett P, Wilton DJ, et al. New methods for ligand-based virtual screening: Use of data fusion and machine learning to enhance the effectiveness of similarity searching. *J Chem Info Model* 2006;46:462–470.
45. Moffat K, Gillet VJ, Whittle M, et al. A comparison of field-based similarity searching methods: CatShape, FBSS, and ROCS. *J Chem Info Model* 2008;48:719–729.
46. Muchmore SW, Debe DA, Metz JT, et al. Application of belief theory to similarity data fusion for use in analog searching and lead hopping. *J Chem Info Model* 2008;48:941–948.
47. Bender A, Jenkins JL, Scheiber J, et al. How similar are similarity searching methods? A principal component analysis of molecular descriptor space. *J Chem Info Model* 2009;49:108–119.
48. Nicholls A, McGaughey GB, Sheridan RP, et al. Molecular shape and medicinal chemistry: A perspective. *J Med Chem* 2010;53:3862.
49. Pearson R. *Exploring Data in Engineering, the Sciences, and Medicine*. New York: Oxford University Press; 2011. p 792.
50. Nicholls A. What do we know and when do we know it? *J Comput Aided Mol Des* 2008;22:239–255.
51. Jain AN, Nicholls A. Recommendations for evaluation of computational methods. *J Comput Aided Mol Des* 2008;22:133–139.
52. Hawkins PCD, Warren GL, Skillman AG, et al. How to do an evaluation: Pitfalls and traps. *J Comput Aided Mol Des* 2008;22:179–190.

SOURCE CODE LISTINGS

install_libraries.R

```
install.packages(c("randomForest", "Kendall", "plyr", "ggplots"), dependencies = TRUE)
```

listing_1.R

```
# load the solubility data
hus = read.table("huuskonen.sol",header = T,row.names = 1)
jcim_train = read.table("jcim_train.sol",header = T,row.
names = 1)
pub = read.table("pubchem_sample.sol",header = T,row.
names = 1)

# label the data
hus = labelData(hus,"HUUSKONEN")
jcim_train = labelData(jcim_train,"JCIM")
pub = labelData(pub,"PUBCHEM")

# combine the 3 datasets into 1 dataframe
allData = rbind(hus,jcim_train,pub)

# define an order for the x-axis in the boxplot
allData$DATASET = factor(allData$DATASET,levels = c("HUU
SKONEN","JCIM","PUBCHEM"))

# display a boxplot of LogS vs Dataset and a corresponding
table of boxplot statistics
print(summaryTable(allData))
```

listing_2.py

```
#!/usr/bin/env python

# Listing 2
# calculate molecular descriptors using the RDKit library

import sys,string
from rdkit import Chem
from rdkit.Chem import Descriptors

if len(sys.argv) != 3:
    print >> sys.stderr,"usage %s infile.smi outfile" %
(sys.argv[0])
    sys.exit(0)

# setup a molecule supplier for the input molecules
suppl = Chem.SmilesMolSupplier(sys.argv[1],titleLine =
False)
# open the output file
ofs = open(sys.argv[2],"w")
```

```

# create a list of descriptor names for the header
nameList = ["Name"] + [x[0] for x in Descriptors.
descList[3:]]
print >> ofs,string.join(nameList)
# read the input molecules
for idx,mol in enumerate(suppl):
    print >> sys.stderr,"\r",idx + 1,
    sys.stderr.flush()
    print >> ofs,mol.GetProp("_Name"),
    # write out the descriptors
    for name,desc in Descriptors.descList[3:]:
        val = desc(mol)
        print >> ofs,"%0.2f" % (val),
    print >> ofs
print >> sys.stderr,"\r",idx + 1

```

listing_3.R

```

# Listing 3
# train and test a random forest model based on the
Huuskonen dataset

# load the required libraries
library(randomForest)
library(Kendall)

# calculate the root mean squared error
rmsError<-function(a,b){
  sqrt(sum((a-b)**2)/length(a))
}

# split a dataset into training and test sets
splitTrainTest<-function(dataSet,trainingFraction=0.7){
  idxList = sample(1:nrow(dataSet))
  numTrain = floor(trainingFraction * nrow(dataSet))
  trainIdx = idxList[1:numTrain]
  testIdx = idxList[(numTrain + 1):length(idxList)]
  list("train" = trainIdx,"test" = testIdx)
}

# merge descriptors and experimental data into a single
dataframe
mergeData < -function(descriptors,logS){
  mergedData = merge(logS,descriptors,by = 0)
  mergedData = mergedData[,-c(1,3)]
  mergedData
}

```

```

# use descriptors and experimental data to train and test
a random forest model
predictRf <- function(mergedData, trainingFraction = 0.7)
{
  ttSplit = splitTrainTest(mergedData, trainingFraction)
  rf = randomForest(LOGS ~ ., mergedData[ttSplit$train,])
  pred = predict(rf, mergedData[ttSplit$test,])
  list("train" = ttSplit$train, "test" = ttSplit$test,
"pred" = pred, "exper" = mergedData[ttSplit$test,]$LOGS,
"model" = rf)
}

# read the data
desc = read.table("huuskonen.rdkit", header = T, row.names
= 1)
logS = read.table("huuskonen.sol", header = T, row.names = 1)
# merge descriptors and solubility data
mergedData = mergeData(desc, logS)
# build and test the random forest model
res = predictRf(mergedData)
# plot the test set results
plot(res$exper, res$pred, xlab = "Experimental LogS", ylab
= "Predicted LogS")
# output results
cat(sprintf("      Train = %d\n", length(res$train)))
cat(sprintf("      Test = %d\n", length(res$test)))
cat(sprintf(" Pearsonr = %.2f\n", cor(res$pred, res$exper)))
cat(sprintf(" Kendall      tau      =
%.2f\n", Kendall(res$pred, res$exper)$tau))
cat(sprintf("      RMS      error      =
%.2f\n", rmsError(res$pred, res$exper)))

```

listing_4.R

```

# Listing 4
# train and test a random forest model based on a subset
of the Huuskonen dataset
# not that this listing is just a variation on Listing 3

# load the required libraries
library(randomForest)
library(Kendall)

# calculate the root mean squared error
rmsError<-function(a,b){
  sqrt(sum((a-b)**2)/length(a))
}

```

```

# split a dataset into training and test sets
splitTrainTest <- function(dataSet, trainingFraction =
0.7){
  idxList = sample(1:nrow(dataSet))
  numTrain = floor(trainingFraction * nrow(dataSet))
  trainIdx = idxList[1:numTrain]
  testIdx = idxList[(numTrain + 1):length(idxList)]
  list("train" = trainIdx, "test" = testIdx)
}

# merge descriptors and experimental data into a single
dataframe
mergeData <- function(descriptors, logS){
  mergedData = merge(logS, descriptors, by = 0)
  mergedData = mergedData[, -c(1,3)]
  mergedData
}

# use descriptors and experimental data to train and test
a random forest model
predictRf <- function(mergedData, trainingFraction = 0.7){
  ttSplit = splitTrainTest(mergedData, trainingFraction)
  rf = randomForest(LOGS ~ ., mergedData[ttSplit$train,])
  pred = predict(rf, mergedData[ttSplit$test,])
  list("train" = ttSplit$train, "test" = ttSplit$test,
"pred" = pred, "exper" = mergedData[ttSplit$test,]$LOGS, "
model" = rf)
}

# read the data
desc = read.table("huuskonen.rdkit", header = T, row.names
= 1)
logS = read.table("huuskonen.sol", header = T, row.names =
1)
# merge descriptors and solubility data
mergedData = mergeData(desc, logS)
# create a subset with LogS between -6 and -3
realisticSubset = mergedData[mergedData$LOGS > = -6 &
mergedData$LOGS < = -3,]
res = predictRf(realisticSubset)
# plot the test set results
plot(res$exper, res$pred, xlab = "Experimental LogS", ylab
= "Predicted LogS")
# output results
cat(sprintf("      Train = %d\n", length(res$train)))

```

```

cat(sprintf("      Test = %d\n",length(res$test)))
cat(sprintf(" Pearsonr=%.2f\n",cor(res$pred,res$exper)))
cat(sprintf("Kendall      tau      =
%.2f\n",Kendall(res$pred,res$exper)$tau))
cat(sprintf("      RMS      error      =
%.2f\n",rmsError(res$pred,res$exper)))

```

listing_5.R

```

# Listing 5
# add simulated error to experimental data to examine the
# impact of error on correlations

# calculate the maximum correlation that can be achieved
# give a measure of experimental error
# values - the experimental data
# error - the standard deviation of the error to be added,
# mean is 0
# repeats - the number of times to repeat the simulation
maxError <- function(values,error = 0.3,repeats = 1000){
  correlationList = c()
  for (i in 1:repeats){
    errorList = rnorm(n = length(values),mean = 0,sd =
error)
    correlationList[i] = cor(values,values + errorList)**2
  }
  mean(correlationList)
}

# load the solubility data
hus = read.table("huuskonen.sol",header = T,row.names =
1)
jcim_train = read.table("jcim_train.sol",header = T,row.
names = 1)
pub = read.table("pubchem_sample.sol",header = T,row.
names = 1)

# write the output table
cat(sprintf("%-20s      %8      s      %8      s      %8
s\n", "Dataset", "Max@0.3", "Max@0.5", "Max@1.0"))
for (data in list(c(hus, "Huuskonen"),c(jcim_
train, "JCIM"),c(pub, "PubChem"))){
  name = data[2]
  maxCorrelationList = c()
  i = 1

```

```

for (err in c(0.3,0.5,1.)){
  maxCorrelationList[i] = maxError(data[1]$LOGS,err)
  i = i + 1
}
cat(sprintf("%-20s      %8.2f      %8.2f      %8.2f\n",name,
maxCorrelationList[1],maxCorrelationList[2],maxCorrelationList[3]))

```

listing_6.py

```

#!/usr/bin/env python

# Listing 6
# calculate similarity between pairs of SMILES files and
# report
# the most similar training set molecule for each test set
# molecule

import sys,string
from rdkit import Chem
from rdkit import DataStructs
from rdkit.Chem.Fingerprints import FingerprintMols

# build a list of fingerprints from an input file
def buildFingerprintList(fileName):
    suppl = Chem.SmilesMolSupplier(fileName,titleLine =
    False)
    fpList = []
    for idx,mol in enumerate(suppl):
        print >> sys.stderr,"\rGenerating fingerprints for %s
" % (fileName),idx + 1,
        sys.stderr.flush()
    fpList.append([mol.GetProp("_Name"),FingerprintMols.
    FingerprintMol(mol)])
    print >> sys.stderr,"\rGenerating fingerprints for %s "
    % (fileName),idx + 1
    return fpList

# for each molecule in queryFpList, find and report the
# most similar molecule in refFpList
def findMostSimilar(refFpList,queryFpList):
    outList = []
    for idx,[name,fp] in enumerate(queryFpList):
        print >> sys.stderr,"\rCalculating similarity ",idx,
        sys.stderr.flush()

```

```

    simList = [[x[0],DataStructs.FingerprintSimilarity
(fp,x[1])]] for x in refFpList]
    simList.sort(lambda a,b: cmp(b[1],a[1]))
    outList.append([name,simList[0]])
    print >> sys.stderr,"\rCalculating similarity ",idx + 1
    return outList

# setup training and test sets
trainingFiles = ["huuskonen_train.smi"]
testFiles = ["huuskonen_test.smi","jcm.smi","pubchem.
smi"]
trainDict = {}
testDict = {}

# open the output file
ofs = open("similarity.txt","w")

# generate fingerprints
for fileName in trainingFiles:
    trainDict[fileName] = buildFingerprintList(fileName)
for fileName in testFiles:
    testDict[fileName] = buildFingerprintList(fileName)

# write the output
print >> ofs,"Query Reference Similarity Dataset"
for trainFileName,trainFpList in trainDict.iteritems():
    for testFileName,testFpList in testDict.iteritems():
        print >> sys.stderr,"Processing %s" % testFileName
        simList = findMostSimilar(trainFpList,testFpList)
        for query,[ref,sim] in simList:
            print >> ofs,query,ref,"%0.2f" % (sim),testFileName.
split(".") [0].upper()

print >> sys.stderr,"Results have been written to simi-
larity.txt"

```

listing_7.R

```

# Listing 7
# load data and display box plots to compare distributions

# load the required library
library(plyr)

# read the data
d = read.table("similarity.txt",header = T)

```

```
# generate the boxplot
boxplot(Similarity ~ Dataset,d,ylab = "Tanimoto Similarity
to the Huuskonen Training Set")
# generate a table of mean and median for each dataset
res = ddply(d, c("Dataset"), function(x)c(mean(x$Similarity),median(x$Similarity)))
# format the results into a table
res = data.frame(res)
names(res) = c("Dataset","Mean","Median")
res$Mean = round(res$Mean, digits = 2)
res$Median = round(res$Median, digits = 2)
# print the table
print(res)
```

listing_8.R

```
# Listing 8
# predict activities of 3 test sets for a training set

library(randomForest)
library(Kendall)

# calculate the root mean squared error
rmsError <- function(a,b){
  sqrt(sum((a-b)**2)/length(a))
}

# merge descriptors and experimental data into a single
data frame
mergeData <- function(descriptors,logS){
  mergedData = merge(logS,descriptors,by = 0)
  mergedData = mergedData[,-c(1,3)]
  mergedData
}

# build a predictive model from descriptors and experi-
mental data
buildModel <- function(descriptors,logS){
  mergedData = mergeData(descriptors,logS)
  rf = randomForest(LOGS ~ .,mergedData)
  rf
}

# use a model to predict test set activity
predictTestSet <- function(model,name,suffix){
```

```

descriptors = read.table(paste(name, suffix, sep =
" "), header = T, row.names = 1)
logS = read.table(paste(name, ".sol", sep = " "), header
= T, row.names = 1)
mergedData = mergeData(descriptors, logS)
pred = predict(model, mergedData)
r2 = cor(mergedData$LOGS, pred)**2
err = rmsError(mergedData$LOGS, pred)
kt = Kendall(mergedData$LOGS, pred)$tau
output = list("name" = name, "r2" = r2, "rms_error" =
err, "exper" = mergedData$LOGS, "pred" = pred, "kendall" =
kt)
output
}

```

```
descriptorSuffix = ".rdkit"
```

```

# read the training set and build a random forest model
huuskonenTrainDes = read.table(paste("huuskonen_train",
descriptorSuffix, sep = " "), header = T, row.names = 1)
huuskonenTrainSol = read.table("huuskonen_train.
sol", header = T, row.names = 1)
rf = buildModel(huuskonenTrainDes, huuskonenTrainSol)

# read the test sets and predict activity
testSets = c("huuskonen_test", "jcim", "pubchem")
cat(sprintf("%-15s %8s %8s %8s\n", "DataSet", "R**2", "Kendall", "RMS Error"))
for (fileName in testSets){
  res = predictTestSet(rf, fileName, descriptorSuffix)
  cat(sprintf("%-15s %8.2f %8.2f %8.2f\n", res$name, res$r2, res$kendall, res$rms_error))
}

```

listing_9.R

```

# Listing 9
# calculate errors for regression and plot Pearson r with
associated error bars

library(gplots)

# read the data
rc = read.table("compare_regression.txt", header = T, row.
names = 1)

```

30 WHAT ARE OUR MODELS REALLY TELLING US?

```
# generate subsets of 25, 50, and 100
subset25 = rc[0:25,]
subset50 = rc[0:50,]
subset100 = rc[0:100,]

# createList (actually vectors) to loop over
nameList = c("subset25","subset50","subset100")
dataList = list(subset25,subset50,subset100)

# print the header
header = sprintf("%-15 s %-15 s %8 s %8 s %8 s %8 s",
"Dataset","Method","R","CI LB","CI UB")
cat(header,"\n")

# loop over the 3 subsets
res = c()
for (i in 1:length(nameList)){
  name = nameList[i]
  data = data.frame(dataList[i])
  # calculate the correlation coefficients and confidence
intervals
  resA = cor.test(data$LOGS,data$Method_A)
  resB = cor.test(data$LOGS,data$Method_B)
  # convert from a list to a vector of numbers
  outA = (as.numeric(c(resA$estimate,resA$conf.int)))
  # print the output
  strA = sprintf("%-15 s %-15 s %8.2f %8.2f %8.2f",name,
"Method_A",resA$estimate,resA$conf.int[1],resA$conf.
int[2])
  res = rbind(res,list(name,"Method_A",resA$estimate,
resA$conf.int[1],resA$conf.int[2]))
  cat(strA,"\n")
  strB = sprintf("%-15 s %-15 s %8.2f %8.2f %8.2f",name,"M
ethod_B",resB$estimate,resB$conf.int[1],resB$conf.int[2])
  res = rbind(res,list(name,"Method_B",as.numeric(resB$e
stimate),resB$conf.int[1],resB$conf.int[2]))
  cat(strB,"\n")
}
printDf = data.frame(res)
names(printDf) = c("DATASET","METHOD","R","CI_LB",
"CI_UB")

# reformat the results for a barplot with error bars
# reformat the r values
ma = as.numeric(printDf[printDf$METHOD=="Method_A",]$R)
```

```

mb = as.numeric(printDf[printDf$METHOD=="Method_B",]$R)
plotDf = data.frame(ma,mb)
row.names(plotDf) = c("subset25","subset50","subset100")
names(plotDf) = c("Method A","Method B")

# reformat the confidence interval lower bound
la = as.numeric(printDf[printDf$METHOD=="Method_A",
]$CI_LB)
lb = as.numeric(printDf[printDf$METHOD=="Method_B",
]$CI_LB)
lbDf = data.frame(la,lb)
row.names(lbDf) = c("subset25","subset50","subset100")
names(lbDf) = c("Method A","Method B")

# reformat the confidence interval upper bound
ua = as.numeric(printDf[printDf$METHOD=="Method_A",
]$CI_UB)
ub = as.numeric(printDf[printDf$METHOD=="Method_B",
]$CI_UB)
ubDf = data.frame(ua,ub)
row.names(ubDf) = c("subset25","subset50","subset100")
names(ubDf) = c("Method A","Method B")

# plot the barplot
par(fig = c(0,0.8,0,1))
barplot2(t(plotDf),beside = T,ci.l = t(lbDf),ci.u =
t(ubDf),plot.ci = TRUE,ylim = c(0,1),ylab = "Pearson r")
# plot the legend
par(fig = c(0.4,1,0,1),new = TRUE)
smartlegend(x = "right",y = "top",legend = c("Method
1","Method 2"),ncol=1,fill=c("red","yellow"))

```

